YOLO5Face: Why Reinventing a Face Detector

Delong Qi[1], Weijun Tan[1,2,3](✉), Qi Yao[2], and Jingfeng Liu[2]

[1] Shenzhen Deepcam, Shenzhen, China
{delong.qi,weijun.tan}@deepcam.com
[2] Jovision-Deepcam Research Institute, Shenzhen, China
{sz.twj,sz.yaoqi,sz.ljf}@jovision.com, {qi.yao,jingfeng.liu}@deepcam.com
[3] LinkSprite Technology, Longmont, CO 80503, USA
weijun.tan@linksprite.com

Abstract. Tremendous progress has been made on face detection in recent years using convolutional neural networks. While many face detectors use designs designated for the detection of face, we treat face detection as a general object detection task. We implement a face detector based on YOLOv5 object detector and call it YOLO5Face. We add a five-point landmark regression head into it and use the Wing loss function. We design detectors with different model sizes, from a large model to achieve the best performance to a super small model for real-time detection on an embedded or mobile device. Experiment results on the WiderFace dataset show that our face detectors can achieve state-of-the-art performance in almost all the Easy, Medium, and Hard subsets, exceeding the more complex designated face detectors. The code is available at https://github.com/deepcam-cn/yolov5-face.

Keywords: Face detection · Convolutional neural network · YOLO · Real-time · Embedded device · Object detection

1 Introduction

Face detection is a very important computer vision task. Tremendous progress has been made since deep learning, particularly convolutional neural network (CNN), has been used in this task. As the first step of many tasks, including face recognition, verification, tracking, alignment, expression analysis, face detection attracts a lot of research and developments in academia and industry. And the performance of face detection has improved significantly over the years. For a survey of the face detection, please refer to the benchmark results [39,40]. There are many methods in this field from different perspectives. Research directions include the design of CNN networks, loss functions, data augmentations, and training strategies. For example, in the YOLOv4 paper, the authors explore all these research directions and propose the YOLOV4 object detector based on optimizations of network architecture, selection of bags of freebies, and selection of bags of specials [1].

In our approach, we treat face detection as a general object detection task. We have the same intuition as the TinaFace [55]. Intuitively, a face is an object. As discussed in the TinaFace [55], from the perspective of data, the properties that a face has, like pose, scale, occlusion, illumination, blur, etc., also exist in other objects. The unique properties of faces, like expression and makeup, can also correspond to distortion and color in objects. Landmarks are special to face, but they are not unique either. They are just key points of an object. For example, in license plate detection, landmarks are also used. And adding landmark regression in the object prediction head is straightforward. Then from the perspective of challenges encountered by face detection like multi-scale, small faces, and dense scenes, they all exist in generic object detection. Thus, face detection is just a sub-task of general object detection.

In this paper, we follow this intuition and design a face detector based on the YOLOv5 object detector [42]. We modify the design for face detection considering large faces, small faces, and landmark supervision for different complexities and applications. Our goal is to provide a portfolio of models for different applications, from very complex ones to get the best performance to very simple ones to get the best trade-off of performance and speed on embedded or mobile devices.

Our main contributions are summarized as following,

- We redesign the YOLOV5 object detector [42] as a face detector and call it YOLO5Face. We implement key modifications to the network to improve the performance in terms of mean average precision (mAP) and speed. The details of these modifications will be presented in Sect. 3.
- We design a series of models of different model sizes, from large models to medium models, to super small models, for needs in different applications. In addition to the backbone used in YOLOv5 [42], we implement a backbone based on ShuffleNetV2 [26], which gives the state-of-the-art (SOTA) performance and fast speed for a mobile device.
- We evaluate our models on the WiderFace [40] dataset. On VGA resolution images, almost all our models achieve the SOTA performance and fast speed. This proves our goal; as the tile of this paper claims, we do not need to reinvent a face detector since the YOLO5Face can accomplish it.

Please note that our work contributes not to the novelty of new ideas but the engineering community of face detection. Since open source, our code has been used widely in many projects.

2 Related Work

2.1 Object Detection

General object detection aims at locating and classifying the pre-defined objects in a given image. Before deep CNN is used, traditional face detection uses hand-crafted features, like HAAR, HOG, LBP, SIFT, DPM, ACF, etc. The seminal work by Viola and Jones [51] introduces integral image to compute HAAR-like features. For a survey of face detection using hand-crafted features, please refer to [38,52].

Since deep CNN shows its power in many machine learning tasks, face detection is dominated by deep CNN methods. There are two-stage and one-stage object detectors. Typical two-stage methods are the RCNN family, including RCNN [12], fast-RCNN [11], faster-RCNN [31], mask-RCNN [15], Cascade-RCNN [2].

The two-stage object detectors have very good performance but suffer from long latency and slow speed. To overcome this problem, one-stage object detectors are studied. Typical one-stage networks include SSD [23], YOLO [1,28–30,42].

Other object detection networks include FPN [21], MMDetection [4], EfficientDet [33], transformer (DETR) [3], Centernet [8,54], and so on.

2.2 Face Detection

The research for face detection follows general object detection. After the most popular and challenging face detection benchmark WiderFace dataset [40] is released, face detection develops rapidly, focusing on the extreme and real variation problems including scale, pose, occlusion, expression, makeup, illumination, blur, etc.

A lot of methods are proposed to deal with these problems, particularly the scale, context, and anchor in order to detect small faces. These methods include MTCNN [44], FaceBox [48], S3FD [47], DSFD [19], RetinaFace [7], RefineFace [45], and the most recent ASFD [43], MaskFace [41], TinaFace [55], MogFace [24], and SCRFD [13]. For a list of popular face detectors, the readers are referred to the WiderFace website [39].

