

Hierarchical Convolutional Neural Network for Face Detection

Dong Wang^(✉), Jing Yang, Jiankang Deng, and Qingshan Liu

B-DAT Lab, School of Information and Control, Nanjing University of Information
Science and Technology, Nanjing, China

{wangdong2013nuist,yang.xiaojing00,dengjiankang6}@gmail.com,

qsliu@nuist.edu.cn

<http://bdat.nuist.edu.cn/>

Abstract. In this paper, we propose a new approach of hierarchical convolutional neural network (CNN) for face detection. The first layer of our architecture is a binary classifier built on a deep convolutional neural network with spatial pyramid pooling (SPP). Spatial pyramid pooling reduces the computational complexity and remove the fixed-size constraint of the network. We only need to compute the feature maps from the entire image once and generate a fixed-length representation regardless of the image size and scale. To improve the localization effectiveness, in the second layer, we design a bounding box regression network to refine the relative high scored non-face output from the first layer. The proposed approach is evaluated on the AFW dataset, FDDB dataset and Pascal Faces, and it reaches the state-of-the-art performance. Also, we apply our bounding box regression network to refine the other detectors and find that it has effective generalization.

Keywords: Convolutional neural network · Hierarchical · Spatial pyramid pooling · Face detection

1 Introduction

In the field of computer vision, face detection achieved much success in these years. A capable face detector is a pre-condition for the task of facial keypoint localization and face recognition. In the anthropocentric world, face detection is widely used in a myriad of consumer products (e.g. social networks, smart mobile device applications, digital cameras, and even a wearable device).

In fact, detecting face “in the wild” is always to be a challenge to traditional detection methods due to large variations in facial appearances, occlusions, and clutters. As the textbook version of a face detector, the Viola & Jones architecture [1] has been the source of inspiration for countless variants such as SURF cascades [2]. They support very fast face detection but the features have limited representation capacity. So, they cannot differentiate faces in uncontrolled environments. The part based models [3, 4] have been proposed to address these problem. These models benefit from the fact that the part of an object often

presents less visual variations, and the global variation of the object can be modeled by the flexible configuration of different parts. However, they face the same issue as the previous feature-based holistic face detectors [1], because the part modeling is still based on the low level descriptors.

In recent years, convolution neural networks(CNNs) [5–7] became very popular. Convolution neural network has powerful learning ability and it has been widely used in the field of computer vision tasks such as image classification [8,9], object detection [10], scene parsing [11], facial point detection [12], and face parsing [13]. Studies in [9] show that a large, deep convolutional neural network was capable of achieving record-breaking results on a highly challenging dataset using purely supervised learning. The leading object detection method R-CNN [10] combines region proposals with CNNs and shows remarkable detection accuracy. Reference [14] proposed a new SPP-net, in which the spatial pyramid pooling (SPP) [15,16] layer is introduced to remove the fixed-size constraint of the network. Different from R-CNN [10] that repeatedly applies the deep convolutional networks to thousands of warped regions per image, SPP-net run the convolutional layers only once on the entire image, and then extract features by SPP on the feature maps. SPP-net is faster than R-CNN over one hundred times.

To address these challenges mentioned above, we propose a new face detector, which is based on a hierarchical convolution neural network. We consider a structure combination of deep convolutional neural network and the spatial pyramid pooling. This combination helps us to obtain powerful feature representation and fast running speed. We build the first layer of our architecture by applying the spatial pyramid pooling layer to a modification on the network in [9]. We make a structural modifications to the network in [9] by replacing the last pooling layer (e.g. *pool*₅, after the last convolutional layer) with a spatial pyramid pooling layer. SPP layer is applied on each candidate window of the *conv*₅ feature maps to pool a fixed-length representation of this window. The first layer of our structure in fact is a binary classifier of face and non-face. We name the binary classifier the “*FaceHunter*”. We observe in the experiment that some bounding boxes of our *FaceHunter* deviate from the center of the face area after being integrated in the stage of non-maximum suppression [3]. So we introduce a CNN-based calibration layer (i.e. the second layer) to improve the localization effectiveness. It is a 2-level CNN-Refine structure for refining the relative high scored non-face output from our *FaceHunter*. Recent work [12] shows that a deep convolutional network based regression can reach top results in facial point detection task. Inspired by [12], we apply their approach in this paper and find that it can do well in face detection as well. The first level of our CNN-Refine structure, similar to [12], is a regression approach with a deep convolutional neural network. The second level is designed to judge the new boxes which come from the first level. In this level, we put the regressed boxes into the same classifier as *FaceHunter* to preserve the positive faces and delete the negative faces. See details in Sect. 3.2.

