Multi-Scale Fully Convolutional Network for Fast Face Detection

Yancheng Bai[1]
yancheng@iscas.ac.cn
Wenjing Ma[1]
wenjing@iscas.ac.cn
Yucheng Li[1]
yucheng@iscas.ac.cn
Liangliang Cao[2]
liangliang.cao@gmail.com
Wen Guo[3]
grewen@126.com
Luwei Yang[4]
luweiy@sfu.ca

[1] Institute of Software, Chinese Academy of Science, Beijing, China
[2] Columbia University and Yahoo Labs New York, USA
[3] Shandong Technology and Business University, Shandong, China
[4] Simon Fraser University Vancouver, Canada

Abstract

Image pyramid is a common strategy in detecting objects with different scales in an image. The computation of features at every scale of a finely-sampled image pyramid is the computational bottleneck of many modern face detectors. To deal with this problem, we propose a multi-scale fully convolutional network framework for face detection. In our detector, face models at different scales are trained end-to-end and they share the same convolutional feature maps. During testing, only images at octave-spaced scale intervals need to be processed by our detector. And faces of different scales between two consecutive octaves can be detected by multi-scale models in our system. This makes our detector very efficient and can run about 100 FPS on a GPU for VGA images. Meanwhile, our detector shows superior performance over most of state-of-the-art ones on three challenging benchmarks, including FDDB, AFW, and PASCAL faces.

1 Introduction

Face detection is an active research topic in computer vision and has many applications including facial expression recognition, face recognition, face parsing and human computer interface (HCI), just to name a few. During the past degrade, great successes have been made due to the availability of large amount of training data in unconstrained conditions and the development of robust computer vision algorithms, e.g., boosting-based methods, deformable part based models (DPM) and convolutional neural networks (CNN). A thorough review can be found in the survey [30].

Generally speaking, face detection is usually addressed by sliding window based methods. Most detection systems only train one scale (single-view or multi-view) model. To

Pages 51.1-51.12
DOI: https://dx.doi.org/10.5244/C.30.51

detect faces at different scales in an image, an image pyramid needs to build via repeated smoothing and sub-sampling the original image, which is shown in Fig. 1 (A). And then the corresponding feature (e.g. HOG, LBP) pyramid is built by computing a specific feature from each level of the image pyramid. The scale sampling in an image pyramid is determined by a parameter K that defines the number of levels in an octave. That is, K is the number of levels that is required to go down in the pyramid to get to a feature map computed at twice the resolution of another one. In Fig. 1 (A), an image with red border is an octave and an image with baby blue border is the finely-sampled one. In practice, K is usually set as $3-10$. With a typical setting $K=7$, there are about 40 levels of a VGA (640×480) image pyramid. Therefore, it is a burden to compute features at every level of an image pyramid, which is the bottleneck of many modern face detectors. To address this problem, Dollár *et al*. [3] argue that features on a finely-sampled pyramid can be approximated by features computed at octave-spaced scale intervals, rather than being computed explicitly. With such an approximation, the detector [28] can run at real-time frame rate on typical VGA size images, but with slight loss in the detection accuracy.

Considering the efficiency, we propose a multi-scale fully convolutional network (MS-FCN) framework for face detection. In our MS-FCN model, K face models with different scales are trained end-to-end. More importantly, these models share the same full-image convolutional features. During testing, only images at octave-spaced scale intervals (images with red border in Fig. 1 (A)) need to be processed by our detector. And faces of different scales between two consecutive octaves can be dealt with K face models at different scales in our system. The above strategies make our detector very efficient, which can run about 100 FPS on a GPU (Nvidia GTX 980) for VGA images. Meanwhile, our detector can achieve state-of-the-art detection performance on three public face detection benchmarks, including FDDB, AFW, and PASCAL faces.

The remainder of this paper is organized as follows. In Sec. 2, we briefly review the related work. In Sec. 3, we give a detailed description of our MS-FCN face detector, including the network architecture, training strategies and parameter settings. Experimental results and comparison with other state-of-the-art approaches are presented in Sec. 4. And we conclude our work in Sec. 5.