It is worth noting that some of these face detectors explore unique characteristics of a human face; the others are just general object detectors adopted and modified for face detection. Use RetinaFace [7] as an example. It uses landmark (2D and 3D) regression to help the supervision of face detection, while TinaFace [55] is simply a general object detector.

2.3 YOLO

YOLO first appeared in 2015 [29] as a different approach than popular two-stage approaches. It treats object detection as a regression problem rather than a classification problem. It performs all the essential stages to detect an object using a single neural network. As a result, it achieves not only very good detection performance but also achieves real-time speed. Furthermore, it has excellent generalization capability, and can be easily trained to detect different objects.

Over the next five years, the YOLO algorithm has been upgraded to five versions with many innovative ideas from the object detection community. The first three versions - YOLOv1 [29], YOLOv2 [30], YOLOv3 [28] are developed by the author of the original YOLO algorithm. Out of these three versions, the YOLOv3 [28] is a milestone with big improvements in performance and speed by introducing multi-scale features (FPN) [21], better backbone network (Darknet53), and replacing the Softmax classification loss with the binary cross-entropy loss.

In early 2020, after the original YOLO authors withdrew from the research field, YOLOv4 [1] was released by a different research team. The team explores a lot of options in almost all aspects of the YOLOv3 [28] algorithm, including the backbone and what they call bags of freebies and bags of specials. It achieves 43.5% AP (65.7% AP50) for the MS COCO dataset at a real-time speed of 65 FPS on Tesla V100.

One month later, the YOLOv5 [42] was released by another different research team. From the algorithm perspective, the YOLOv5 [42] does not have many innovations. And the team does not publish a paper. These bring quite some controversies about if it should be called YOLOv5. However, due to its significantly reduced model size, faster speed, and similar performance as YOLOv4 [1], and full implementation in Python (Pytorch), it is welcome by the object detection community.

3 YOLO5Face Face Detector

Summary of Key Modifications. In this section, we present the key modifications we make in YOLOv5 and make it a face detector - YOLO5Face. 1) We add a landmark regression head to the YOLOv5 network. 2) We replace the Focus layer of YOLOv5 [42] with a Stem block structure [37]. 3) We change the SPP block [16] and use a smaller kernel. 4) We add a P6 output block with a stride of 64. 5) We optimize the data augmentation methods for face detection. 6) We design two super light-weight models based on ShuffleNetV2 [26].

3.1 Network Architecture

We use the YOLOv5 object detector [42] as our baseline and optimize it for face detection. The network architecture of our YOLO5Face face detector is depicted in Fig. 1. It consists of the backbone, neck, and head. In YOLOv5, a newly designed backbone called CSPNet [42] is used. In the neck, an SPP [16] and a PAN [22] are used to aggregate the features. In the head, regression and classification are both used.

In Fig. 1(a), the overall network architecture is depicted, where a newly added P6 block is highlighted with a box. In Fig. 1(b), a key block called CBS is defined, which consists of a Conv layer, BN layer, and a SILU [9] activation function. This CBS block is used in many other blocks. In Fig. 1(c), an output label for the head is shown, which includes the bounding box (bbox), confidence (conf),

