Information Sciences 495 (2019) 136–149

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Proposal pyramid networks for fast face detection

Dan Zeng[a], Han Liu[a], Fan Zhao[a], Shiming Ge[b,*], Wei Shen[a], Zhijiang Zhang[a]

[a] Key laboratory of Specialty Fiber Optics and Optical Access Networks, Joint International Research Laboratory of Specialty Fiber Optics and Advanced Communication, Shanghai Institute of Advanced Communication and Data Science, Shanghai University, Shanghai, 200040, China

[b] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100095, China

ARTICLE INFO

Article history:
Received 2 July 2017
Revised 15 January 2019
Accepted 21 January 2019
Available online 26 April 2019

Keywords:
Face detection
Multi-scale proposals
Cascaded networks
Deep convolutional neural networks

ABSTRACT

Recent face detectors based on deep convolutional neural networks (DCNN) have substantially improved the detection performance in the wild. However, the detection speed is still the biggest bottleneck, which hinders the practical deployment of these face detectors on resource-limited platforms. To address this issue, the paper proposes a Proposal Pyramid Network (PPN) to generate face candidates extremely fast, which reduces the major computational complexity in cascaded DCNN face detectors. PPN is a lightweight fully convolutional network with multiple branches, which takes a single image as input and generates face proposals with different scales in terms of separate branches simultaneously. In this manner, it avoids the traditional image pyramid structure and thus achieves a very fast processing speed. To evaluate the effectiveness of our proposed method, a three-stage cascaded DCNN face detector is realized based on PPN. Extensive experimental results on several public benchmarks show that the proposed face detector achieves comparable accuracy to the-state-of-arts while the detection speed reaches 60 FPS with an i5 CPU, which significantly exceeds previous DCNN-based face detectors. (The source code is available at https://github.com/zhaofan0622/PPN).

1. Introduction

Face detection aims to locate all faces in one image. It is a fundamental step in many face-related vision tasks, such as face registration [24,45], facial image enhancement [48] and face recognition. Since the groundbreaking work of Viola-Jones face detector [36,37], many researchers are dedicated to developing multi-view and variations-robust face detectors by designing better features or classifiers. However, hand-crafted features are not always robust to extreme variations in illumination, posture and occlusion, etc.

In recent years, deep convolutional neural networks (DCNN) have been developed rapidly and brought tremendous progress in many fields including image classification. Compared with hand-crafted features, DCNN generally take raw images as inputs and automatically learn more targeted representations for different tasks. As a result, face detectors based on DCNN have potential to find faces under extreme conditions. Unfortunately, DCNN often contain a lot of parameters and computations which slows down most of these detectors.

To improve the speed of DCNN based face detectors, some methods [14,17,28,46] introduce the cascaded framework and achieve nearly real-time speeds. The cascaded framework is fast and widely used in many commercial face detectors, since

* Corresponding author.
E-mail address: geshiming@iie.ac.cn (S. Ge).

https://doi.org/10.1016/j.ins.2019.01.083
0020-0255/

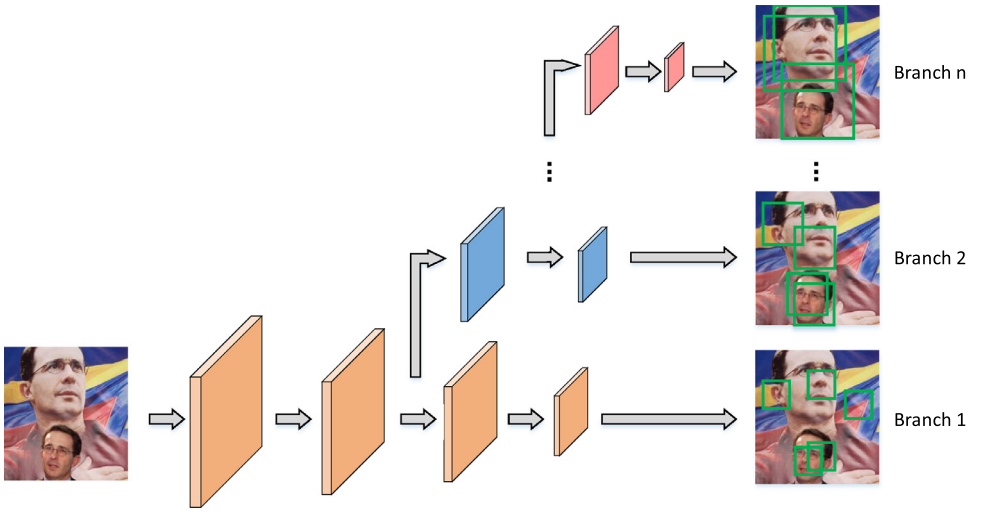

Fig. 1. The network structure of *PPN*. It takes a single image as input and generates multi-scale face proposals simultaneously via multiple branches in a pyramid manner.

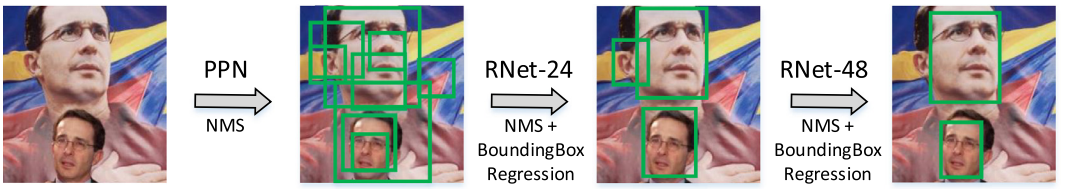

Fig. 2. The pipeline of our three-stage cascaded face detector. The first stage PPN generates multi-scale face proposals with a single input image. Then these candidate windows are refined with the second stage RNet-24 and the third stage RNet-48 successively.

its proposal stage can filter out most background regions with a lightweight model. Meanwhile, an image pyramid is also needed to generate face candidates at multiple scales, which cause that the proposal stage takes most of the detection time. Besides, this proposal strategy has following two drawbacks: (1) Building an image pyramid needs to perform image scaling multiple times, which is time consuming and degrades the image quality. (2) Detecting bigger faces needs to perform a fixed-size detection template on a series of zoom-out images. So bigger faces can't obtain higher recognition accuracies than smaller faces, which goes against our common sense.

To address these issues, this paper proposes a Proposal Pyramid Network (PPN) to fast generate high performance face proposals. PPN is a fully convolutional network (FCN) with total 11 branches, where the odd branches handle detection windows with size $8 \times 2^{(i-1)/2}, i \in [1, 3, 5, 7, 9, 11]$, and the even branches handle detection windows with size $12 \times 2^{i/2-1}, i \in [2, 4, 6, 8, 10]$. Fig. 1 shows how PPN generates face proposals in different scales in terms of different branches simultaneously. Taking a single image with arbitrary size as input, each branch will generate a probability map, in which each element represents the probability that whether a specified size window on the input image contains a face.

The advantages of our PPN are in three folds. First, this network can generate candidate faces at multiple scales with just one input image and one network propagation. Building an image pyramid is no longer necessary. There are total 11 network branches that correspond to 11 different scales. In this way, if the smallest face that can be detected is set to 48×48, the largest face that can be detected will be larger than 1536×1536, which has met most actual requirements. Second, bigger detection windows can achieve higher classification accuracy without extra computations. Compared with most methods that apply the same template on scaled images, our higher-level branch represents a deeper network flow and can naturally find bigger faces well. Third, the architecture of our network is ultra-lightweight. Except the first branch, each of the others just consists of two convolutional layers and one PReLU [7] activation layer. Although the network architecture is simple, its performance is still satisfying, since these network branches equivalently become deeper and deeper.