2 Related work

During the past decade, many face detection systems have been proposed, such as [19] [24]. Among these methods, the boosting cascade framework [24] proposed by Viola and Jones (VJ) is a milestone work in face detection. Thanks to simple Haar features and the integral image trick, the computation cost of feature pyramids is relatively small. And with the attentional cascade structure, the VJ framework can reject more negative sample at early stages. Therefore, the VJ framework shows its superior advantage in speed, and becomes the most popular method in face detection. However, the simple Haar features have limited representation, which leads to poor performances of VJ detectors in uncontrolled environments, due to the large appearance variations caused by the unconstrained illumination, sever occlusion, highly exaggerated expressions and so on. To enrich the capacity of feature representation, HOG [2], SURF [15] and other complicated features are exploited, which improve the detection accuracy. However, most above detection systems only train one scale model and feature pyramids have to be built, which increases the computational costs drastically, especially when complicated features are used.

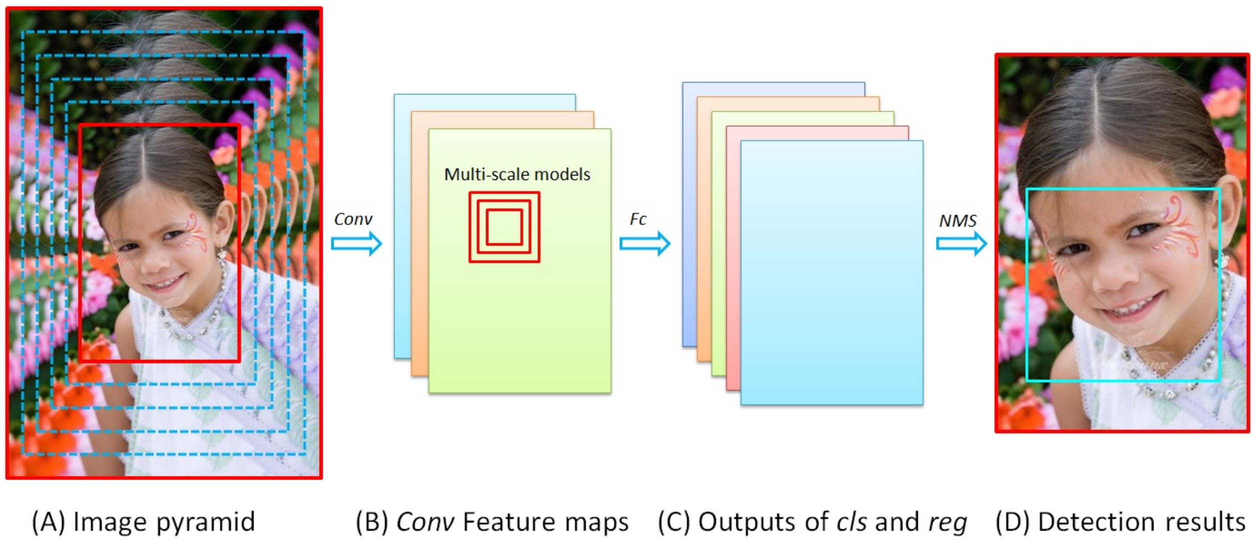


Figure 1: The pipeline of the proposed MS-FCN detector system. (A) Only images at octave-spaced scale intervals in the image pyramid (images with red border) are fed to the network; (B) After several layers of convolution, output the shared convolutional (*conv*) feature maps; (C) Multi-scale fully connected layers slide on the *conv* feature maps and output classification (*cls*) and regression (*reg*) results; (D) Convert *cls* and *reg* outputs to bounding boxes, and apply non-maximum suppression (NMS) to all bounding boxes over the threshold and get the final detection results.

DPM-based methods [25] [17] [26] are another stream for face detection. DPM models learn root filters, part filters and their spatial relationships via latent support vector machine (SVM). Therefore, DPM models are very robust to occlusion. [25] [17] [26] demonstrate state-of-the-art performance of DPM models. However, building feature pyramids makes DPM-based methods computationally intensive.

There is a long history of deploying neural network for the task of face detection. As early as in 1994, Vaillant *et al.* [23] propose a face detection algorithm based on neural network. And in 1998, Rowley *et al.* [19] present a retinally connected neural network-based face detection system to detect upright frontal face in image pyramid. And they fuse multiple networks to improve performance. Obviously, it is hard to know the performance of these ancient detectors on today's face detection benchmarks. However, they are still worth reviewing, as there are many similarities in design with modern CNN-based face detection systems.