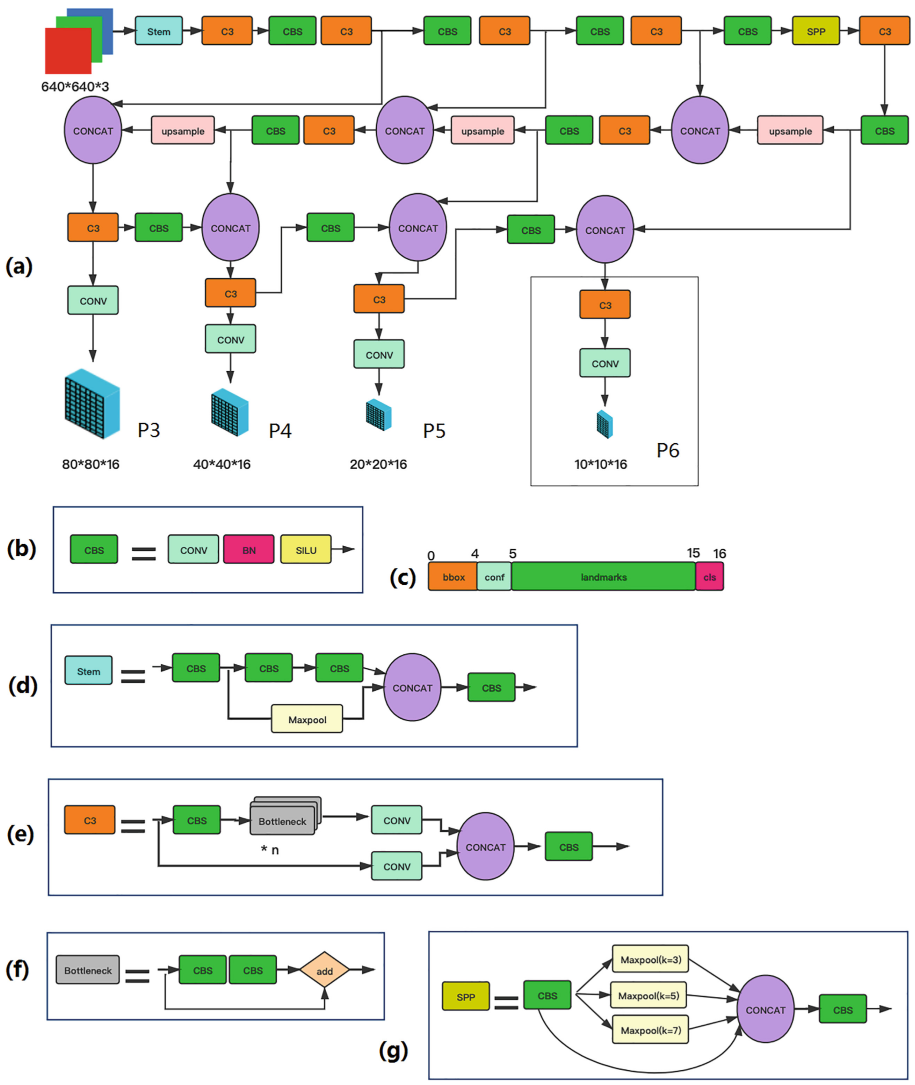

Fig. 1. The proposed YOLO5Face network architecture.

classification (cls), and five-point landmarks. The landmarks are our addition to the YOLOv5 to make it a face detector with landmark output. If without the landmark, the last dimension 16 should be 6. Please note that the output dimensions $80 * 80 * 16$ in P3, $40 * 40 * 16$ in P4, $20 * 20 * 16$ in P5, and $10 * 10 * 16$ in optional P6 are for every anchor. The real dimension should be multiplied by the number of anchors.

In Fig. 1(d), a Stem structure [37] is shown, which is used to replace the original Focus layer in YOLOv5. The introduction of the Stem block into YOLOv5 for face detection is one of our innovations.

In Fig. 1(e), a CSP block (C3) is shown. This block is inspired by the DenseNet [18]. However, instead of adding the full input and the output after some CNN layers, the input is separated into two halves. One half is passed through a CBS block, a number of Bottleneck blocks, which is shown in Fig. 1(f), the another half is sent to a Conv layer. The outputs are then concatenated, followed by another CBS block.

Figure 1(g), an SPP block [16] is shown. In this block, the three kernel sizes 13×13, 9×9, and 5×5 in YOLOv5 are revised to 7×7, 5×5, and 3×3 in our face detector. This has been shown as one of the innovations that improve face detection performance.

Note that we only consider VGA-resolution input images. The longer edge of the input image is scaled to 640, and the shorter edge is scaled accordingly. The shorter edge is also adjusted to be a multiple of the largest stride of the SPP block. For example, when P6 is not used, the shorter edge needs to be a multiple of 32; when P6 is used, the shorter edge needs to be a multiple of 64.

3.2 Landmark Regression

Landmarks are important characteristics of a human face. They can be used to do face alignment, face recognition, face express analysis, age analysis, etc. Traditional landmarks consist of 68 points. They are simplified to 5 points in MTCNN [44] Since then, the five-point landmarks have been used widely in face recognition. The quality of landmarks affects the quality of face alignment and face recognition.

The general object detector does not include landmarks. It is straightforward to add it as a regression head. Therefore, we add it to our YOLO5Face. The landmark outputs will be used in aligning face images before they are sent to the face recognition network.

General loss functions for landmark regression are L2, L1, or smooth-L1. The MTCNN [44] uses the L2 loss function. However, it has been found these loss functions are not sensitive to small errors. To overcome this problem, the Wing-loss is proposed [10],

$$wing(x) = \begin{cases} w \cdot ln(1 + |x|/e), & \text{if } x < w \\ |x| - C, & \text{otherwise} \end{cases} \tag{1}$$

where x is the error signal. The non-negative w sets the range of the nonlinear part to $(-w, w)$, e limits the curvature of the nonlinear region, and $C = w - wln(1 + w/e)$ is a constant that smoothly links the piecewise-defined linear and nonlinear parts. Plotted in Fig. 2 is this Wing loss function with different parameters w and e. It can be seen that the response at a small error area near zero is boosted compared to the L2, L1, or smooth-L1 functions.

The loss functions for landmark point vector $s = \{s_i\}$, and its ground truth $s' = \{s_i\}$, where $i = 1, 2, ..., 10$, is defined as,

$$loss_L(s) = \sum_i wing(s_i - s'_i) \tag{2}$$

Let the general object detection loss function of YOLOv5 be $loss_O(bounding\_box, class, probability)$, then the new total loss function is,

$$loss(s) = loss_O + \lambda_L \cdot loss_L \tag{3}$$

where the λ_L is a weighting factor for the landmark regression loss function.

3.3 Stem Block Structure

We use a stem block similar to [37]. The stem block is shown in Fig. 1(d). With this stem block, we implement a stride = 2 in the first spatial down-sampling on the input image and increase the number of channels. With this stem block, the computation complexity only increases marginally, while a strong representation capability is ensured.

3.4 SPP with Smaller Kernels

Before forwarding to the feature aggregation block in the neck, the output feature maps of the YOLO5 backbone are sent to an additional SPP block [16] to increase the receptive field and separate out the most important features. Instead of many CNN models containing fully connected layers which only accept input images of specific dimensions, SPP is proposed to aim at generating a fixed-size output irrespective of the input size. In addition, SPP also helps to extract important features by pooling multi-scale versions of itself.

In YOLO5, three kernel sizes 13×13, 9×9, 5×5 are used [42]. We revise them to use smaller size kernels 7×7, 5×5, and 3×3. These smaller kernels help to detect small faces more efficiently, and increase the overall face detection performance.