With PPN, another two networks named RNet-24 and RNet-48 are added to build our three-stage cascaded face detector as shown in Fig. 2. RNet-24 and RNet-48 respectively take 24×24 and 48×48 images as inputs. A bounding box regression module is also added into these two networks to adjust the face position. Although our RNet-24 and RNet-48 have similar structures and functions to the RNet and ONet in MTCNN [46], our models perform much faster by abandoning the use of pooling layers and increasing convolutional strides as much as possible. The performance of our face detector is evaluated on face detection benchmarks FDDB [11] and PASCAL Face [39], which reports a 93.8% accuracy with 500 false positives on FDDB, an average precision (AP) of 93.24% on PASCAL Face and 79.5% on WIDER FACE. As for the detection speed, we

successfully achieve 60 FPS with VGA images in Intel I5 CPU. The accuracy and recall are comparable to state-of-arts while the runtime speed far exceeds them.

Our major contributions are in three folds: (1) a novel pyramid network named PPN is proposed to quickly generate high performance multi-scale face proposals from a single input image. (2) two efficient networks named RNet-24 and RNet-48 are introduced to complete our cascaded detector, which achieves a 60-FPS speed with VGA images in resource-limited i5 CPU while preserving comparable accuracy with state-of-the-arts. (3) a detailed analysis of the role of bounding box regression in a face detector is given in experimental section.

The rest of paper is structured as follows. Section 2 introduces related works. Section 3 describes the proposed method in detail. Section 4 introduces the network design principles and training strategies. Section 5 verifies the effectiveness on public datasets. Section 6 concludes this paper.

2. Related work

Before the rise of deep learning, Viola-Jones face detector [36,37] has promoted the development of face detection. It adopted a cascaded adaboost classifier with haar-like features and converted a classifier to a detector by sliding windows on multi-scale images. Afterwards, a large amount of researchers paid attention to multi-view face detection and variations-robust face detection by developing discriminative features or classifiers [1,18,20,26,34]. DCNN has been rapidly applied in many fields since AlexNet [16] made great success on image recognition task. A series of DCNN based face detection methods have dramatically boosted recall and precision. In this part, we divide these methods into four categories according to the proposal methods being utilized.

2.1. Sliding window based methods

Sliding window is the most direct way to convert a binary classifier to a detector. The basic idea is sliding a detection window across the entire image and evaluating a face model at every location. Many DCNN based face detection methods consider DCNN as a combination of feature extractor and classifier. They detect multi-scale faces by sliding windows on an image pyramid. Deep Dense Face Detector (DDFD) [5] proposed a single model to find faces in various orientations. It is mainly because deep features learned from a large number of samples are more robust to face poses and orientations than traditional hand-crafted features. Convolutional Channel Features (CCF) [40] integrated DCNN features and boosting forest model together. Instead of sliding on each scaled image one by one, they put images with different scales into one large image for feature extraction. Ranjan et al. [29] also built an image pyramid and extracted deep pyramidal features. Deformable part model (DPM) [6] is utilized to perform face detection. DPM defines a face as a collection of facial parts and is robust to occlusion. Also to address the occlusion issue, Opitz et al. [27] proposed a novel loss function named grid loss, which divided the feature map into several independent detectors. To improve the detection efficiency, Bai et al. [2] proposed a fully convolutional network which detects faces in three-scales simultaneously by sliding detection windows on a shared feature map. Then a coarse pyramid image is adopted to find faces in other scales. Cascade network is also widely adopted to increase detection speed, since it can reject most of background regions with small cost in early stages. CascadeCNN [17] trained a cascaded face detector in three stages. The first stage is a mini network to quickly generate face proposals by sliding windows on multi-scale images. They also trained another three networks to regress bounding box positions. Similarly, MTCNN [46] trained three networks including face classification, bounding box regression and facial landmarks location simultaneously. Their multi-task model greatly enhanced the performance of face detection.

2.2. Selective search based methods

Selective Search [35] is used to generate category-independent proposals in object detection. Proposals are detected by computing hierarchical grouping of similar regions based on color, texture, size and shape compatibility. Chu et al. [4] utilized selective search to find candidate face regions in cartoons and refined face regions with DCNN. Faceness [41] trained five models to detect hair, eye, nose, mouth and beard separately. Then a modified selective search was used to generate face candidates based on these five facial parts' response maps. Although Selective Search is designed to be fast maintaining high recall, it is not appropriate for face detection. Because it produces quantities of useless candidate windows and tends to miss partially occluded faces.

2.3. Region proposal network based methods

Region Proposal Network (RPN) is first introduced in Faster R-CNN [30] to generate multi-scale proposals at one time. The classification layer in RPN outputs the confidence scores to indicate the probability of predesigned anchor boxes containing objects. The regression layer in RPN outputs the corresponding bounding box offsets. Faster R-CNN was originally designed to detect multi-class objects. Jiang et.al [13] applied it to face detection task and achieved great performance enhancements. However, Faster R-CNN needs another model to refine proposals from RPN, which causes significant gap between them. To achieve end-to-end optimization, Qin et al. [28] trained RPN and cascaded DCNN jointly to improve the quality of proposals. DeepIR [33] further improved the performance of Faster R-CNN for face detection, serval effective strategies including feature

concatenation, multi-scale training are added to DeepIR. To further enhance the robustness while dealing with extreme situations, Wan et al. [38] improved Faster R-CNN face detection models with a boosting strategy which subsequently finding hard negative samples.

2.4. Dense proposal network based methods

Unlike Faster R-CNN uses an extra network to refine proposals from RPN. DenseBox [10] detects faces of any size with a single input image and a single DCNN model. It utilized a fully convolutional network to produce a dense proposal map, where each pixel predicts a bounding box with a confidence score. To make face localization more accurate, UnitBox [44] replaced the ℓ_2 loss in DenseBox with a novel Intersection over Union (IoU) loss function to regress the bounding box. To improve the robustness of scale variation, SSD [22] found objects of various sizes from multiple feature maps with different resolution. Feature Pyramid Network (FPN) [21] further improved SSD by building a feature pyramid with lateral connections. Although these methods are scale-invariant, they usually fail to find tiny faces. To address this issue, Hu et al. [9] proposed a deep foveal descriptor to capture both context and resolution features, then multi-scale feature templates were performed on multi-scale images to find faces with a wide range of scales.

Although many end-to-end methods [10,13,28,33,38,44] can detect faces of any size on a single input image, they are much slower than some sliding windows and cascade based methods [14,17,46]. Because the former methods always need wider and deeper networks to capture rich features, such as VGGNet [31] and ResNet [8] etc. In contrast, the latter cascade methods can reject most of background regions with few computations and achieve real-time speed.