To evaluate our approach, we run our detector respectively on AFW [4], FDDB [17] and Pascal Faces dataset [18]. All the datasets contain faces in uncontrolled conditions with cluttered backgrounds and large variations in both face

viewpoint and appearance, and thus bring forward great challenges to the current face detection algorithms. Surprisingly, our face detector puts up a good show and reach the state-of-the-art performance. We also evaluate our CNN-Refine structure that is applied to two different algorithms (i.e. Structured Models [18] and OpenCV [1]) on two public datasets. The result shows that it has effective generalization.

2 Overview

Figure 1 shows the construction of our two-layer hierarchical deep detector, which is a SPP-based face detector (i.e. *FaceHunter*) followed by a 2-level CNN-Refine structure. Once the picture is put into the *FaceHunter*, it will output initial face detecting results. The positive faces from *FaceHunter* will be directly output. However, the non-faces will not be deleted in general terms due to some ones of them in fact are close to the faces in the space of image. So we introduce a CNN-Refine structure to refine the negative output of first layer. The rects with relative high score, not all for efficiency, will be regressed by the first level of the CNN-Refine structure (i.e. *Deep CNN*). Then the regressed rects are fed to the second level which is a classifier the same as the first layer. It will finally delete the negatives and output the positives.

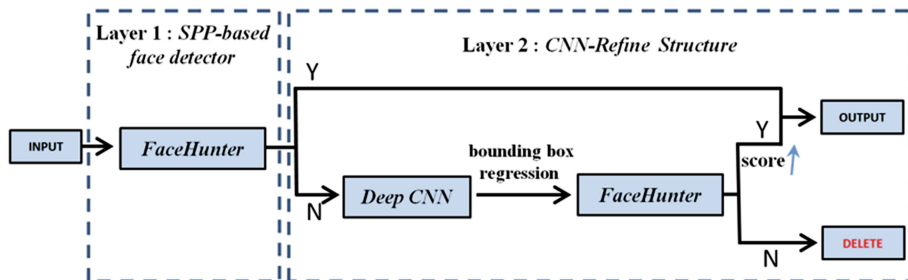


Fig. 1. The overview of our proposed face detector: a SPP-based face detector (i.e. *FaceHunter*) followed by a 2-level CNN-Refine structure.

3 Hierarchical Deep Detector

3.1 SPP-based Face Detector

We resize the image and extract the feature maps from the entire image only once. Then we apply the spatial pyramid pooling on each candidate window of the feature maps to pool a fixed-length representation. These representations are provided to the fully-connected layers of the network. Then we score these windows via a pre-trained binary linear SVM classifier and use non-maximum suppression on the scored windows.

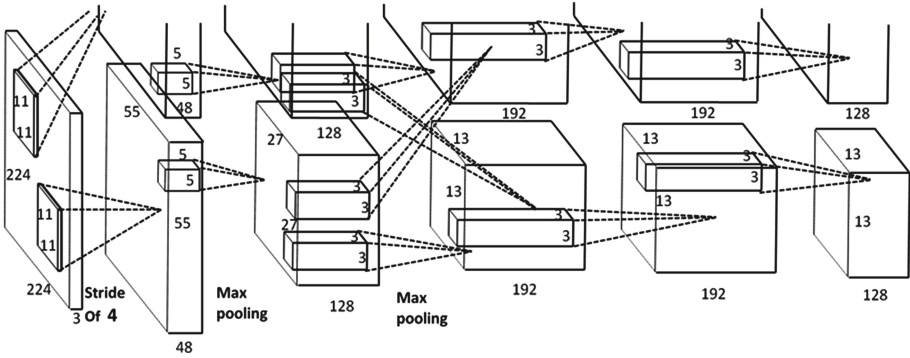


Fig. 2. The deep structure which is used for generating feature maps. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers.

Feature Maps. We consider a variant of the network in [9] to generate the feature maps, which removes the last pooling layer (i.e. $pool_5$). As depicted in Fig. 2, the net contains 5 layers. The kernels of the second, fourth, and fifth convolutional layers are connected only to those kernel maps in the previous layer. The kernels of the third convolutional layer are connected to all the kernel maps in the second layer. The response-normalization layers follow the first and second convolutional layers. The Max-pooling layers follow response-normalization layers. The first convolutional layer filters the input with 96 kernels of size $11 \times 11 \times 3$ with a stride of 4 pixels. The second convolutional layers input is the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size $5 \times 5 \times 48$. The third, fourth, and fifth convolutional layers are connected without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size $3 \times 3 \times 256$ connected to the outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size $3 \times 3 \times 192$, and the fifth convolutional layer has 256 kernels of size $3 \times 3 \times 192$.