3.5 P6 Output Block

The backbone of the YOLO object detector has many layers. As the feature becomes more and more abstract as the layers go deeper, the spatial resolution of feature maps decreases due to downsampling, which leads to a loss of spatial information as well as fine-grained features. In order to preserve these fine-grained features, the FPN [21] is introduced to YOLOv3 [28].

In FPN [21], the fine-grained features take a long path traveling from low-level to high-level layers. To overcome this problem, the PAN is proposed to add a bottom-up augmentation path along the top-down path used in FPN. In addition, in the connection of the feature maps to the lateral architecture, the element-wise addition operation is replaced with concatenation. In FPN, object predictions are made independently on different scale levels, which do not utilize information from other feature maps and may produce duplicated predictions. In PAN [22], the output feature maps of the bottom-up augmentation pyramid are fused by using (Region of Interest) ROI align and fully connected layers with the element-wise max operation.

In YOLOv5, there are three output blocks in the PAN output feature maps, called P3, P4, and P5, corresponding to $80 \times 80 \times 16$, $40 \times 40 \times 16$, $20 \times 20 \times 16$, with strides 8, 16, 32, respectively. In our YOLO5Face, we add an extra P6 output block, whose feature map is $10 \times 10 \times 16$ with stride 64. This modification particularly helps the detection of large faces. While almost all face detectors focus on improving the detection of small faces, the detection of large faces can be easily overlooked. We fill this hole by adding the P6 output block.

3.6 ShuffleNetV2 as Backbone

The ShuffleNet [49] is an extremely efficient CNN for a mobile device. The key block is called the ShuffleNet block. It utilizes two new operations, pointwise group convolution and channel shuffle, to greatly reduce computation cost while maintaining accuracy.

The ShuffleNetv2 [49] is an improved version of ShuffleNet. It borrows the shortcut network architecture similar to the DenseNet [18], and element-wise addition is changed to concatenation, similar to the change in PAN [22] in YOLOv5 [42]. But different from DenseNet, ShuffleNetV2 does not densely concatenate, and after the concatenation, channel shuffling is used to mix the features. This makes the ShuffleNetV2 a super-fast network.

We use the ShuffleNetV2 as the backbone in YOLOv5 and implement super small face detectors YOLOv5n-Face, and YOLOv5n0.5-Face.

4 Experiments

4.1 Dataset

The WiderFace dataset [40] is the largest face detection dataset, which contains 32,203 images and 393,703 faces. For its large variety of scale, pose, occlusion, expression, illumination, and event, it is close to reality and is very challenging.

The whole dataset is divided into train/validation/test sets by ratio 50%/10%/40% within each event class. Furthermore, each subset is defined into three levels of difficulty: Easy, Medium, and Hard. As its name indicates, the Hard subset is the most challenging. So the performance on the Hard subset reflects best the effectiveness of a face detector.

Unless specified otherwise, the WiderFace dataset [40] is used in this work. In the face recognition with YOLO5Face landmark and alignment, the Webface dataset [56] is used. The FDDB dataset [36] is used in testing to demonstrate our model's performance on cross-domain datasets.

Table 1. Detail of implemented YOLO5Face models, where (D, W) are the depth and width multiples of the YOLOv5 CSPNet [42]. The number of parameters and Flops are listed in Table 4.

Model	Backbone	(D, W)	With P6?
YOLOv5s	YOLO5-CSPNet [42]	(0.33, 0.50)	No
YOLOv5s6	YOLO5-CSPNet	(0.33, 0.50)	Yes
YOLOv5m	YOLO5-CSPNet	(0.50, 0.75)	No
YOLOv5m6	YOLO5-CSPNet	(0.50, 0.75)	Yes
YOLOv5l	YOLO5-CSPNet	(1.0, 1.0)	No
YOLOv5l6	YOLO5-CSPNet	(1.0, 1.0)	Yes
YOLOv5n	ShuffleNetv2 [26]	-	No
YOLOv5n-0.5	ShuffleNetv2-0.5 [26]	-	No

4.2 Implementation Details

We use the YOLOv5-4.0 codebase [42] as our starting point and implement all the modifications we describe earlier in PyTorch.

The SGD optimizer is used. The initial learning rate is 1E−2, the final learning rate is 1E−5, and the weight decay is 5E−3. A momentum of 0.8 is used in the first three warming-up epochs. After that, the momentum is changed to 0.937. The training runs 250 epochs with a batch size of 64. The $\lambda_L = 0.5$ is optimized by exhaust search.

Implemented Models. We implement a series of face detector models, as listed in Table 1. We implement eight relatively large models, including extra large-size models (YOLOv5x, YOLOv5x6), large-size models (YOLOv5l, YOLOv5l6), medium-size models (YOLOv5m, YOLOv5m6), and small-size models (YOLOv5s, YOLOv5s6). In the name of the model, the last postfix 6 means it has the P6 output block in the SPP. These models all use the YOLOv4 CSPNet as the backbone with different depth and width multiples, denoted as D and W in Table 1.

Furthermore, we implement two super small-size models, YOLOv5n and YOLOv5n0.5, which use the ShuffleNetv2 and ShuffleNetv2-0.5 [26] as the backbone. Except for the backbone, all other main blocks, including the stem block, SPP, and PAN, are the same as in the larger models.

The number of parameters and number of flops of all these models is listed in Table 4 for comparison with existing methods.