3. Our method

This section introduces our fast three-stage cascaded face detector. The overall pipeline is shown in Fig. 2. The first stage is the Proposal Pyramid Network (PPN) to generate multi-scale face proposals. The second stage named RNet-24 and the third stage named RNet-48 are both dual-task networks, which are used to refine proposals from PPN and predict offsets of corresponding bounding boxes. In the view of balancing speed and accuracy, we adopt three stages for our detector. More stages lead to higher accuracy while fewer stages make detection faster. Most face detection methods based on cascaded networks have three steps [14,17,46]. For fair comparison, we adapt the same configuration.

3.1. Proposal pyramid networks

PPN is a fully-convolutional network (FCN) with 11 branches, consisting of convolutional layers, PReLU [7] activation layers and Softmax normalization layers. FCN can be trained with fixed-size images and tested with images of arbitrary size. Taking Fig. 3 as an example, the network on the top is the first branch of our PPN. The first convolutional layer has a stride of 2 and the remaining three have the default stride of 1. This network is trained using fixed $8 \times 8 \times 3$ images. During the testing phase, the size of input images and output feature maps are:

$$\begin{aligned} Input &:= M \times N \times 3, \quad (M \geq 8, N \geq 8) \\ Output &:= \left\lfloor \left(\frac{M-8}{2} + 1 \right) \right\rfloor \times \left\lfloor \left(\frac{N-8}{2} + 1 \right) \right\rfloor \times 2 \end{aligned} \tag{1}$$

Each pixel on this output feature map represents the probability of containing a face within an 8×8 detection window on the input image. Actually, the process described above is equivalent to sliding a 8×8 window on the input image with a stride of 2.

Different from methods performing FCN on image pyramid for multi-scale face detection, our PPN with 11 branches generates face proposals with 11 different scales from a single input image. Each branch is equivalent to sliding a detection window of different size on input image. Window size $K(i)$ and sliding stride $S(i)$ of each branch are defined in Eq. (2), where i is the serial number of each branch. Fig. 4 illustrates the specific configurations of our PPN, where $Number \times Height \times Width \times Channel$ indicates convolutional kernels and the red color indicates convolutional layers with a stride of 2. All convolutional layers are followed by PReLU layer except the last layer of each branch. Each branch is ended with a softmax layer to produce normalized probability.

$$\begin{aligned} K(i) &= \begin{cases} 8 \times 2^{(i-1)/2}, & i \in [1, 3, 5, 7, 9, 11] \\ 12 \times 2^{i/2-1}, & i \in [2, 4, 6, 8, 10] \end{cases} \\ S(i) &= \begin{cases} 2^{(i+1)/2}, & i \in [1, 3, 5, 7, 9, 11] \\ 2^{i/2}, & i \in [2, 4, 6, 8, 10] \end{cases} \end{aligned} \tag{2}$$

More branches represent a larger range of face sizes can be detected, however, the number of branches is restrained by training data. If the 12th branch is added to PPN, we need to prepare sufficient 512×512 training images. However, most labeled faces in WIDER FACE [42] are much smaller than 512×512. Besides, we think 11 branches are enough. Suppose the smallest face that can be detected is 48×48, the largest would be 1536×1536 which has met most actual requirements.

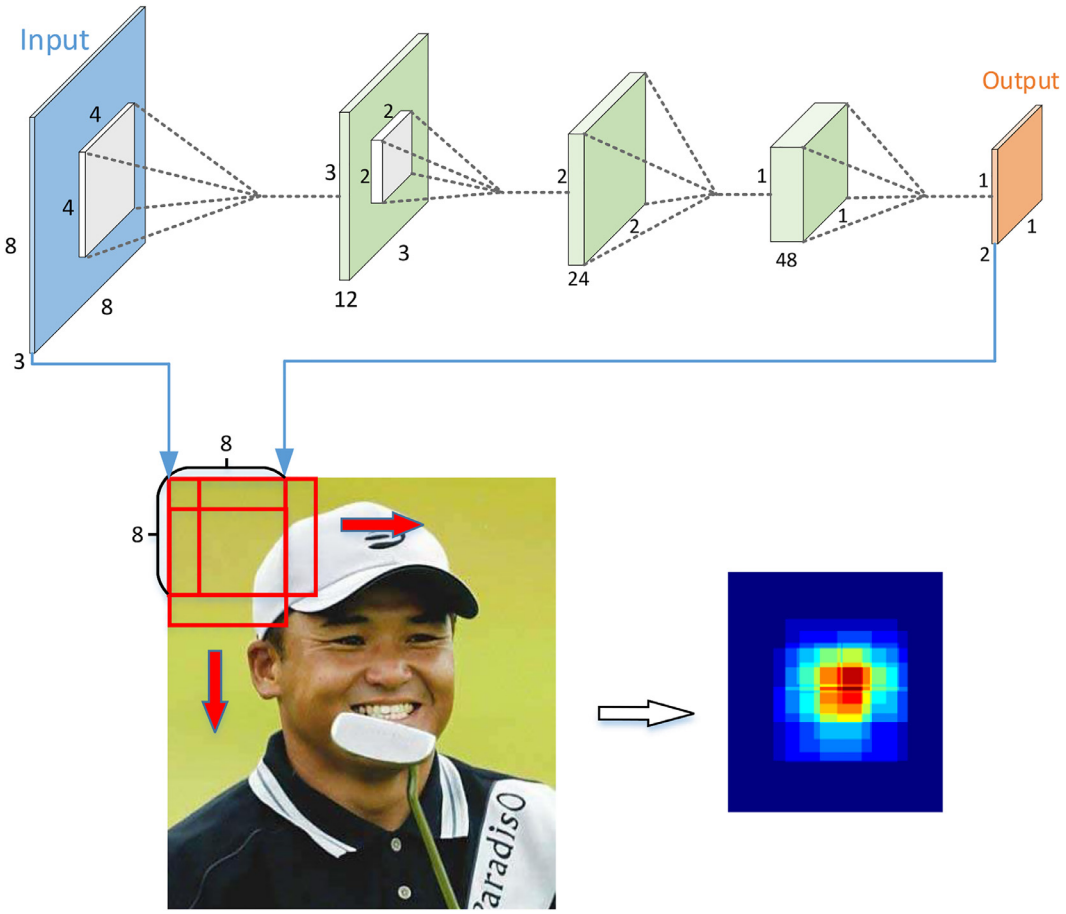

Fig. 3. FCN takes an image of any size as input. The FCN shown above trained with fixed $8 \times 8 \times 3$ images is the first branch of our PPN. The below illustrates that if an input image is larger than 8×8, the process of running this FCN is equivalent to sliding an 8×8 window with a stride of 2.

Table 1
Test classification accuracies of the first six branches in PPN.

	branch1	branch2	branch3	branch4	branch5	branch6
Accuracy	92.4%	94.0%	95.1%	95.2%	94.9%	94.5%

More specifically, given a $6M \times 6N \times 3$ color image, it is scaled 6 times firstly and fed into the pyramid network. Then each branch will generate a probability map with following size:

$$Output := \left\lfloor \left(\frac{M - K(i)}{S(i)} + 1 \right) \right\rfloor \times \left\lfloor \left(\frac{N - K(i)}{S(i)} + 1 \right) \right\rfloor \times 2 \tag{3}$$

Each pixel $p(m, n)$ on ith output feature map corresponds to a square area on input image:

$$x = m * S(i); \quad y = n * S(i); \quad w = K(i); \quad h = K(i); \tag{4}$$

where (x, y) is the top left corner of square area, w or h is the length of square's side.

