Image Grid Recognition and Regression for Fast and Accurate Face Detection

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

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

Chao Zhang
Institut für Informatik VI
Technical University of Munich
Garching, Germany
ge37kik@tum.de

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, for high performance face detection, there are still many challenging problems, e.g., the speed-accuracy balance and the performance degradation in adverse conditions. In this paper, by taking advantage of the characteristic of CNN, we propose an effective anchor generation and bounding-box regression method that can make a good balance between speed and accuracy, and also work well in bad conditions. The classic structure of CNN produces pyramid-like feature maps due to the pooling or other downscale operations. According to the size of a feature map, we divide the image into grids. Each grid corresponds to a point in the feature map. We make the corresponding feature point responsible for identifying the content of the grid. If this grid area belongs to the face area, it is a natural anchor for face bounding-box regression. Since this anchor is square, it is reasonable to use it to predict the face bounding-box which is square-like. The points in the lower-level feature map correspond to smaller grids, which are dedicated to predicting the bounding-boxes of smaller faces. The points in the higher-level feature maps correspond to larger grids, which are responsible for predicting the bounding-boxes of larger faces. Hence our method can effectively detect multi-scale faces. With this effectiveness, our method can achieve a high detection accuracy using fewer parameters which leads to a fast detection speed. The experiments demonstrate the effectiveness of our method.

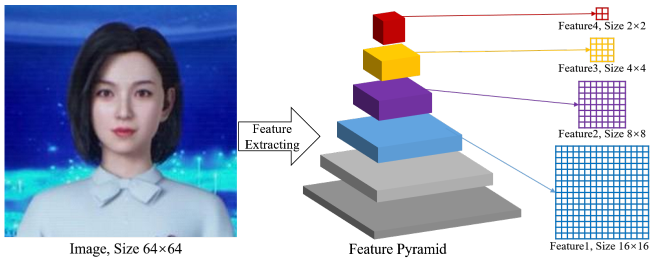
I. Introduction

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 networks are proposed, which results in serious time consumption. In adverse conditions, performance degradation is another problem that hinders the applying of face detection algorithms. In this paper, we propose an effective anchor generation and bounding-box regression method that can obtain a good speed-accuracy balance and also work well in bad conditions.

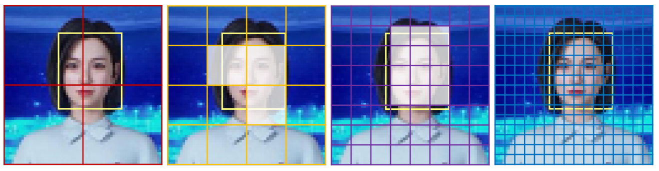
As a special object detection, the pipeline of face detection is similar to that of general object detection. R-CNN [2], the first successful CNN-based object detection method, contains two stages. First, thousands of the candidate regions of objects are proposed by the selective search [3]. Second, each of the candidate regions is cropped from the image and input the CNN to classify what kind of object it contains. For those regions containing needed objects, their positions are refined by bounding-box regression. Fast R-CNN [4] and Faster R-CNN [5] improve R-CNN and focus on selecting candidate regions of objects better and faster. Faster R-CNN designs Region Proposal Networks (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 refined to obtain the candidate region. While the series methods of R-CNN only use a single scale of feature map to detect objects, Feature Pyramid Networks (FPN) [6] utilizes multi-scales feature maps to enhance the network's ability to detect multi-scale objects. To reduce the time consumption, SSD [7] and YOLO [8] combine the region proposal and the region classification to one stage by mapping the anchors and their containing 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 the generation of anchors. Although CNN has experienced considerable development, the most popular networks always have a similar structure with the very original LeNet [9] 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 [10]. 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 with its corresponding feature point's Receptive Field. Each point in the feature map is responsible for determining whether its corresponding square area is foreground. Naturally, the side length of this square area should be the side length ratio between the input image and the feature map. In Fig. 1(a), the size of the input image is 64×64 and the sizes of the four selected feature maps are 16×16, 8×8, 4×4, and 2×2, respectively. Fig. 1(b) shows the assigned areas that the points in the four feature maps correspond to. If a grid area locates at or has a larger part overlapped with the foreground of the input image, we label it as foreground. Otherwise, we label this grid area as background. In Fig. 1(b), the white ones are labeled as foreground, and the other grids are labeled as background. In the first grid image, no grid is selected as foreground because no grid has a larger part overlapped with the ground-truth bounding-box. In the last grid image, the grids are too small compared to the ground-truth bounding-box and the small grids are not used for detecting too large faces. Each grid's corresponding point in the feature map is responsible for recognizing this grid. If a grid is recognized as foreground, this grid is used as an anchor for regressing to the ground-truth bounding-box. The lower-level feature map whose points correspond to smaller grids is responsible for detecting smaller faces, while the higher-level feature map whose points correspond to larger grids is dedicated to detecting larger faces. In testing, the face detection pipeline is shown in Fig. 2. Assume that the foreground and background grids are all classified correctly, then we get 12 anchors. By regressing the 12 anchors to face bounding-boxes, we get 12 predicted bounding-boxes. After Non-Maximum Suppression (NMS) [11], we get the best-predicted bounding-box. Fig. 2 only shows detection on the second-level feature map. The final result comes from the fusion of all the four levels of feature maps.