4.3 Ablation Study

In this subsection, we present the effects of the modifications we have in our YOLO5Face. In this study, we use the YOLO5s model. We use the WiderFace [40] validation dataset and use the mAP as the performance metric. The results are presented in Table 2. Please note we do the experiments incrementally, where we add a modification at a time.

Table 2. Ablation study results on the WiderFace validation dataset.

Modification	Easy	Medium	Hard
Baseline (focus block)	94.70	92.90	83.27
+ Stem block	94.46	92.72	83.57
+ Landmark	94.47	92.79	83.21
+ SPP (3, 5, 7)	94.66	92.83	83.33
+ Ignore small faces	94.71	93.01	83.33
+ P6 block	95.29	93.61	83.27

Stem Block. We use the standard YOLOv5 [42] as a baseline which has a focus layer. By replacing the focus layer with our stem block, the mAP on the hard subset increased by 0.30%, which is non-trivial. The mAPs on the easy and medium subsets degrade slightly.

Landmark. Landmarks are used in many applications, including face recognition, where the landmarks are used to align the detected face image. Using the stem block as a baseline, adding the landmark achieves about the same performance on the easy and medium subsets while causing a little performance loss on the hard dataset. We will show in the next subsection the benefits of our landmarks to face recognition.

SPP with Smaller Size Kernels. The stem block and landmark are used in this test. The SPP kernel sizes (13×13, 9×9, 5×5) are changed to (7×7, 5×5, 3×3). mAPs are improved in all easy, medium, and hard subsets. The mAPs are improved by 0.1–0.2%.

Data Augmentation. A few data augmentation methods are studied. We find that ignoring small faces combined with random crop and Mosaic [1] helps the mAPs. It improves the mAPs on the easy and medium subset by 0.1–0.2%.

P6 Output Block. The P6 block brings significant mAP improvements, about 0.5–0.6%, on the easy and medium subsets. Perhaps motivated by our findings, the P6 block is added to YOLOv5 [42] after us.

Please note that since we do not use exactly the same parameters in this ablation study as in the final benchmark tests, the results in Table 2 may be slightly different from that in Table 4.

4.4 YOLO5Face for Face Recognition

Landmark is critical for face recognition accuracy. In RetinaFace [7], the accuracy of the landmark is evaluated with the MSE between estimated landmark coordinates and their ground truth and with the face recognition accuracy. The results show that the RetinaFace has better landmarks than the older MTCNN [44].

In this work, we use the face recognition framework in [27] to evaluate the accuracy of landmarks of the YOLO5Face. We use the Webface test dataset, which

Table 3. Evaluation of YOLO5Face landmark on Webface test dataset [56]. Two YOLO5Face models, the small model YOLOv5s and the medium model YOLOv5m, are used.

Backbone	FaceDetect	Training dataset	FNMR
R124	RetinaFace [7]	WiderFace [40]	0.1065
R124	YOLOv5s	WiderFace	0.1060
R124	YOLOv5s	+Multi-task facial [53]	0.1058
R124	YOLOv5m	WiderFace	0.1056
R124	YOLOv5m	+Multi-task facial	0.1051
R124||R152	YOLOv5m	All above	0.1021
R152||R200	YOLOv5m	All above	**0.0923**

is the largest face dataset with noisy 4M identities/260M faces, and cleaned 2M identities/42M faces [56]. This dataset is used in the ICCV2021 Masked Face Recognition (MFR) challenge [57]. In this challenge, both masked face images and standard face images are included, and a metric False Non-Match Rate (FNMR) at False Match Rate (FMR) = 1E−5 is used. The FNMR*0.25 for MFR plus FNMR*0.75 for standard face recognition are combined as the final metric.

By default, the RetinaFace [17] is used as the face detector on the dataset. We compare our YOLO5Face with the RetinaFace on this dataset. We use ArcFace [6] framework with Resnet124 or larger backbones [14] as backbone. We also explore using concatenated features from two parallel models to get better performance. We replace the RetinaFace with our YOLO5Face. We test two models, a small model YOLOv5s and a medium model YOLOv5m. More details can be found in [27].

The results are listed in Table 3. From the results, we see that both our small and medium models outperform the RetinaFace [7]. In addition, we notice that there are very few large face images in the WiderFace dataset, so we add some large face images from the Multi-task-facial dataset [53] into the YOLO5Face training dataset. We find that this technique improves face recognition performance shown in Fig. 3 are some detected Webface [56] faces and landmarks using the RetinaFace [7] and our YOLOv5m. On the faces of a large pose, we can visually observe that our landmarks are more accurate, which has been proved in our face recognition results shown in Table 3.

4.5 YOLO5Face on WiderFace Dataset

We compare our YOLO5Face with many existing face detectors on the WiderFace dataset. The results are listed in Table 4, where the previous SOTA results and our best results are both highlighted.

We first look at the performance of relatively large models whose number of parameters is larger than 3M and the number of flops is larger than 5G. All existing methods achieve mAP in 94.27–96.06% on the Easy subset, 91.9–94.92% on the Medium subset, and 71.39–85.29% on the Hard subset. The most recently released SCRFD [13] achieves the best performance in all subsets. Our YOLO5Face (YOLOv5x6) achieves 96.67%, 95.08%, and 86.55% on the three subsets, respectively. We achieve the SOTA performance on all the Easy, Medium, and Hard subsets.