Fig. 4 shows that higher-level branches take inputs from lower-level branches, which means higher-level branch would have stronger classification ability. The first six branches are trained with the same training images, and the test classification accuracies on the same testing set are shown in Table 1. The classification accuracies firstly increase and then descend with the increase of branch number. There are two main reasons to explain this phenomenon. First, higher-level branches have deeper network architecture for better classification. Second, overfitting is inevitable after training with the same image data for several times. The evidence is that the gap between training accuracy and testing accuracy keeps growing. Yet despite all that, the accuracies of the higher-level branches are all obviously higher than the first branch. This phenomenon also explains why we can get faster and better performance than MTCNN [46] while each of our branches is smaller than PNet in MTCNN.

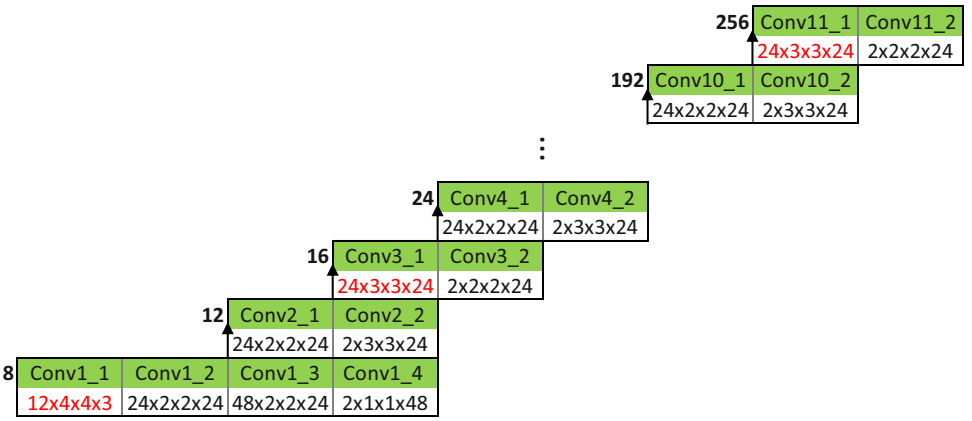

Fig. 4. The specific configuration of PPN. The mathematical expressions indicate the size of convolution kernels. The red color indicates the corresponding convolutional layer has the stride of 2 and the others have a default stride of 1. All convolutional layers are followed by PReLU layer except the last one in each branch. And each branch is ended with a softmax layer. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

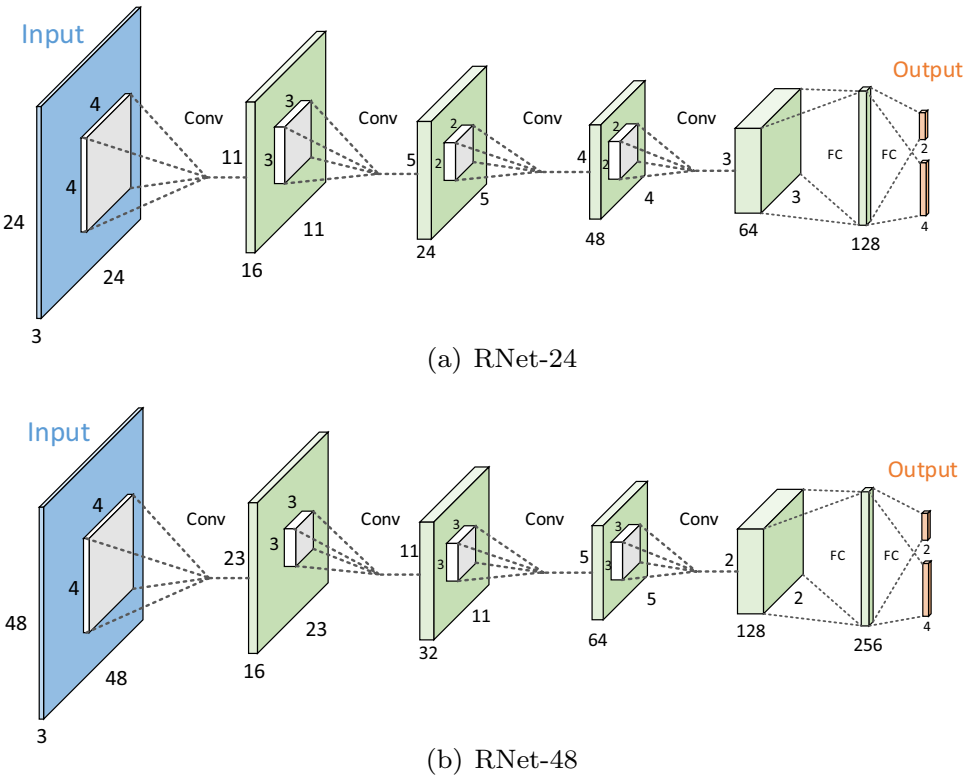

Fig. 5. The architecture of RNet-24 and RNet-48. "Conv" means a convolutional layer, "FC" represents a fully connected layer. All convolutional layers and fully connected layers are followed by PReLU layer except the output layers.

3.2. RNet-24 and RNet-48

The RNet-24 and RNet-48 both are dual-task networks to refine proposals from PPN. Their network architecture are shown in Fig. 5. One of our tasks is binary classification, the other is bounding box regression. We regress relative offsets of bounding boxes instead of absolute coordinates. Offsets are denoted by $[\Delta l, \Delta t, \Delta r, \Delta b]$:

$$\begin{aligned} w^g &= x_r^g - x_l^g; & h^g &= y_b^g - y_t^g \\ \Delta l &= (x_l^p - x_l^g)/w^g; & \Delta t &= (y_t^p - y_t^g)/h^g; \\ \Delta r &= (x_r^p - x_r^g)/w^g; & \Delta b &= (y_b^p - y_b^g)/h^g; \end{aligned} \tag{5}$$

Table 2
Comparison of size and speed with RNet and ONet in MTCNN respectively.

	CaffeModel Size	1000x Forward Propagation
RNet-24	**379 K**	**228 ms**
RNet [46]	399 K	736 ms
RNet-48	**905 K**	**824 ms**
ONet [46]	1522 K	3825 ms

where $[(x_l^p, y_t^p), (x_r^p, y_b^p)]$ denote the top left coordinates and bottom right coordinates of the proposal box respectively, $[(x_l^g, y_t^g), (x_r^g, y_b^g)]$ denote the top left coordinates and bottom right coordinates of the ground truth box respectively.

The final loss function contains two parts, one is the softmax loss for classification and the other is a restricted $L2$ distance loss for regression. A weight factor α is used to balance these two losses. Since classification is the major task, α is set to 0.5 in our experiments.

$$Loss = L_{cls} + \alpha L_{reg} \tag{6}$$

Different from general Euclidean distance loss for regression, a restricted $L2$ loss described in Eq. (7) is adopted. There is no definitive principle to accurately label face bounding boxes, which means the ground truth labels for regression are not completely credible. So, we ignore minor loss and only optimize the loss greater than a threshold σ.

$$\begin{aligned} L_{reg} &= \sum_{i\in[l,t,r,b]} \frac{1}{2} H_i \| \Delta i - y_i \|^2 \\ H_i &= \begin{cases} 0 & if \ \| \Delta i - y_i \|^2 < \sigma \\ 1 & elsewise \end{cases} . \end{aligned} \tag{7}$$

where y_i, $i \in [l, t, r, b]$ is the network prediction of bounding box offset.