Recently, with the break-through results of CNNs for image classification [13] and object detection [7] [6] [18], deep CNN-based face detectors [4] [14] [29] [8] have been proposed. Inspired by the boosting-based algorithms, Li *et al.* [14] propose a cascaded architecture called CascadeCNN for real-world face detection. Two lower-resolution models are used to quickly reject most false detection windows and higher resolution models are applied to verify the detections carefully. Although multi-resolution models are used in CascadeCNN, the insight is totaly different. Our MS-FCN models use multi-scale models to deal with different sizes of faces while CascadeCNN intends to reject majority of non-face windows in images. What is more, every stage of CascadeCNN needs to be designed carefully and is trained separately, while our MS-FCN models are simpler and can be trained end-to-end. Yang *et al.* [29] demonstrate that facial attribute CNN models can be applied to find face proposals and the proposed windows can be further processed by an AlexNet-like CNN model. The Faceness-Net [29] show slightly lower performance and less efficiency than our MS-FCN

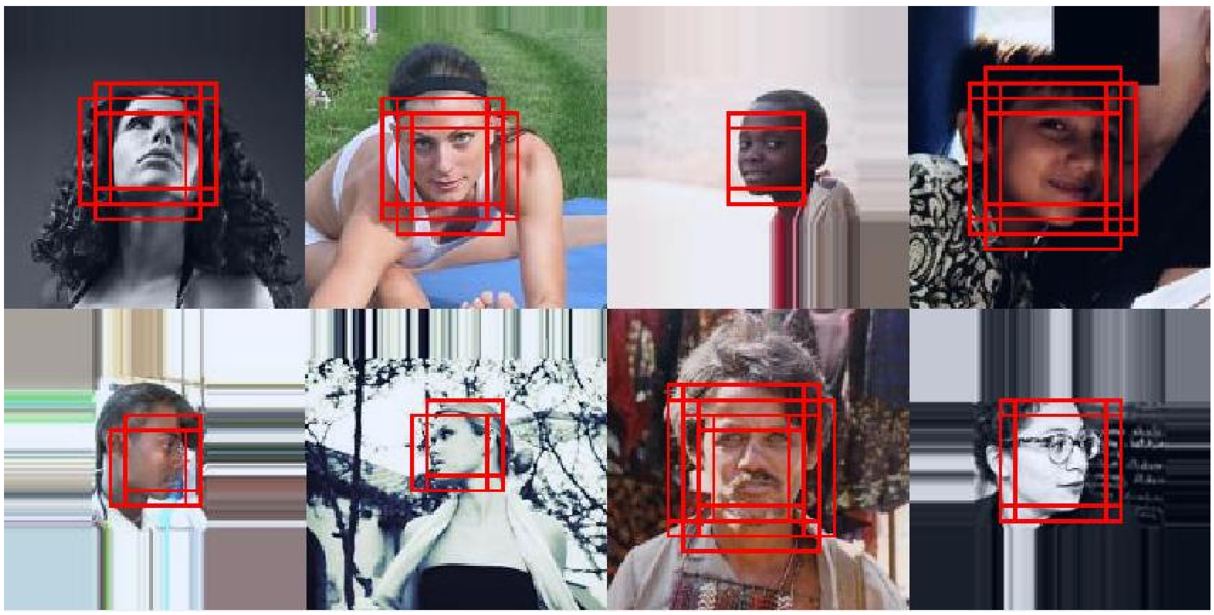


Figure 2: An example in our training dataset. The region with a red rectangle is labeled as positive because its IoU overlap with one ground-truth box is higher than the pre-set threshold.

model. In [4], Farfade *et al*. introduce deep dense face detector (DDFD), which fine-tunes the pre-trained AlexNet [13] is able to detect faces in a wide range of orientations. In [8], Huang *et al*. propose an end-to-end FCN framework called DenseBox for face detection. The performance of DenseBox [29] is slightly better than our MS-FCN model, however, DenseBox is trained with more labeled data and facial landmark information. DDFD and DenseBox only train one single scale model and have to build more levels of image pyramids during testing. Moreover, DDFD, Faceness-Net and DenseBox have wider and deeper convolution layers compared to the architecture of CNN used in our model. Therefore, DDFD, Faceness-Net and DenseBox are inefficient compared with our method.

3 MS-FCN Model

The whole detection system of our MS-FCN model is illustrated in Fig. 1. Given an image of any size, our detection system simultaneously outputs multiple predicted bounding boxes, each with a class confidence. In more detail, an octave of the image pyramids is taken as the input (Fig. 1 (A)) and passed through the shared convolutional (*conv*) layers and the *conv* feature maps are output (Fig. 1 (B)). Every scale face model slides on the feature maps and we get the regression (*reg*) and classification (*cls*) outputs (Fig. 1 (C)). Finally, we convert the *reg* and *cls* outputs to bounding box with score, apply non-maximum suppression (NMS) to those boxes whose confidence is above the predefined threshold and get the detection results (Fig. 1 (D)). In the following, we give a detailed description on how to train our multi-scale face detector.