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 each level of feature map to detect are finer. These enhance the network's ability to detect multi-scale faces. For one feature point, the single square anchor we set is more suitable for the regression of face bounding-boxes and can make better use of the representation ability of the network than the multi-anchors with different aspect ratios and multi-scales set by other methods. Our source code is released on https://github.com/zhouliguo/GRR .



(a) Select Feature Maps for 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 image is divided into grids according to the sizes of the feature maps. Let the side length of a grid be d, if the grid has half part or more overlapped with the ground-truth and the side length of the ground-truth belongs to $[d, d^2]$, the grid is labeled as foreground. The white grids are the labeled foreground grids.

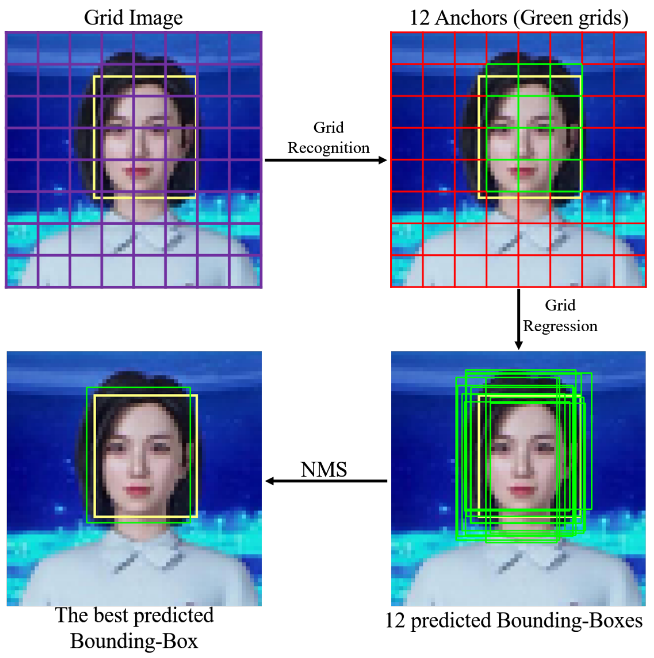


Fig. 2. The detection pipeline using the second-level feature map.