4. Network designing and training

4.1. Network designing

The network designing aims to improve speed while maintaining high accuracy. For the first stage PPN, all branches are designed to be lightweight. In particular, each of the latter 10 branches in PPN has only two convolutional layers and one PReLU layer. Even so, the overall performance is guaranteed. Because higher-level branches with deeper architecture have stronger classification ability.

In order to accelerate the inference speed of RNet-24 and RNet-48 as much as possible, we increase strides of convolutional layers and abandon using pooling layers (Max pooling or Average pooling) to realize downsampling operations inspired by Springenberg et al. [32]. We use im2col [3] to simplify complex convolution to matrix-matrix multiplication. Even so, a convolutional layer still costs more time than a fully connected layer with the same amount of parameters. There are two main reasons. Firstly, im2col takes extra time to rearrange image blocks into columns and merge these column vectors into one matrix. Secondly, the large matrix generated from im2col further increases the computational complexity of matrix multiplication. Increasing convolutional stride can reduce both stacking frequency in im2col and computational complexity in matrix multiplication. To verify the acceleration effect, we compare our models with the RNet and ONet in MTCNN [46], which is also a three-stage cascaded face detector, whose RNet and ONet are equivalent to our RNet-24 and RNet-48 respectively. Results are shown in Table 2.

4.2. Training dataset

The public datasets used to train our three-stage cascade face detector are WIDER FACE [42] and AFLW [15]. WIDER FACE contains 32,203 images with 393,703 labeled faces, most of these faces have extreme variations in scale, pose, illumination, and occlusion, as shown in Fig. 6. However, since the majority of labeled faces are smaller than 48×48, WIDER FACE fails to support a complete training of PPN. Hence AFLW is introduced to supplement positive samples during training high-level branches. AFLW consists of 25,993 labeled faces in 21,997 images. More than 80 percent of labeled faces in AFLW are larger than 128×128. Note that labeled bounding boxes in WIDER FACE are rectangles but squares in AFLW as shown in Fig. 6. So we only use WIDER FACE to train RNet-24 and RNet-48 which both have the bounding box regression task.

4.3. Training strategies

Caffe [12] is used to train our three-stage network. The weights initialization method is Xavier and the network is optimized by SGD with a momentum of 0.9. The batch size for all training stages is 256 and the ratio of positive samples to negative samples is 1:3.

Fig. 6. Some face examples in WIDER FACE [42] and AFLW [15] datasets.

Table 3
There are three sizes of training images for PPN. "source size" means the minimum size of valid training samples. "target size" means the size that all training images will be firstly resized to.

branch ID	target size	source size	dataset
branch1-branch6	48	≥ 32	WIDER FACE
branch7-branch9	128	≥ 64	WIDER FACE + AFLW
branch10-branch11	256	≥ 128	WIDER FACE + AFLW

PPN: PPN has 11 branches which are trained one by one. Higher-level branches need larger training images. But most labeled faces in WIDER FACE are too small. To increase the amount of valid positive samples as much as possible, three sizes of training images are cropped and resized from WIDER FACE and AFLW as shown in Table 3. Take the first row in Table 3 as an example to make clear how we prepare training data. Image patches with IoU overlapping with ground truth bounding boxes below 0.25 are assigned to negative samples. Image patches larger than 32×32 with IoU overlapping with ground truth bounding boxes over 0.5 are assigned to positive samples. The rest will be ignored. All valid image patches are cropped and resized to 48×48 for training branches 1 to 6. These images are further resized to 8×8 without antialiasing to train the branch 1 with a dynamic learning rate:

$$l = l_b \left(1 - \frac{i}{i_{max}}\right)^p. \tag{8}$$

where l_b is the initial learning rate, i is the number of iterations, i_{max} is the maximum number of iterations, and p is a power factor. In our experiments, $[l_b, i_{max}, p]$ are set to $[0.001, 6.4 \times 10^5, 2]$. Meanwhile we adopt early stopping to choose final models.

The shared layers are fixed after training branch 1. Then images are resized to 12×12 to train branch 2. The rest branches are trained sequentially in the same way.

RNet-24: Firstly, PPN is used to detect faces on WIDER FACE [42]. Then all detected faces are divided into positive set or negative set based on the IoU overlapping with ground truth bounding boxes. Besides, bounding box offsets of positives samples are computed as Eq. (5). Since the initial loss of bounding box regression is much larger than the initial loss of classification, the previous strategy to adjust learning rate is improper. In order to achieve faster convergence, we pre-train this network with a novel strategy. The learning rate is initially set to 10^{-8} and magnified 10 times every 1K iterations until 5K iterations. This pre-trained model is retrained with the strategy used in PPN training.

RNet-48: Training strategies of RNet-48 is similar to RNet-24. The only difference is that samples passed through PPN and RNet-24 need to be corrected with the predicted bounding box offsets.

5. Experiments

In this section, we first evaluate our proposed PPN face detector on three challenging face detection benchmarks (FDDB [11], PASCAL Face [39] and WIDEF FACE [42]) to verify its performance. Then, we conduct an ablation study. Finally, we show the runtime effectiveness and conclude some interesting observations.

5.1. Evaluation on FDDB

FDDB [11] is a popular face detection benchmark. 5171 faces are labelled in 2845 images with complicated variations of scale, pose and occlusion. The authors also provide a standard evaluation tool, which considers a detection to be positive if the ratio of the intersection of detected face region with annotated face region is greater than 0.5. Besides, Receiver Operating Characteristic (ROC) curve is shown in Fig. 7 by plotting the number of false positives against the true positive

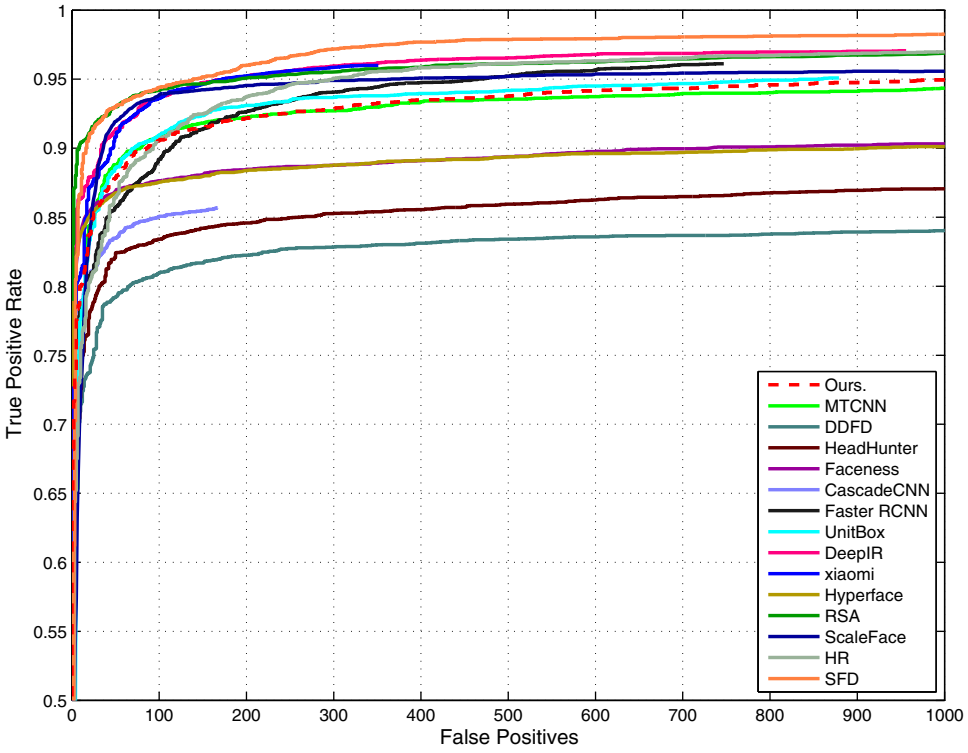

Fig. 7. ROC curves of our face detector on FDDB.

rate (TPR) at various threshold settings. It is obvious that our detector achieves comparable accuracy with state-of-the-arts. Though some methods achieve better accuracies, they all use huge models and are extremely slow.