3.1 Data Preparation

Enough training data is very important for good performance of CNN model. To train our MS-FCN face detector, we use the Annotated Facial Landmarks in the Wild (AFLW) database [12] to generate positive examples. The AFLW database contains $25,993$ face

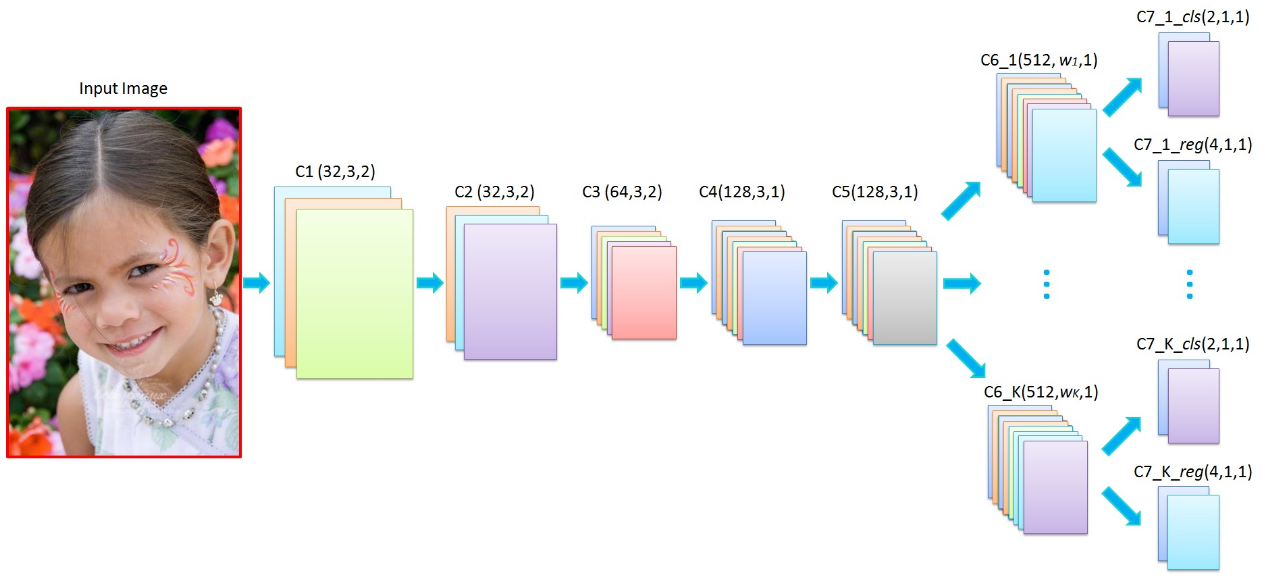


Figure 3: The architecture of our MS-FCN network. In the notation $str(n,s,k)$, str represents the layer name and n,s,k mean the number of filters and the filter size and the stride. Our multi-scale face models share the same *conv* feature maps from $C1$ to $C5$. K models with different spatial windows w_i slide on the shared $C5$ feature maps and are followed by two sibling fully-connected layers, a *cls* layer and a *reg* layer.

annotations in $21,997$ images collected from Flickr. This unconstrained database includes many faces with large variations in pose, expression, ethnicity, age, illumination, etc.

If the network takes a whole image as the input for training, it would spend most computational time in convolving on background. Obviously, this is an unwise strategy. To accelerate the training process, we crop large patches containing faces and sufficient background information for training. A patch is cropped and resized to 160×160 with a face in the center that has roughly specific height t_k, where t_k is the template size of the k-th scale model. And any face labeled in the AFLW dataset is cropped K times. After patches have been cropped, they are randomly sampled, horizontally flipped with probability 0.5 and arranged together to form a mosaic of faces, as shown in Fig. 2. This makes our model "see" more faces at one time during training and the diversity of faces leads to more steady training process.

For training MS-FCN, we assign a binary label (of being a face or not) to each spatial region. A region that has an intersection of union (IoU) overlap higher than 0.65 with any ground-truth box will be assigned a positive label. As shown in Fig. 2, regions with red rectangles are labeled as positive. A region will be assigned a negative label if its IoU ratio is lower than 0.4 for all ground-truth boxes. Those regions that are neither positive nor negative are ignored and will not contribute to the training objective.