II. Related Works

A. Object detection

As a special object detection task, the progress of face detection benefits from the development of object detection. Since R-CNN [2] 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 [4] and Faster R-CNN [5], all belong to the two-stage. They firstly search the region proposals and then classify the contents in the proposals. SSD [7] and YOLO [8] 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 are assumed to contain objects and regressed to the ground-truth bounding-box. 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 [12], [13]. To improve the accuracy on the popular face benchmarks, FDDB [14] and WIDER FACE [15], loads of complicated components are added in the networks, which result in over-fitting in these benchmarks and loss of speed. PyramidBox [16] proposes a context-assisted single shot face detector. DSFD [17] introduces a feature enhance module to extend the single shot detector to the dual shot detector. EXTD [18] generates the feature maps by iteratively reusing a shared lightweight and shallow backbone network instead of a single backbone network. RetinaFace [19] 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 [20] achieves a better performance on the WIDER FACE benchmark by adding a five-point landmark regression head and using the Wing loss function [21]. Hambox [22] proposes an online strategy to mine high-quality anchors for detecting outer faces.

III. Method

A. Network

Our network is shown in Fig. 3. The feature extracting part includes a backbone, FPN [6] and PAN [23]. Our backbone is similar to the CSPDarkNet which is used in YOLOv5 [24]. CSP [25] and SPP [26] 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 map with the same size.

At the end of the network, there are four branches for recognizing grid area and predicting face bounding-box. 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, so a feature point in Feature1-4 can be denoted as a vector $Z(z_0,\ z_1,\ z_2,\ z_3,\ z_4)$.

B. Grid Area Recognition (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 label each grid for training.

If a grid has half part or more overlapped with the ground-truth bounding-box, we label it as 1, otherwise 0. As shown in Fig. 1(b), all the grids in the first grid image should be labeled as 0. The white grids in the second and third grid images should be labeled as 1, and the other grids should be labeled as 0.

To enhance the ability of our method to detect multi-scale faces, Feature1-4 are used for detecting faces of different sizes. We set the side-length of a grid to d. If the length of the longer side of a ground-truth belongs to $[d, d^2]$, the grid image's corresponding feature map is responsible for detecting this face. For this reason, none of the grids in the last grid image in Fig. 1(b) are labeled as 1.

For grid area recognition, the first element z_0 of a feature point Z in Feature1-4 is used for recognizing whether its corresponding grid area is foreground by

$$p^{'} = \frac{1}{1+e^{-z_0}}, \tag{1}$$

where $p^{'}$ is the probability that the grid belongs to the foreground.

In training, we optimize the recognition loss for each branch. The loss function is defined in

$$L_{cls}^{(d)} = -\frac{1}{N^{(d)}} \sum_{n=1}^{N^{(d)}} [w_0(1-p_n)\log(1-p_n^{'}) + w_1 p_n \log(p_n^{'})], \tag{2}$$

where (d) is used for distinguishing the loss functions in different branches, $N^{(d)}$ is the number of grids that needed to be recognized in one batch and is defined in (3), p_n and p_n' are the true probability and the predicted probability that the n-th grid belongs to the foreground. w_0 and w_1 are set to solve the class imbalance. We set w_0 to 1 and w_1 to the multiple between the number of background grids and the number of foreground grids.

$$N^{(d)} = (\frac{32}{d})^2 nab. \tag{3}$$

In (3), n, a and b are defined in Fig. 3. The loss of the recognition is defined in (4). It is the sum of the losses of the four branches.

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

In testing, those grids that are recognized as having a high probability (e.g. higher than 0.5) of being the foreground are the anchors we selected for face bounding-box regression.

C. Bounding-box Regression

We design a new method of bounding box regression to support our anchor generation method. We code the ground-truth bounding-box first. In the pixel domain of an image, a ground-truth bounding-box can be denoted as (t_x, t_y, t_w, t_h) and a grid is (g_x, g_y, d, d). (t_x, t_y) and (g_x, g_y) are the center points. (t_w, t_h) and (d, d) are the widths and heights. We code the ground-truth of each branch by (5). Since the upper limit of t_w and t_h is d^2, the t_{x_c} and $t_{y\_c} \in (-0.5, 0.5)$. The t_{w_c} and t_{h_c} are transformed to feature domain.