Fig. 8 shows some detection examples on FDDB. It demonstrates that our face detector is robust to large occlusion and severe illumination, etc. Face rectangle positions become more accurate with bounding box regression.

5.2. Evaluation on PASCAL face

We further conduct experiments on PASCAL Face [39] benchmark which consists of 1341 faces in 851 images. As shown in Fig. 9, our face detector achieves an average precision of 93.24%, which is obviously better than Faceness [41], MTCNN [46] and HeadHunter [25], etc.

5.3. Evaluation on WIDER FACE

To evaluate the capacity of our PPN on the practical deployment in the wild, we further conduct experiments on WIDER FACE [42]. We compare our PPN with five fast face detection models based on multi-stage and/or multiscale scheme, including two baseline models (two-stage CNN [42] and multi-scale cascaded CNN [42]), a two-stage model (Faceness [41]), a three-stage model (MTCNN [46]) and a one-stage multiscale model (ScaleFace [43]). The result is shown in Fig. 10, which demonstrates several observations. First, our PPN achieves the fastest detection speed of 60 FPS beyond its light-weight size, showing that it could facilitate practical deployment (e.g., combining with face recognition model to identify faces in the wild). Second, PPN gives a lower precision than ScaleFace and MTCNN since only faces larger than a certain size (e.g., 48×48) in WIDER FACE are used to train RNet-24 and RNet-48, implying that the first two stages could be further improved by training with more satisfied data.

On WIDER FACE dataset, we also check the impact of various parameter settings on the detection precision. The first parameter is the smallest face size, which is used to control the range of faces that could be detected. In our experiment, we achieve the average precision of 0.781, 0.783, 0.785 and 0.774 when the smallest face size is set to 20, 30, 35 and 40, respectively. It implies that the smallest face size is important for a specified scenario and should be modified accordingly. For example, the best average precision is achieved when the smallest face size is 35 for WIDER FACE, since many face images in the hard subset are captured from crowd scenarios and have relative small face sizes. The thresholds in three stages can also affect average precision, as shown in Fig. 11. The proposal network in the first stage is used to improve the recall while balancing the false detections, thus smaller score factor will let more face candidates pass through. When the score factor increases to a large value, the recall is very low so that no faces can be detected, as shown in the left of

Fig. 8. Detection examples on the FDDB with our face detector.

Fig. 11. In RNet-24 and RNet-48, the thresholds are used to accept more confident faces. A small threshold causes high false positive rate, while a large one decreases false negatives, thus an appropriate threshold is crucial. From the middle and right of Fig. 11, we can find the average precision is stable when the threshold is in a certain range, implying the robustness of our proposed PPN model.

5.4. Evaluation on runtime speed

DCNN based face detectors have greatly improved the recall and accuracy, but most of them are time-consuming. For many practical applications, speed is the priority and low-quality faces are useless. We compare the speed of our detector with many other state-of-the-arts methods. Table 4 shows that our runtime speed significantly exceeds the others. With a single CPU core and a single thread, we can process 60 VGA images per second when the smallest face that can be detected is set to 48×48.

Compared with MTCNN [46] and CascadeCNN [17] which also use cascaded detection framework, our performance and speed are both superior to them. As for xiaomi [38], DeepIR [33], RSA [23], etc, although their accuracies are slightly higher than ours, their detection speeds are much slower. VGGNet [31] and ResNet [8] in their networks both have a huge number of parameters and need nearly 200 ms for one forward propagation on CPU. So, these face detectors using VGGNet or ResNet are too slow for real-time applications.

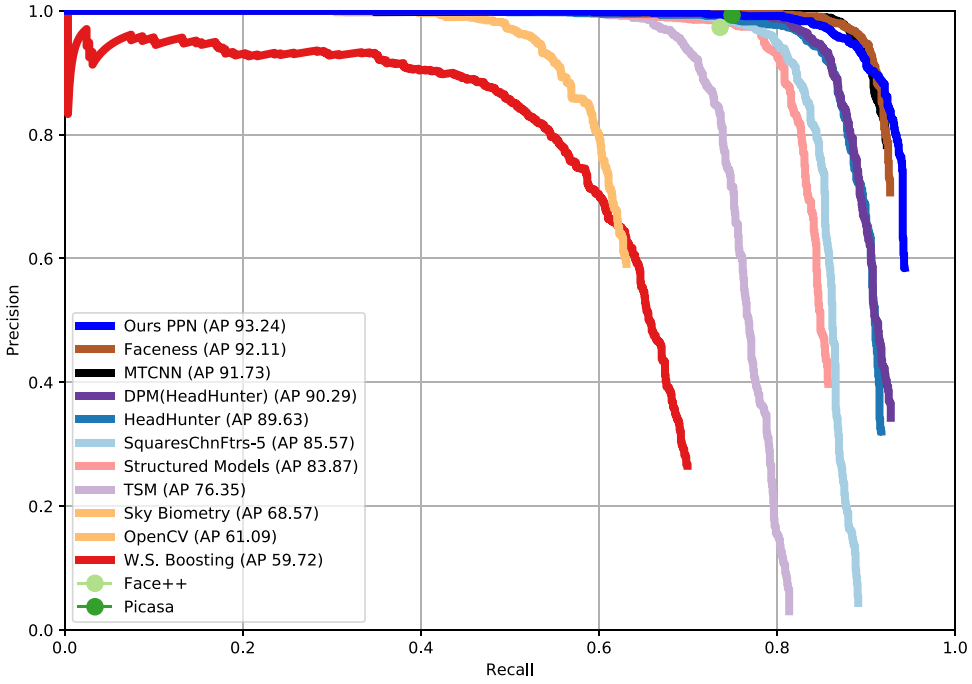

Fig. 9. Comparison of the precision-recall curves on PASCAL Face.