3.2 Architecture

There are a few outstanding architectures like AlexNex [13] and VGGNet [21] which achieve top performance on image classification. These architectures are directly used in DDFD [4], Faceness-Net [29] and DenseBox [8] and have obtained excellent detection results. However, these models are computation-intensive and lead to inefficiency of detector. Rather than using these architectures which are designed for general image classification, we argue that we are able to use a tiny network for specific object detection (such as face) and lead to excellent performance.

Fig. 3 illustrates the architecture of our MS-FCN network. Inspired by [21], all filter sizes are 3×3. At the bottom of the network, there are 5 convolution layers which output the shared feature maps. Following that, there are K mini-networks to deal with faces at different scales between two consecutive octaves. Each mini-network contains one fully connected layer with filter size $w_i \times w_i$ and two sibling fully-connected layers (*reg* and *cls*) with filter size 1×1. The K mini-networks operate in a sliding-window fashion, and the fully-connected layers are shared across all spatial locations.

From Fig. 3, we can also see that the number of filters in each layer is small, which reduces the computation burden. Moreover, there are no pooling layers. Instead a convolution layer with a stride of 2 is used in the first 3 layers inspired by the work in [22]. This also saves the computation and memory and increase the efficiency. In Sec. 4, we demonstrate that our detector can achieve the top performance on public face detection benchmarks with this tiny architecture of CNN.

In our current implementation, the MS-FCN contains $K = 3$ scale models. The template sizes are $t_k \in \{40, 56, 72\}$ for different scales. Because the size of convolutional feature map is one eighth of the original image, that is, the fully-connected window size is $w_k = \lfloor \frac{1}{8} t_k \rfloor$. And we can get the spatial size $w_k \in \{5, 7, 9\}$. In one octave, we only scan faces with height between $40 \sim 72$ pixels and faces larger than 72 can be detected at the following octaves. There are no finely-sampled images between two octaves. Therefore, we do not need to compute the features of finely-sampled images, which contributes the efficiency of our detector.

3.3 Multi-Scale Multi-Task Training

As introduced in Sec. 3.2, the k-th network has two sibling output layers, *cls* and *reg*. The first *cls* layer produces the confidence score y_{ki} of being a target object. That is, y_{ki} is the predicted probability of $w_k * w_k$ spatial region centered at pixel i in the shared *conv* feature maps. Given the ground truth label $y^*_{ki} \in \{0, 1\}$, the classification loss is the softmax loss of two classes and can be defined as:

$$L_{cls}(y_{ki}, y^*_{ki}) = y^*_{ki} \log(y_{ki}) + (1 - y^*_{ki}) \log(1 - y_{ki}) \tag{1}$$

The second *reg* layer of our k-th network outputs the 4 parameterized coordinates of the predicted bounding box $d_{ki} = \{d_x, d_y, d_w, d_h\}_{ki}$. And we represent the ground-truth box $d^*_{ki} = \{d^*_x, d^*_y, d^*_w, d^*_h\}_{ki}$ associated with the k-th spatial region centered at pixel i. Then, we utilize the regression loss proposed in [6] which is formulated as follows:

$$L_{loc}(d_{ki}, d^*_{ki}) = \sum_{j \in \{x,y,w,h\}} \mathrm{smooth}_{L_1}(d^*_j - d_j)_{ki} \tag{2}$$

where

$$\mathrm{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } ||x|| < 1 \\ ||x|| - 0.5 & \text{otherwise} \end{cases} \tag{3}$$

is a robust L_1 loss that is less sensitive to outliers than L_2 loss.

With these definitions, we can minimize the following multi-scale multi-task loss L at each pixel i in the shared *conv* feature maps to jointly train for classification and bounding-box regression:

$$L(y_i, d_i) = \sum_{k=1}^{K} \gamma_k (L_{cls}(y_{ki}, y_{ki}^*) + \lambda y_{ki}^* L_{loc}(d_{ki}, d_{ki}^*)) \tag{4}$$

where $y_i = \{y_{1i}, ..., y_{Ki}\}$ and $d_i = \{d_{1i}, ..., d_{Ki}\}$ denote the K predicted labels and bounding boxes located at the i-th pixel, respectively. γ_k balances the importance of models at different scales. γ_k is set as 1, which means that all models show the same importance to us. The term $y_{ki}^* L_{loc}(d_{ki}, d_{ki}^*)$ means the regression loss is activated only for the positive region ($y_{ki}^* = 1$) and is disabled otherwise ($y_{ki}^* = 0$). For background regions, there is no notion of a ground-truth bounding box and hence L_{loc} is ignored. λ is the loss-balancing parameter and is set to 5, which means that we bias towards better box locations.