The Spatial Pyramid Pooling Layer. Spatial pyramid pooling can maintain spatial information by pooling in local spatial bins. The size of these spatial bins is proportional to the image size, so the number of bins is fixed regardless of the image size. Spatial pyramid pooling uses the multi-level spatial bins, which has been shown to be robust to object deformations. After $conv_5$, as shown in Fig. 3, we use a 4-level spatial pyramid ($1 \times 1, 2 \times 2, 3 \times 3, 6 \times 6$, totally 50 bins) to pool the features on the feature maps. The $conv_5$ layer has 256 filters. So the spp layer generates a 12,800-d (i.e. 50×256) representation. We use the max pooling throughout this paper.

3.2 CNN-Refine Structure

Bounding Box Regression. Different detectors output different sizes and different styles of bounding boxes. We observe that some bounding boxes of our

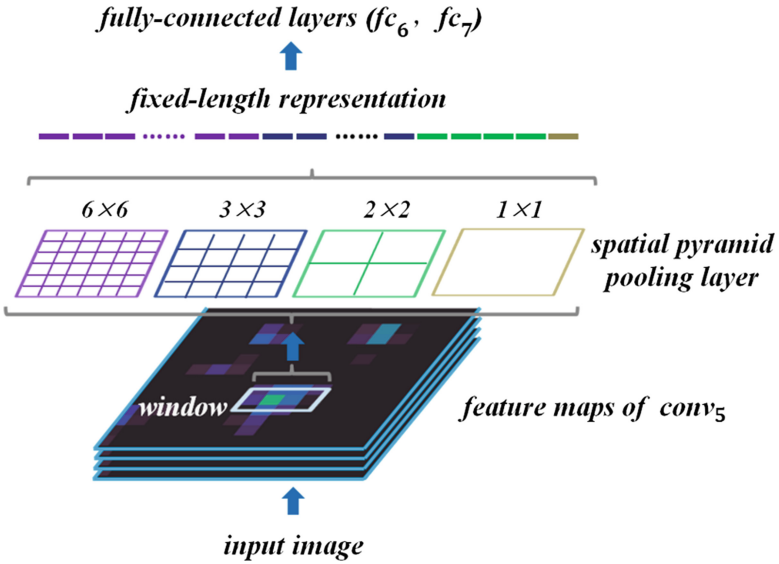


Fig. 3. The network with a spatial pyramid pooling layer: the feature maps are computed from the entire image and the pooling is performed in candidate windows.

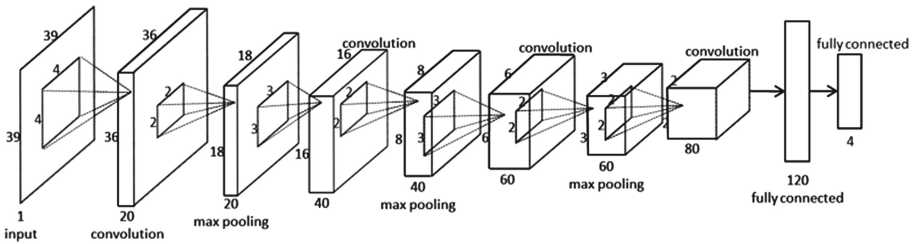


Fig. 4. The structure of *Deep CNN*: four convolutional layers followed by max pooling, and two fully connected layers.

FaceHunter deviate from the center of the face area after being integrated in the stage of non-maximum suppression. So, we introduce a CNN-based calibration layer to improve the localization effectiveness. We use a regression approach with a CNN structure to finely tune the bounding boxes which are not far from the faces in the space of image. We name these bounding boxes “*badbox*”.

In this paper, we focus on the structural design of individual networks. Figure 4 illustrates the deep structure, which contains four convolutional layers followed by max pooling, and two fully connected layers. Figure 5 is the overview of this level. The green box is the ground truth, while the red one is a *badbox*. The input of Deep CNN is the yellow box whose side length is 1.8 times the one of the *badbox*.

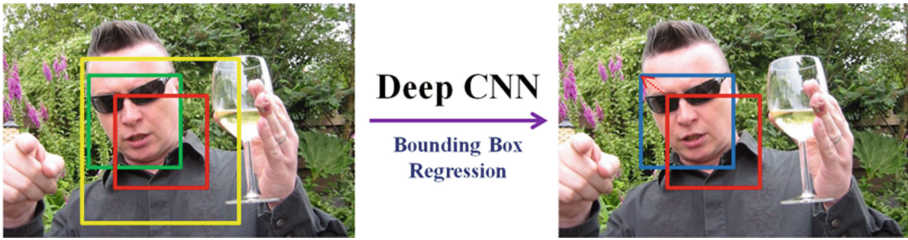


Fig. 5. The green box is the ground truth and the red one is the *badbox*. The input of *Deep CNN* is the yellow box whose side length is 1.8 times the one of the *badbox*. The output of *Deep CNN* is the location of the finely tuned bounding box (blue box) (Color figure online).