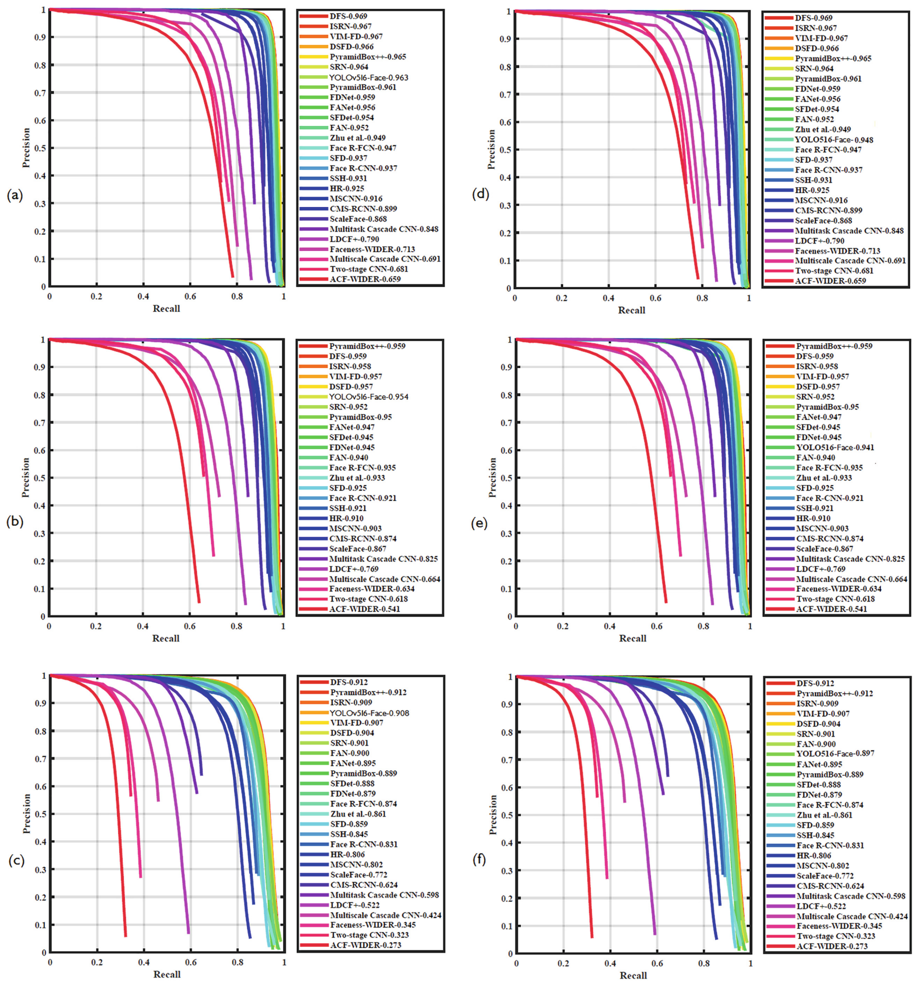

Fig. 2. The precision-recall (PR) curves of face detectors, (a) validation-Easy, (b) validation-Medium, (c) validation-Hard, (d) test-Easy, (e) test-Medium, (f) test-Hard.

Table 4. Comparison of our YOLO5Face and existing face detectors on the WiderFace validation dataset [40].

Detector	Backbone	Easy	Medium	Hard	Params (M)	Flops (G)
DSFD [19]	ResNet152 [14]	94.29	91.47	71.39	120.06	259.55
RetinaFace [7]	ResNet50 [14]	94.92	91.90	64.17	29.50	37.59
HAMBox [25]	ResNet50 [14]	95.27	93.76	76.75	30.24	43.28
TinaFace [55]	ResNet50 [14]	95.61	94.25	81.43	37.98	172.95
SCRFD-34GF [13]	Bottleneck ResNet	**96.06**	**94.92**	**85.29**	9.80	34.13
SCRFD-10GF [13]	Basic ResNet [14]	95.16	93.87	83.05	3.86	9.98
Our YOLOv5s	YOLOv5-CSPNet [42]	94.33	92.61	83.15	7.075	5.751
Our YOLOv5s6	YOLOv5-CSPNet	95.48	93.66	82.8	12.386	6.280
Our YOLOv5m	YOLOv5-CSPNet	95.30	93.76	85.28	21.063	18.146
Our YOLOv5m6	YOLOv5-CSPNet	95.66	94.1	85.2	35.485	19.773
Our YOLOv5l	YOLOv5-CSPNet	95.9	94.4	84.5	46.627	41.607
Our YOLOv5l6	YOLOv5-CSPNet	96.38	94.90	85.88	76.674	45.279
Our YOLOv5x6	YOLOv5-CSPNet	**96.67**	**95.08**	**86.55**	141.158	88.665
SCRFD-2.5GF [13]	Basic Resnet	**93.78**	**92.16**	**77.87**	0.67	2.53
SCRFD-0.5GF [13]	Depth-wise Conv	90.57	88.12	68.51	0.57	0.508
RetinaFace [7]	MobileNet0.25 [32]	87.78	81.16	47.32	0.44	0.802
FaceBoxes [48]	-	76.17	57.17	24.18	1.01	0.275
Our YOLOv5n	ShuffleNetv2 [26]	**93.61**	**91.54**	**80.53**	1.726	2.111
Our YOLOv5n0.5	ShuffleNetv2-0.5 [26]	90.76	88.12	73.82	0.447	0.571

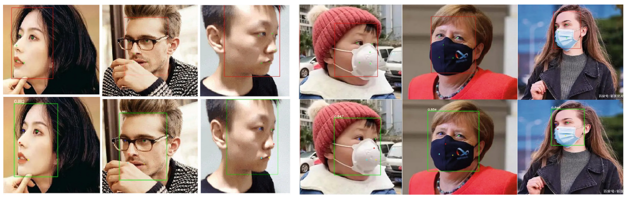

Fig. 3. Some examples of detected face and landmarks, where the first row is from RetinaFace [7], and the second row is from our YOLOv5m.