3.4 Optimization

As seen in Fig. 3, MS-FCN is naturally a fully-convolutional network [16], and can be trained end-to-end by back-propagation and stochastic gradient descent (SGD). The sampling strategy from [6] [18] can be applied to train our network. Each mini-batch contains many positive and negative examples that are sampled from a single image, shown in Fig. 2. There are K models in our system, and we keep the same number of training examples of different scales. The negative samples dominate in all samples, which will contribute to biased prediction towards negative if all of them are used to compute the loss function of a mini-batch. To avoid this degradation, we keep the sampled positive and negative regions at a ratio of $1:1$. And we use mini-batches of size $R = 512$ for each scale. Hard negative example mining strategy in DenseBox [8] is also utilized to make training more efficient.

The weights of the filters of all layers are initialized by randomly drawing from a zero-mean Gaussian distribution with standard deviation 0.01. Biases were initialised to 0.1. The learning rate is initially set to 0.01 and then reduced by factor of 10 after every $100k$ mini-batches. And the learning was stopped after $300k$ iterations. We also use a momentum coefficient of 0.9 and a weight decay factor of 0.0005. Our system is implemented in Caffe [10].

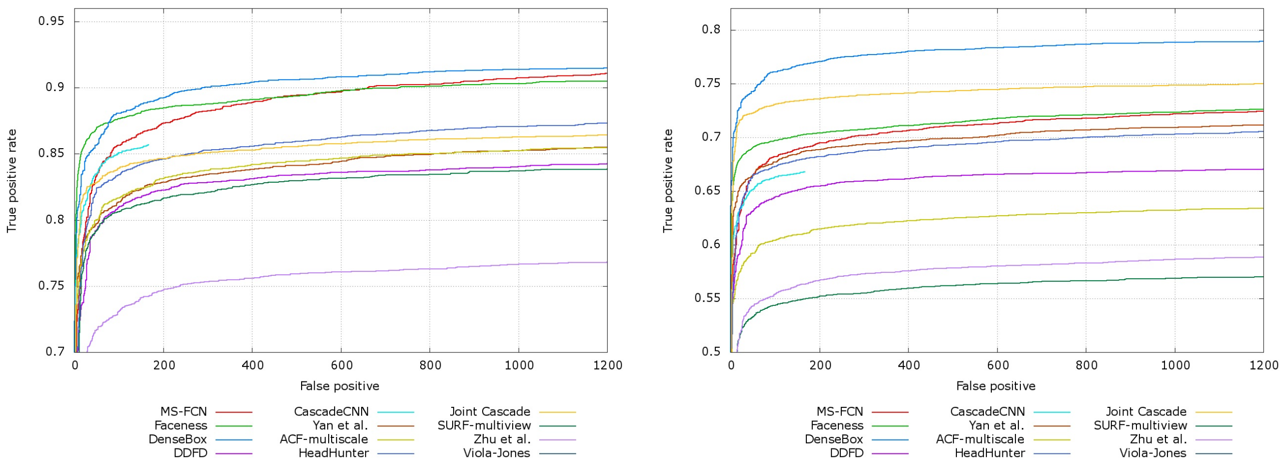
4 Experiments

We evaluate the proposed detector on three public face detection benchmarks, including FDDB, AFW, and PASCAL faces and compare our approach against the state-of-the-art ones. From the comparison, we can see that our detector can achieve top detection performance while running at super real-time speed.

4.1 Evaluation on FDDB

The FDDB dataset [9] is a challenging benchmark for face detection. It contains $2,845$ images with a total of $5,171$ faces, in a wide range of challenging scenarios including arbitrary posea, occlusions, and blurred faces. All faces in FDDB have been annotated with elliptical regions.

An evaluation toolbox is provided in [9] for comparisons of different face detection algorithms. There are two metrics for performance evaluation: the discrete score and continuous



(a) Discontinuous score (b) Continuous score

Figure 4: On the FDDB dataset we compare MS-FCN detector with the state-of-the-art methods including: Faceness [29], DenseBox [8], DDFD [4], CascadeCNN [14], Yan *et al.* [26], ACF-multiscale [28], HeadHunter [17], Joint Cascade [1], SURF-multiview [15], Zhu *et al.* [31] and Viola-Jones [24].

score. The discontinuous score metric counts the number of detected faces versus the number of false alarms. The detection result is regarded as true positive only if it has an IoU above 0.5 to a ground-truth face. In the continuous score metric, the IoU ratio is considered as the matching metric of the detection bounding box. The above two metrics correspond to coarse match and precise match between the detection and the ground truth, respectively.