$$\begin{aligned} t_{x\_c} &= \frac{t_x - g_x}{d^2}, \quad & t_{y\_c} &= \frac{t_y - g_y}{d^2}, \\ t_{w\_c} &= \frac{t_w}{d}, & t_{h\_c} &= \frac{t_h}{d}. \end{aligned} \tag{5}$$

The rest elements (z_1, z_2, z_3, z_4) of a feature point Z is used for bounding-box regression. These values are normalized to

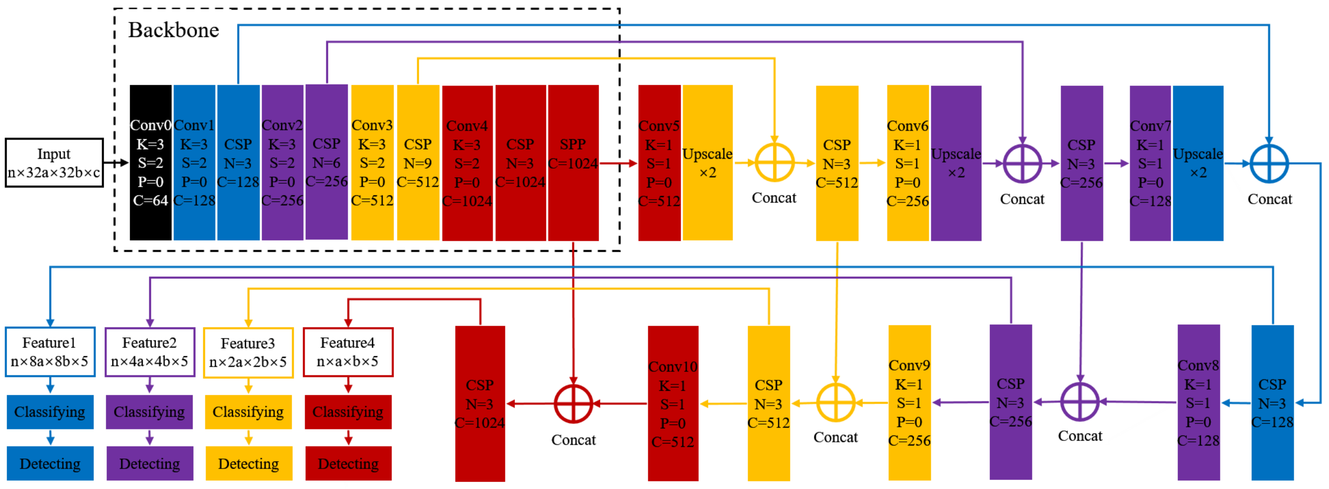


Fig. 3. The network consists of a backbone and a detection head with four branches. The input size is $n \times 32a \times 32b \times c$ where n is batch-size, $32a$, $32b$ and c are height, width and channel. K, S, P, C and N denote the kernel size, stride, padding size, output channel and number of repeated module, respectively.

$(0, 1)$ by (1) and the result is (z'_1, z'_2, z'_3, z'_4). The predicted coded bounding-box $P_c(p_{x\_c}, p_{y\_c}, p_{w\_c}, p_{h\_c})$ in each branch can be got by

$$\begin{aligned} p_{x\_c} &= z'_1 - 0.5, \quad & p_{y\_c} &= z'_2 - 0.5, \\ p_{w\_c} &= \frac{d^{2z'_3}}{d}, & p_{h\_c} &= \frac{d^{2z'_4}}{d}. \end{aligned} \tag{6}$$

Since $z' \in (0, 1)$, p_{x_c} and $p_{y\_c} \in (-0.5, 0.5)$ and $d^{2z'} \in (0, d^2)$. The $P_c(p_{x\_c}, p_{y\_c}, p_{w\_c}, p_{h\_c})$ can be used to present the $T_c(t_{x\_c}, t_{y\_c}, t_{w\_c}, t_{h\_c})$.