Next, we look at the performance of super small models whose number of parameters is less than 2M and the number of flops is less than 3G. All existing methods achieve mAP in 76.17–93.78% on the Easy subset, 57.17–92.16% on the Medium subset, and 24.18–77.87% on the Hard subset. Again, the SCRFD [13] achieves the best performance in all subsets. Our YOLO5Face (YOLOv5n) achieves 93.61%, 91.54%, and 80.53% on the three subsets, respectively. Our face detector has a little bit worse performance than the SCRFD [13] on the Easy and Medium subsets. However, on the Hard subset, our face detector is leading by 2.66%. Furthermore, our smallest model, YOLOv5n0.5, has good performance, even though its model size is much smaller.

Table 5. Evaluation of YOLO5Face on the FDDB dataset [36].

Method	MAP
ASFD [43]	**0.9911**
RefineFace [45]	**0.9911**
PyramidBox [34]	0.9869
FaceBoxes [48]	0.9598
Our YOLOv5s	0.9843
Our YOLOv5m	0.9849
Our YOLOv5l	0.9867
Our YOLOv5l6	0.9880

The precision-recall (PR) curves of our YOLO5Face face detector, along with the competitors, are shown in Fig. 2. The leading competitors include DFS [35], ISRN [46], VIM-FD [50], DSFD [19], PyramidBox++ [20], SRN [5], PyramidBox [34] and more. For a full list of the competitors and their results on the WiderFace [40] validation and test datasets, please refer to [39]. In the results on the validation dataset, our YOLOv5x6-Face detector achieves 96.9%, 96.0%, 91.6% mAP on the Easy, Medium, and Hard subset, respectively, exceeding the previous SOTA by 0.0%, 0.1%, 0.4%. In the results on the test dataset, our YOLOv5x6-Face detector achieves 95.8%, 94.9%, 90.5% mAP on the Easy, Medium, and Hard subset, respectively with 1.1%, 1.0%, 0.7% gap to the previous SOTA. Please note that, in these evaluations, we only use multiple scales and left-right flipping without using other test-time augmentation (TTA) methods. Our focus is more on the VGA input images, where we achieve the SOTA in almost all conditions.

4.6 YOLO5Face on FDDB Dataset

FDDB dataset [36] is a small dataset with 5171 faces annotated in 2845 images. To demonstrate our YOLO5Face's performance on the cross-domain dataset, we test it on the FDDB dataset without retraining on it. The performances of the true positive rate (TPR) when the number of false-positive is 1000 are listed in Table 5. Please note that it is pointed out in RefineFace [45] that the annotation of FDDB misses many faces. In order to achieve their performance of 0.9911, the RefineFace modifies the FDDB annotation. In our evaluation, we use the original FDDB annotation without modifications. The RetinaFace [7] is not evaluated on the FDDB dataset.

5 Conclusion

In this paper, we present our YOLO5Face based on the YOLOv5 object detector [42]. We implement eight models. Both the largest model YOLOv5l6 and the super small model YOLOv5n achieve close to or exceeding SOTA performance on the WiderFace [40] validation Easy, Medium, and Hard subsets. This proves the effectiveness of our YOLO5Face in not only achieving the best performance but also running fast. Since we open-source the code, a lot of applications and mobile apps have been developed based on our design and achieved impressive performance.

References

1. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: YOLOv4: optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)
2. Cai, Z., Vasconcelos, N.: Cascade R-CNN: delving into high quality object detection. In: CVPR (2018)
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12346, pp. 213–229. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58452-8_13
4. Chen, K., et al.: MMDetection: open MMLab detection toolbox and benchmark. In: ECCV (2020)
5. Chi, C., Zhang, S., Xing, J., Lei, Z., Li, S.Z.: SRN - selective refinement network for high performance face detection. arXiv preprint arXiv:1809.02693 (2018)
6. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: ArcFace: additive angular margin loss for deep face recognition. In: CVPR, June 2019
7. Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I., Zafeiriou, S.: RetinaFace: single-stage dense face localisation in the wild. In: CVPR (2020)
8. Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q.: CenterNet: keypoint triplets for object detection. In: ICCV (2019)
9. Elfwinga, S., Uchibea, E., Doyab, K.: Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. arXiv preprint arXiv:1702.03118 (2017)
10. Feng, Z., Kittler, J., Awais, M., Huber, P., Wu, X.: Wing loss for robust facial landmark localisation with convolutional neural networks. In: CVPR (2018)
11. Girshick, R.: Fast R-CNN. In: Proceedings of the International Conference on Computer Vision (ICCV) (2015)
12. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
13. Guo, J., Deng, J., Lattas, A., Zafeiriou, S.: Sample and computation redistribution for efficient face detection. arXiv preprint arXiv:2105.04714 (2021)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
15. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proceedings of the International Conference on Computer Vision (ICCV) (2017)
16. He, K., Zhang, X., Ren, S., Sun, J.: Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. TPAMI (2015)
17. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks for image classification with convolutional neural networks. In: CVPR (2019)
18. Huang, G., Liu, Z., Maaten, L., Weinberger, K.: Densely connected convolutional networks. In: CVPR (2017)