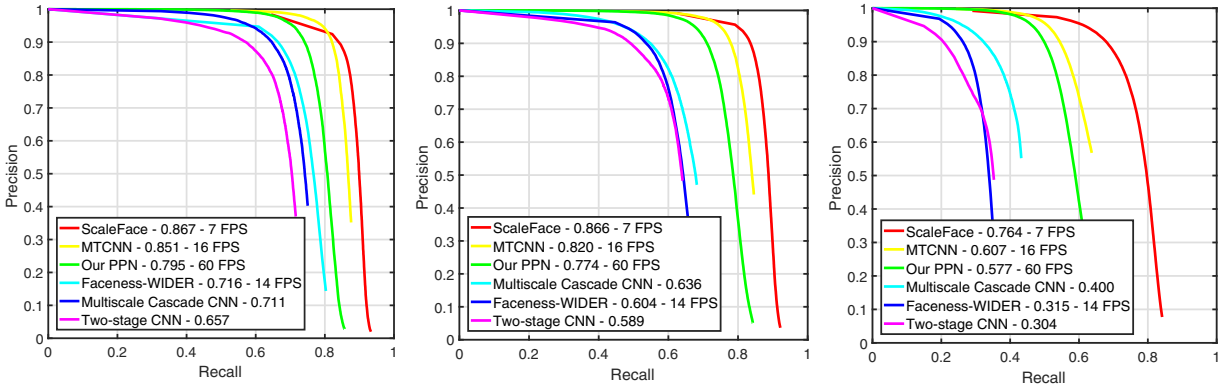

Fig. 10. The performance on WIDER FACE. Precisions are given when testing on easy subset (left), medium subset (middle) and hard subset (right), respectively.

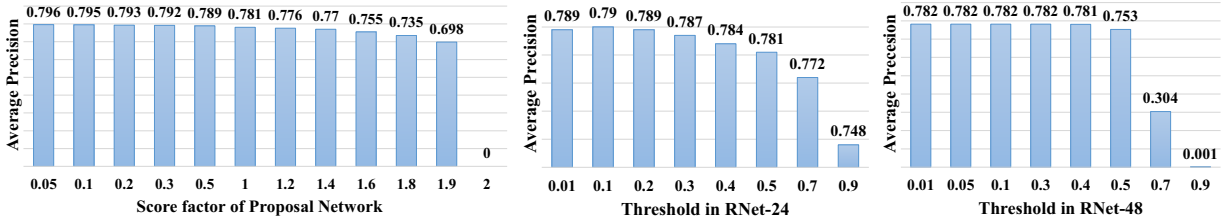

Fig. 11. Effect of parameter settings in three stages.

Table 4
Runtime comparisons with state-of-the-art methods.

Methods	Speed	Configuration	Method category
Faceness [41]	20 FPS	TITAN GPU	Modified Selective Search
UnitBox [44]	< 6 FPS	CPU (VGG [31])	
HR [9]	3.1 FPS	GPU (ResNet [8])	Dense Proposal
SFD [47]	36 FPS	TITAN X GPU	Network
Qin et al. [28]	10 FPS	CPU	
Faster R-CNN [13]	< 6 FPS	CPU (VGG [31])	
DeepIR [33]	< 6 FPS	CPU (VGG [31])	Region Proposal
xiaomi [38]	< 8 FPS	CPU (ResNet [8])	Network
ScaleFace [43]	< 8 FPS	GPU (ResNet [8])	
RSA [23]	< 8 FPS	GPU (ResNet [8])	
DP2MFD [29]	0.285 FPS	Tesla K20 GPU	
Conv3D [19]	< 6 FPS	CPU (VGG [31])	
MTCNN [46]	15 FPS	CPU	Sliding Window
CascadeCNN [17]	14 FPS	CPU	
Ours	**60 FPS**	**I5 CPU**	
	173 FPS	**GTX 1080 Ti**	

Table 5
RNet-24 training and testing results with or without bounding box regression.

With bounding box regression?	Train loss	Train acc	Test loss	Test acc
No	0.1045	97.3%	0.1497	94.5%
Yes	0.1126	96.4%	0.1357	94.9%

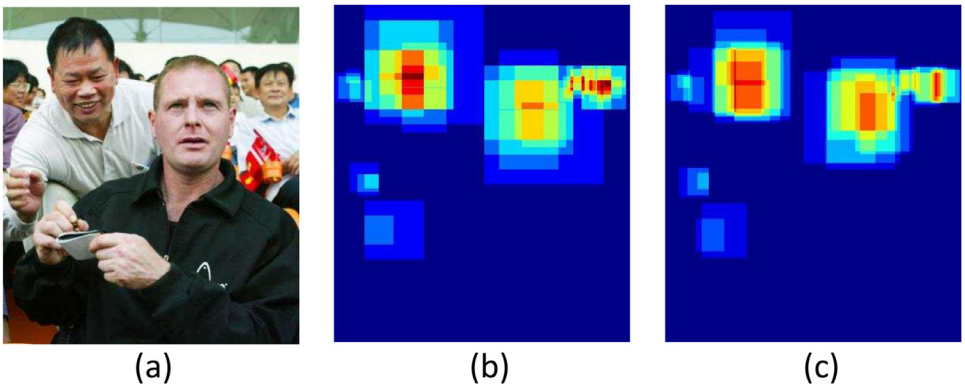

Fig. 12. Bounding box regression makes NMS more efficient. (a) Is an input image, (b) illustrates the output of RNet-24 without bounding box correction, and (c) shows the output of RNet-24 with bounding box correction.

5.5. Bounding box regression

RNet-24 and RNet-48 are both dual-task networks, dealing with bounding box regression and binary classification at the same time. We think bounding box regression produces three key benefits:

1) Bounding box regression can be regarded as a regularizator to the classification task. Based on our experimental results, the binary classification is easier to converge but more likely to be overfitting than the regression. Bounding box regression can effectively avoid overfitting by adding some gradient "noise". To verify it, we train another network with the same architecture without regression task. Results are shown in Table 5. Obviously, the gap between training and testing narrows with bounding box regression, which means the generalization ability gets enhanced.
2) Bounding box regression improves the classification accuracy of positive samples. After bounding box correction, the face position becomes more accurate and the detection window will have a higher probability to be classified correctly.
3) Bounding box regression makes non-maximum suppression (NMS) more efficient. NMS is used to merge overlapping bounding boxes, which continuously selects the bounding box with maximum detection score and suppresses its neighbor bounding boxes with a predesigned IoU threshold. As shown in Fig. 12, bounding box regression makes candidate boxes more concentrated and suppresses useless boxes after NMS. We further prove it by counting the number of boxes after NMS with or without bounding box regression. The results in Table 6 show that NMS reduces more candidate boxes with bounding box correction, which also means the computation of RNet-48 is reduced.

Table 6
The number of bounding boxes after NMS with or without bounding box regression on FDDB [11].

	With bounding box regression	Without bounding box regression
RNet-24	25762	35695
RNet-48	6139	6481

6. Conclusion

This paper presented a novel convolutional neural network called PPN to generate face proposals. Compared with sliding detection windows on pyramid images, PPN generates multi-scale face proposals with a single input image, which is faster and more accurate. By cascading PPN with another two dual-task networks, our three-stage cascaded face detector achieves comparable performance on FDDB, PASCAL Face and WIDER FACE benchmarks. Particularly, our detection speed reaches 60 FPS with VGA images in i5 CPU, significantly exceeds previous methods. Our future work will explore how to further reduce the amount of proposals with satisfied recall and accelerate the networks without performance degradation.

Conflict of interest

None.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61572307 & 61402463) and the National Key Research and Development Plan (2016YFC0801005). Shiming Ge is also supported by Youth Innovation Promotion Association, CAS.