We use CIoU [27] as the loss function in training. The regression loss of each branch is defined in

$$L^{(d)}_{reg} = \frac{1}{N^{(d)}} \sum_{n=1}^{N^{(d)}} l_{CIoU}(P_c, T_c), \tag{7}$$

where $N^{(d)}$ is the number of grids that are labeled as foreground in one batch. The loss function of regression is

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

The loss function of the whole network is

$$L = L_{cls} + L_{reg} + \lambda ||W||^2, \tag{9}$$

where W are the weights in the network and $\lambda ||W||^2$ is added to avoid over-fitting. If a grid is predicted as foreground, the predicted bounding-box $P(p_x, p_y, p_w, p_h)$ can be obtained by

$$\begin{aligned} p_x &= (z'_1 - 0.5)d^2 + g_x, \; & p_y &= (z'_2 - 0.5)d^2 + g_y, \\ p_w &= d^{2z'_3}, & p_h &= d^{2z'_4}. \end{aligned} \tag{10}$$

D. Post Processing

In testing, if input an image into the network, each point in Feature1-4 will be transformed into a probability and a bounding-box. We fuse those bounding-boxes with high probability produced in each branch by Non-Maximum Suppression (NMS) and get the best bounding-boxes of each branch. Then we fuse the results of the four branches by NMS to obtain the final predicted bounding-boxes. Fig. 2 shows the process of obtaining the best bounding-boxes in the second branch.

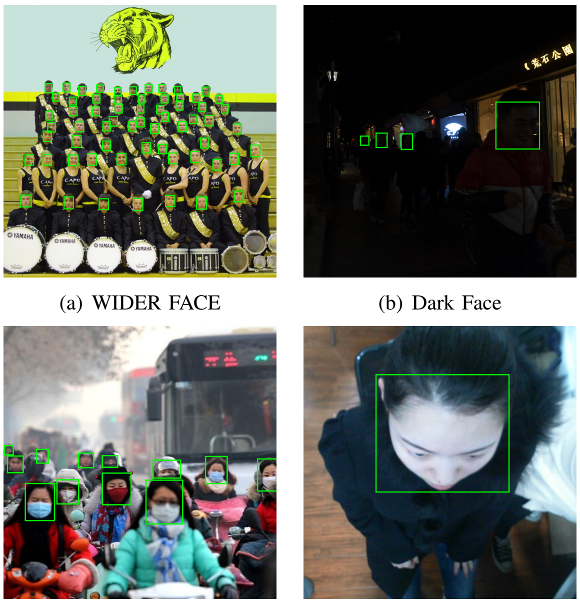
IV. Experiment

Our experiments mainly consist of two parts. One is training large models using the network in Fig. 3 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. 3 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 [28]. 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 [29] until 0.001. The optimizer used in the training is stochastic gradient descent [9].

A. Datasets

WIDER FACE [15], Dark Face [30], MAFA [31], and TFD [32] 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 mask. TFD contains 11,124 face images from 927 subjects, covering a variety of tilt angles on the overhead view. Fig. 4 shows the examples of these datasets.



(a) WIDER FACE (b) Dark Face (c) MAFA (d) TFD

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

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 I, the performances of the detection methods with our detection head are better.

TABLE I
COMPARISON OF ACCURACY WITH DIFFERENT DETECTION HEADS (WIDER FACE VAL SET)

Methods	**Average Precision**			**FLOPs** ($\times 10^9$)
	Easy	Medium	Hard	
EfficientDet-D4 [33]	0.938	0.922	0.823	40.4
EfficientDet-D4-Ours	0.944	0.940	0.900	45.1
YOLOv3 [8]	0.966	0.959	0.897	154.9
YOLOv3-Ours	0.969	0.962	0.916	144.8
YOLOv5x [24]	0.969	0.960	0.901	204.2
YOLOv5x-Ours	**0.970**	**0.964**	**0.924**	200.0

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

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, MAFA and TFD, the open-source face detection methods DSFD [17], EXTD [18] and PyramidBox [16] as well as the state-of-the-art object detection methods YOLOv3 [8], EfficientDet [33] and YOLOv5x [24] are selected. Table III 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 IV shows our method is faster than the other methods at each resolution.