To match the ellipse annotation on FDDB better, we uniformly transform our square detection bounding boxes to the ellipse ones. As shown in Fig. 4, the proposed method outperforms most of the baseline methods in both the discontinuous and continuous score metrics. Compared to other CNN-based detectors, our detector shows superior performance over CascadeCNN, DDFD and Faceness detectors and is slightly inferior to DenseBox under the discontinuous score metric. However, DenseBox is trained with three times more training data than our detector and the landmark information is used to boost the performance. Although Faceness is also trained with additional face attribute information, it still shows slightly inferior performance compared to our detector.

4.2 Evaluation on AFW

AFW dataset is built using Flickr images by Zhu *et al.* [31]. It has only 205 images with 473 annotated faces. However, the images tend to contain cluttered background and faces in AFW is with large variations in both face viewpoint and appearance (aging, sunglasses, make-ups, skin color, expression etc.). Therefore, it is very challenging for detectors to achieve good performance.

We evaluate our detector on AFW and the precision-recall curves are shown in Fig. 5. Our MS-FCN detector achieves an average precision (AP) value of 97.7%, compared to 97.2% achieved by the Faceness detector. Our detector also outperforms other state-of-the-art methods by a large margin.

4.3 Evaluation on PASCAL faces

PASCAL face dataset [27] is another widely used face detection benchmark. It consists of 851 images and 1,341 annotated faces. Faces in this dataset is also with large variations in both face viewpoint and appearance.

We test our detector on PASCAL faces and our method performs surprisingly well. Fig. 6 shows the precision-recall curves. Our MS-FCN detector achieves an AP value of 91.8% and outperforms most state-of-the-art methods by a large margin. Our MS-FCN show slightly inferior performance compared to the Faceness detector with an AP value of 92.1%.

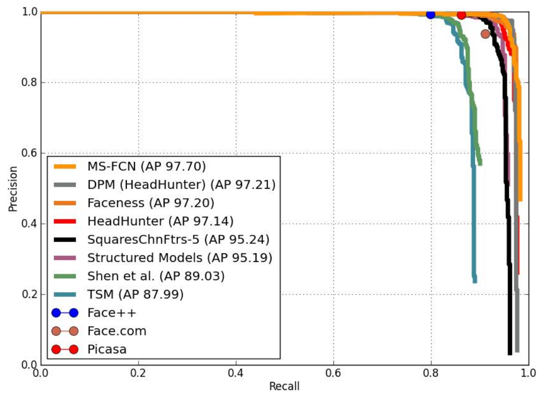


Figure 5: On the AFW dataset we compare MS-FCN detector with the state-of-the-art methods including: Faceness [29], HeadHunter [17], Structured Models [27], Shen *et al.* [20], DPM [5] [17], TSM [31], Face.com, Face++ and Picasa. AP: average precision.

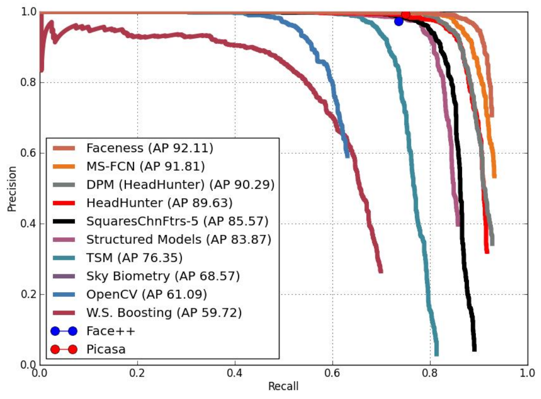


Figure 6: On the Pascal faces dataset we compare MS-FCN detector with the state-of-the-art methods including: Faceness [29], HeadHunter [17], Structured Models [27], DPM [5] [17], TSM [31], W.S. Boosting [11], OpenCV, Sky Biometry, Face++ and Picasa. AP: average precision.

4.4 Runtime Efficiency

One of the important advantages of our MS-FCN detector is its efficiency. Our detector contains K models at different scales. More importantly, they share the same *conv* features. When detecting faces in images, our detector calculates feature maps of an octave only one time and the K models can detect faces at different scales between two consecutive octaves.