19. Li, J., et al.: DSFD: dual shot face detector. arXiv preprint arXiv:1810.10220 (2018)
20. Li, Z., Tang, X., Han, J., Liu, J., He, Z.: PyramidBox++: high performance detector for finding tiny face. arXiv preprint arXiv:1904.00386 (2019)
21. Lin, T., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)
22. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. arXiv preprint arXiv:1803.01534 (2018)
23. Liu, W., et al.: YOLOv3: an incremental improvement. In: ECCV (2016)
24. Liu, Y., Wang, F., Sun, B., Li, H.: MogFace: rethinking scale augmentation on the face detector. arXiv preprint arXiv:2103.11139 (2021)
25. Liu, Y., Tang, X., Wu, X., Han, J., Liu, J., Ding, E.: HAMBox: delving into online high-quality anchors mining for detecting outer faces. In: CVPR (2020)
26. Ma, M., Zhang, X., Zheng, H., Sun, J.: ShuffleNet V2: practical guidelines for efficient CNN architecture design. arXiv preprint ArXiv:1807.11164 (2018)
27. Qi, D., Hu, K., Tan, W., Yao, Q., Liu, J.: Balanced masked and standard face recognition. In: ICCV Workshops (2021)
28. Redmon, J., Farhadi, A.: YOLOv3: an incremental improvement. arXiv preprint arXiv:1804.02767 (2015)
29. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: CVPR (2016)
30. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: CVPR (2017)
31. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. (2016)
32. Sandler, M., Howard, A., Zhu, W., Zhmoginov, A., Chen, L.: MobileNetV2: inverted residuals and linear bottlenecks. In: CVPR (2018)
33. Tan, M., Pang, R., Le, Q.: EfficientDet: scalable and efficient object detection. In: CVPR (2020)
34. Tang, X., Du, D.K., He, Z., Liu, J.: PyramidBox: a context-assisted single shot face detector. arXiv preprint ArXiv:1803.07737 (2018)
35. Tian, W., Wang, Z., Shen, H., Deng, W., Chen, B., Zhang, X.: Learning better features for face detection with feature fusion and segmentation supervision. arXiv preprint arXiv:1811.08557 (2018)
36. Jain, V., Learned-Miller, E.: FDDB: a benchmark for face detection in unconstrained settings. University of Massachusetts Report (UM-CS-2010-009) (2010)
37. Wang, R.J., Li, X., Ling, C.X.: Pelee: a real-time object detection system on mobile devices. In: NeurIPS (2018)
38. Yang, M., Kriegman, D., Ahuja, N.: Detecting faces in images: a survey. TPAMI **24**(1), 34–58 (2002)
39. Yang, S., Luo, P., Loy, C.C., Tang, X.: Wider face: a face detection benchmark. shuoyang1213.me/WIDERFACE/index.html
40. Yang, S., Luo, P., Loy, C.C., Tang, X.: Wider face: a face detection benchmark. In: CVPR (2016)
41. Yashunin, D., Baydasov, T., Vlasov, R.: MaskFace: multi-task face and landmark detector. arXiv preprint arXiv:2005.09412 (2020)
42. YOLOv5. github.com/ultralytics/yolov5
43. Zhang, B., et al.: Automatic and scalable face detector. arXiv preprint arXiv:2003.11228 (2020)
44. Zhang, K., Zhang, Z., Li, Z., Qiao, Y.: Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Sig. Process. Lett. **23**(10), 1499–1503 (2016)

45. Zhang, S., Chi, C., Lei, Z., Li, S.: RefineFace: refinement neural network for high performance face detection. arXiv preprint arXiv:1909.04376 (2019)
46. Zhang, S., et al.: ISRN - improved selective refinement network for face detection. arXiv preprint arXiv:1901.06651 (2019)
47. Zhang, S., Zhu, X., Lei, Z., Shi, H., Wang, X., Li, S.Z.: S^3FD: single shot scale-invariant face detector. In: ICCV (2017)
48. Zhang, S., Zhu, X., Lei, Z., Shi, H., Wang, X., Li, S.Z.: FaceBoxes: a CPU real-time face detector with high accuracy. In: IJCB (2017)
49. Zhang, X., Zhou, X., Lin, M., Sun, J.: ShuffleNet: an extremely efficient convolutional neural network for mobile devices. arXiv preprint arXiv:1707.01083 (2017)
50. Zhang, Y., Xu, X., Liu, X.: Robust and high performance face detector. arXiv preprint arXiv:1901.02350 (2019)
51. Zhang, C., Zhang, Z.: Robust real-time face detection. IJCV **57**, 137–154 (2004). https://doi.org/10.1023/B:VISI.0000013087.49260.fb
52. Zhang, C., Zhang, Z.: A survey of recent advances in face detection. Technical report, Microsoft Research (2010)
53. Zhao, R., Liu, T., Xiao, J., Lun, D.P.K., Lam, K.M.: Deep multi-task learning for facial expression recognition and synthesis based on selective feature sharing. In: ICPR (2020)
54. Zhou, X., Wang, D., Philipp, K.: Objects as points. arXiv preprint arXiv:1904.07850 (2019)
55. Zhu, Y., Cai, H., Zhang, S., Wang, C., Xiong, W.: TinaFace: strong but simple baseline for face detection. arXiv preprint arXiv:2011.13183 (2020)
56. Zhu, Z., et al.: WebFace260M: a benchmark unveiling the power of million-scale deep face recognition. In: CVPR (2021)
57. Zhu, Z., et al.: Masked face recognition challenge: the WebFace260M track report. In: ICCV Workshops (2021)