TABLE II
COMPARISON OF ACCURACY ON FACES WITH DIFFERENT SIZES (WIDER FACE VAL SET)

	Average Precision			
Longer Side of GT BBox	$\leq$16	(16, 64]	(64, 256]	>256
Number of GT BBox	16844	17793	4482	586
PyramidBox [16]	0.566	0.898	0.954	0.934
EXTD [18]	0.507	0.862	0.917	0.917
DSFD [17]	0.534	0.917	0.968	0.951
YOLOv3 [8]	0.569	0.919	0.967	0.874
EfficientDet-D4 [33]	0.356	0.846	0.936	0.941
YOLOv5x [24]	0.579	0.922	0.969	0.895
Ours	**0.685**	**0.928**	**0.971**	**0.956**

TABLE III
COMPARISON OF ACCURACY ON DARK FACE, MAFA AND TFD

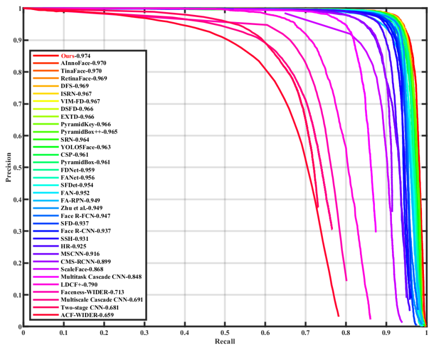
Methods	**Average Precision**		
	Dark Face	MAFA	TFD
PyramidBox [16]	0.796	0.748	0.994
EXTD [18]	0.689	0.661	0.942
DSFD [17]	0.634	0.772	0.989
EfficientDet-D4 [33]	0.755	0.753	0.956
YOLOv3 [8]	0.801	0.773	0.990
YOLOv5x [24]	0.824	0.788	0.995
Ours	**0.852**	**0.807**	**0.997**

TABLE IV
COMPARISON OF DETECTION SPEED ON NVIDIA V100 GPU

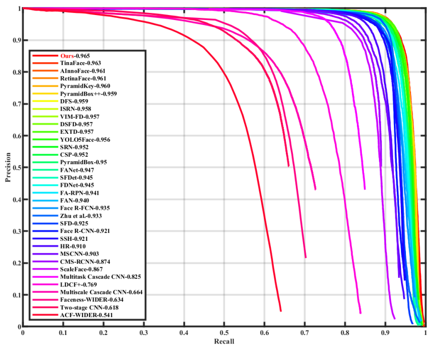
Methods	**Layers**	**Params**	**FLOPs**	**Speed** (FPS)		
		($\times 10^6$)	($\times 10^9$)	640×480	1280×720	1920×1080
PyramidBox	234	67.269	296.2	2.75	1.99	1.40
EXTD	93	0.162	27.9	6.07	4.24	2.45
DSFD	544	120.058	691.6	3.48	3.36	2.37
YOLOv3	333	61.524	154.9	25.27	22.94	22.39
EfficientDet-D4	1111	20.543	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

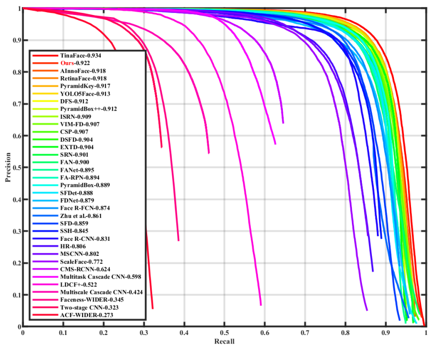
By reducing the channels of each Conv layer in Fig. 3 to 1/4 and setting the module number of each CSP block to 1, we can get a light network that can work on Non-GPU devices. We convert the existing light face detection models, retrained YOLOv5n [24] model and ours to ONNX [34] format and run them on Intel i7-5930K CPU for comparison. The images in the WIDER FACE Val set are resized to 640×480 for the test. Table V shows our method can run on a Non-GPU device with real-time speed while keeping a high detection accuracy.