CascadeCNN [14] is also designed considering the efficiency and can run at 100 fps on the GPU. The cascade framework is used in CascadeCNN and a network with a lower resolution is used at the first stage which rejects most false positive regions in an image. Deeper and wider networks in following stages are applied to evaluate remaining detection windows finely. However, the threshold in each stage needs to be verified carefully, otherwise, the detector's performance may drop quickly. With the same setting, our detector can run at nearly 100 fps, which is as fast as CascadeCNN [1].

Although the performance of DenseBox on the FDDB dataset and Faceness on the Pascal faces dataset is slightly better than our detector, these two detectors are much slower. In [8], it reports that DenseBox needs several seconds to process one image. Faceness is also low in efficiency. In [29], a fast version of Faceness is proposed and can run at 20 fps on GPU,

[1]Our detector was tested on Nvidia GTX 980 while CascadeCNN was tested on Nvidia TITAN BLACK.

however, the performance drops a lot and only achieves a 87% recall rate on FDDB, which is much lower than 90.7% achieved by our MS-FCN detector.

5 Conclusion

In this paper, we propose a multi-scale fully convolutional network for face detection. The K models in our system share the same convolutional features and can be trained end-to-end. During testing, images at octave-spaced scale intervals in the pyramid need to be processed by our detector. And faces at different scales at different scales between two consecutive octaves can be detected by the K models at different scales. Experiments demonstrate that our methods outperform most of the state-of-the-art methods across several challenging benchmarks, including FDDB, PASCAL Faces, and AFW, while keeping real-time performance.

6 Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants No. 61303059 and No. 61572296.

References

[1] Dong Chen, Shaoqing Ren, Yichen Wei, Xudong Cao, and Jian Sun. Joint cascade face detection and alignment. In *Computer Vision–ECCV 2014*, pages 109–122. Springer, 2014.

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[3] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36 (8):1532–1545, 2014.

[4] Sachin Sudhakar Farfade, Mohammad J Saberian, and Li-Jia Li. Multi-view face detection using deep convolutional neural networks. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 643–650. ACM, 2015.

[5] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[6] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[8] Lichao Huang, Yi Yang, Yafeng Deng, and Yinan Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015.

[9] Vidit Jain and Erik G Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. *UMass Amherst Technical Report*, 2010.

[10] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.

[11] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk. Weighted sampling for large-scale boosting. In *BMVC*, pages 1–10, 2008.

[12] Martin Köstinger, Paul Wohlhart, Peter M Roth, and Horst Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 2144–2151. IEEE, 2011.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5325–5334, 2015.

[15] Jianguo Li and Yimin Zhang. Learning surf cascade for fast and accurate object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3468–3475, 2013.

[16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[17] Markus Mathias, Rodrigo Benenson, Marco Pedersoli, and Luc Van Gool. Face detection without bells and whistles. In *Computer Vision–ECCV 2014*, pages 720–735. Springer, 2014.

[18] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[19] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1): 23–38, 1998.

[20] Xiaohui Shen, Zhe Lin, Jonathan Brandt, and Ying Wu. Detecting and aligning faces by image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3460–3467, 2013.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[22] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[23] Régis Vaillant, Christophe Monrocq, and Yann Le Cun. Original approach for the localisation of objects in images. In *Vision, Image and Signal Processing, IEE Proceedings-*, volume 141, pages 245–250. IET, 1994.

[24] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[25] Junjie Yan, Xucong Zhang, Zhen Lei, and Stan Z Li. Real-time high performance deformable model for face detection in the wild. In *Biometrics (ICB), 2013 International Conference on*, pages 1–6. IEEE, 2013.

[26] Junjie Yan, Zhen Lei, Longyin Wen, and Stan Li. The fastest deformable part model for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2497–2504, 2014.

[27] Junjie Yan, Xuzong Zhang, Zhen Lei, and Stan Z Li. Face detection by structural models. *Image and Vision Computing*, 32(10):790–799, 2014.

[28] Bin Yang, Junjie Yan, Zhen Lei, and Stan Z Li. Aggregate channel features for multi-view face detection. In *Biometrics (IJCB), 2014 IEEE International Joint Conference on*, pages 1–8. IEEE, 2014.

[29] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. From facial parts responses to face detection: A deep learning approach. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3676–3684, 2015.

[30] Stefanos Zafeiriou, Cha Zhang, and Zhengyou Zhang. A survey on face detection in the wild: Past, present and future. *Computer Vision and Image Understanding*, 138:1 – 24, 2015. ISSN 1077-3142.

[31] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE, 2012.