References

[1] M.S. Al-Ani, A.S. Al-Waisy, Multi-view face detection based on kernel principal component analysis and kernel support vector techniques, Int. J. Softw. Comp. 2 (2) (2011) 1–13.
[2] Y. Bai, W. Ma, Y. Li, L. Cao, W. Guo, L. Yang, Multi-scale fully convolutional network for fast face detection., BMVC, 2016.
[3] K. Chellapilla, S. Puri, P. Simard, High performance convolutional neural networks for document processing, Tenth International Workshop on Frontiers in Handwriting Recognition, Suvisoft, 2006.
[4] W.-T. Chu, W.-W. Li, Manga facenet: face detection in Manga based on deep neural network, in: Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, ACM, 2017, pp. 412–415.
[5] S.S. Farfade, M.J. Saberian, L.-J. Li, Multi-view face detection using deep convolutional neural networks, in: Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, ACM, 2015, pp. 643–650.
[6] P. Felzenszwalb, D. McAllester, D. Ramanan, A discriminatively trained, multiscale, deformable part model, in: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–8.
[7] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.
[8] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[9] P. Hu, D. Ramanan, Finding tiny faces, arXiv preprint arXiv:1612.04402 (2016).
[10] L. Huang, Y. Yang, Y. Deng, Y. Yu, Densebox: unifying landmark localization with end to end object detection, arXiv preprint arXiv:1509.04874 (2015).
[11] V. Jain, E.G. Learned-Miller, Fddb: A Benchmark for Face Detection in Unconstrained Settings, UMass Amherst Technical Report, 2010.
[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, in: Proceedings of the 22nd ACM International Conference on Multimedia, ACM, 2014, pp. 675–678.
[13] H. Jiang, E. Learned-Miller, Face detection with the faster r-cnn, arXiv preprint arXiv:1606.03473 (2016).
[14] I. Kalinovskii, V. Spitsyn, Compact convolutional neural network cascade for face detection, arXiv preprint arXiv:1508.01292 (2015).
[15] M. Köstinger, P. Wohlhart, P.M. Roth, H. Bischof, Annotated facial landmarks in the wild: alarge-scale, real-world database for facial landmark localization, in: Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, IEEE, 2011, pp. 2144–2151.
[16] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[17] H. Li, Z. Lin, X. Shen, J. Brandt, G. Hua, A convolutional neural network cascade for face detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5325–5334.
[18] J. Li, Y. Zhang, Learning surf cascade for fast and accurate object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 3468–3475.
[19] Y. Li, B. Sun, T. Wu, Y. Wang, Face detection with end-to-end integration of a convnet and a 3d model, in: European Conference on Computer Vision, Springer, 2016, pp. 420–436.
[20] S. Liao, A.K. Jain, S.Z. Li, A fast and accurate unconstrained face detector, IEEE Trans. Pattern Anal. Mach. Intell. 38 (2) (2016) 211–223.
[21] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, arXiv preprint arXiv:1612.03144 (2016).
[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, Ssd: single shot multibox detector, in: European Conference on Computer Vision, Springer, 2016, pp. 21–37.
[23] Y. Liu, H. Li, J. Yan, F. Wei, X. Wang, X. Tang, Recurrent scale approximation for object detection in CNN, in: IEEE International Conference on Computer Vision, 2017.
[24] J. Ma, J. Zhao, Y. Ma, J. Tian, Non-rigid visible and infrared face registration via regularized gaussian fields criterion, Pattern Recognit. 48 (3) (2015) 772–784.
[25] M. Mathias, R. Benenson, M. Pedersoli, L. Van Gool, Face detection without bells and whistles, in: European Conference on Computer Vision, Springer, 2014, pp. 720–735.

[26] E. Ohn-Bar, M.M. Trivedi, To boost or not to boost? On the limits of boosted trees for object detection, arXiv preprint arXiv:1701.01692 (2017).
[27] M. Opitz, G. Waltner, G. Poier, H. Possegger, H. Bischof, Grid loss: detecting occluded faces, in: European Conference on Computer Vision, Springer, 2016, pp. 386–402.
[28] H. Qin, J. Yan, X. Li, X. Hu, Joint training of cascaded CNN for face detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3456–3465.
[29] R. Ranjan, V.M. Patel, R. Chellappa, A deep pyramid deformable part model for face detection, in: Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on, IEEE, 2015, pp. 1–8.
[30] S. Ren, K. He, R. Girshick, J. Sun, Faster r-CNN: towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems, 2015, pp. 91–99.
[31] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).
[32] J.T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: the all convolutional net, arXiv preprint arXiv:1412.6806 (2014).
[33] X. Sun, P. Wu, S.C. Hoi, Face detection using deep learning: an improved faster rcnn approach, arXiv preprint arXiv:1701.08289 (2017).
[34] J. Trefnỳ, J. Matas, Extended set of local binary patterns for rapid object detection, in: Computer Vision Winter Workshop, 2010, pp. 1–7.
[35] J.R. Uijlings, K.E. Van De Sande, T. Gevers, A.W. Smeulders, Selective search for object recognition, Int.J.Comput.Vis. 104 (2) (2013) 154–171.
[36] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1, IEEE, 2001. I–I
[37] P. Viola, M.J. Jones, Robust real-time face detection, Int.J.Comput.Vis. 57 (2) (2004) 137–154.
[38] S. Wan, Z. Chen, T. Zhang, B. Zhang, K.-k. Wong, Bootstrapping face detection with hard negative examples, arXiv preprint arXiv:1608.02236 (2016).
[39] J. Yan, X. Zhang, Z. Lei, S.Z. Li, Face detection by structural models, Image Vis. Comput. 32 (10) (2014) 790–799.
[40] B. Yang, J. Yan, Z. Lei, S.Z. Li, Convolutional channel features, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 82–90.
[41] S. Yang, P. Luo, C.-C. Loy, X. Tang, From facial parts responses to face detection: a deep learning approach, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3676–3684.
[42] S. Yang, P. Luo, C.-C. Loy, X. Tang, Wider face: a face detection benchmark, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5525–5533.
[43] S. Yang, Y. Xiong, C.C. Loy, X. Tang, Face detection through scale-friendly deep convolutional networks, arXiv preprint arXiv:1706.02863 (2017).
[44] J. Yu, Y. Jiang, Z. Wang, Z. Cao, T. Huang, Unitbox: an advanced object detection network, in: Proceedings of the 2016 ACM on Multimedia Conference, ACM, 2016, pp. 516–520.
[45] D. Zeng, F. Zhao, Y. Bao, Compressing deep neural network for facial landmarks detection, in: Advances in Brain Inspired Cognitive Systems: 8th International Conference, BICS 2016, Beijing, China, November 28-30, 2016, Proceedings 8, Springer, 2016, pp. 102–112.
[46] K. Zhang, Z. Zhang, Z. Li, Y. Qiao, Joint face detection and alignment using multitask cascaded convolutional networks, IEEE Signal Process. Lett. 23 (10) (2016) 1499–1503.
[47] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, S.Z. Li, Sfd: single shot scale-invariant face detector, arXiv preprint arXiv:1708.05237 (2017).
[48] Z. Zhang, Y. Wang, Z. Zhang, Face synthesis from near-infrared to visual light via sparse representation, in: Biometrics (IJCB), 2011 International Joint Conference on, IEEE, 2011, pp. 1–6.