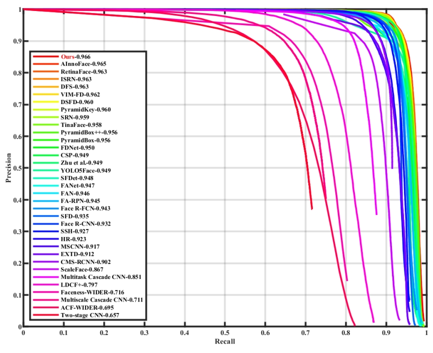
(a) Val-Easy



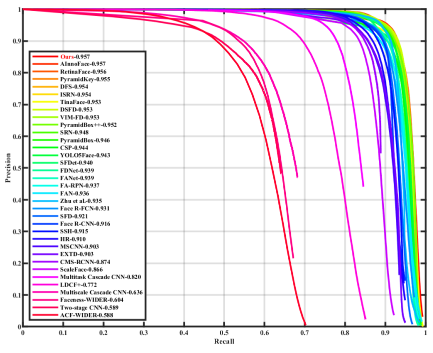
(b) Val-Medium



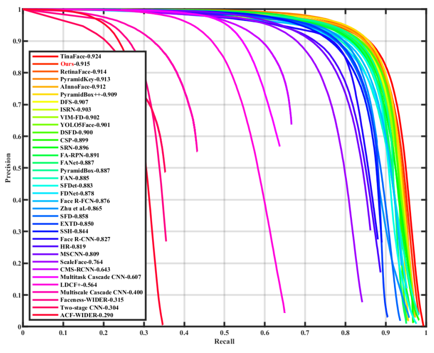
(c) Val-Hard



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

Methods	**Layers**	**Params** ($\times10^6$)	**FLOPs** ($\times10^9$)	**Average Precision**			**Latency** (ms)	
				Easy	Medium	Hard	Forward	Post-Proc
FaceBoxes [35]	33	1.013	1.541	0.845	0.777	0.404	16.52	7.16
ULFG-slim-640 [36]	42	0.401	2.000	0.810	0.794	0.630	19.03	2.37
ULFG-RFB-640 [36]	52	0.401	2.426	0.816	0.802	0.663	21.27	1.90
YuFaceDetectNet [37]	43	0.085	2.549	0.856	0.842	0.727	23.47	32.81
LFFD-v2 [38]	45	1.520	37.805	0.875	0.863	0.752	178.47	6.70
LFFD-v1 [38]	65	2.282	55.555	0.910	0.880	0.778	229.35	10.08
YOLOv5n [24]	270	1.872	4.520	**0.942**	**0.933**	0.856	29.21	0.80
Ours	301	1.746	4.536	0.935	**0.933**	**0.875**	24.15	0.69

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, masked faces, and faces with extreme tilt. Our method takes the 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.

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] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[9] 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.

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

[11] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4507–4515.

[12] J. Guo, J. Deng, A. Lattas, and S. Zafeiriou, "Sample and computation redistribution for efficient face detection," *arXiv preprint arXiv:2105.04714*, 2021.

[13] Y. Zhu, H. Cai, S. Zhang, C. Wang, and Y. Xiong, "Tinaface: Strong but simple baseline for face detection," *arXiv preprint arXiv:2011.13183*, 2020.

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

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

[24] G. Jocher, "ultralytics/yolov5," https://github.com/ultralytics/yolov5, Oct. 2020.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] N. Wang, Z. Wang, Z. He, B. Huang, L. Zhou, and Z. Han, "A tilt-angle face dataset and its validation," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 894–898.

[33] 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.

[34] "Open neural network exchange," https://github.com/onnx/onnx.

[35] 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.

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

[37] S. Yu, "libfacedetection," https://github.com/ShiqiYu/libfacedetection, 2021.

[38] 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.