Re-score. After bounding box regression we use the classifier *FaceHunter* again to judge the output of *Deep CNN*. It is worth noticing that we no more need to run the convolutional layers on the entire image or the region of the regressed box. All we need is to pool the features for the regressed box on the feature maps which are generated in the first layer. Figure 6 is the overview of this level. Once the classifier judge that the input is a positive one, the score of corresponding bounding box will be multiplied by a motivational factor α . It can be written as

$$S = \frac{s}{|s|} (|s| * \alpha), p = 1. \tag{1}$$

where $\alpha > 1$, s is the previous confidence value of a “*badbox*” and S is the tuned confidence value by *FaceHunter*. p is the prediction of *FaceHunter*. The value 1 denote the input is a true face and 0 denote a fake one. It will be deleted directly in this level if it is judged as a negative face.

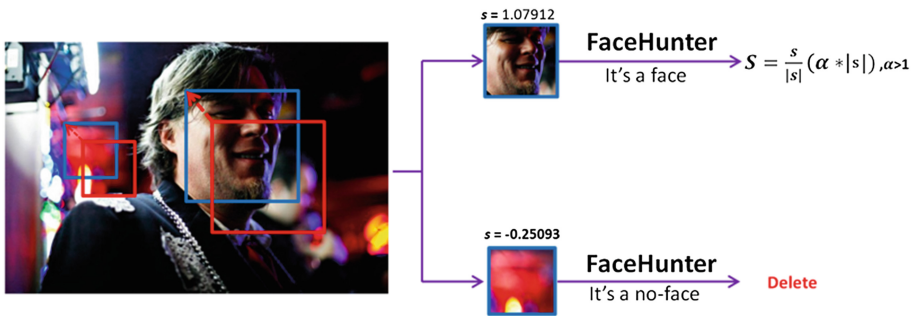


Fig. 6. The overview of the second level of our CNN-Refine structure: This level will encourage positive faces and delete the negative ones. “ s ” is the previous confidence value of a “*badbox*” and “ S ” is the tuned confidence value.

4 Experiments

4.1 Datasets and Experiment Setting

We use five datasets for our experiments: AFW [4] (205 images with bounding boxes), FDDB [17] (2845 images with ellipses annotations) Pascal Faces dataset [18] (851 Pascal VOC images with bounding boxes), AFLW [19] (26000 annotated faces), and a large face dataset of ourselves, which has a total of 980,354 annotated faces. All the datasets contain faces in uncontrolled conditions with cluttered backgrounds and large variations in both face viewpoint and appearance.

Training Setting. The training experiments are conducted in our own dataset and AFLW. 183,200 face images are selected from our own dataset to train *FaceHunter*, while the AFLW [19] dataset is used for validation. Specially, to train the detector, we generate positive samples by cropping image patches which are centered at ground truth windows, and negative samples are those that overlaps a positive window by at most 25% (measured by the intersection-over-union (IoU) ratio). To reduce redundant information resulting from overlapping negative samples, the negative sample that overlaps another negative sample by more than 75% will be removed. Our implementation is based on the publicly available code of cuda-convnet [9] and Caffe [20]. We consider a single-size (i.e. 224×224) training mode. We use the 4-level spatial pyramid to generate the representation for each entire image. These representations are provided to the fully-connected layers of the network. Then we train a binary linear SVM classifier for each category on these features. Our implementation of the SVM training is shown as follows [10, 14]. For the training of bounding box regression, we take training patches according to the face annotations, and augment them by small translation.

Testing Setting. In the general object detection task [10, 14], they use the “fast” mode of selective search [21] to generate 2,000 candidate windows per image. In our experiments, we observe that such method is not suitable due to the fact that faces in datasets suffer from large variations in facial appearances and very small faces. To cope with challenges from appearance variance, in our experiments, a low threshold is set to achieve 99% recall of the ACF [22] detector and each image output 500 windows on average. Then we resize the image to the size of 224×224 and extract the feature maps from the entire image. In each candidate window, we use the 4-level spatial pyramid to generate the representation for each window. These representations are provided to the fully-connected layers of the network. Then the binary SVM classifier is used to score the candidate windows. Finally, we use non-maximum suppression (threshold of 0.3) on the scored windows.

In order to solve the problem that faces appear at all scales in images, we improve our method by extracting feature at multi scales. We resize the image

to five scales (i.e. 480, 576, 688, 864, 1200) and compute the feature maps of conv5 for each scale. Then we empirically choose a single scale such that the scaled candidate window has a number of pixels closest to 224×224 for each candidate window. Then we only use the feature maps extracted from this scale to compute the feature of this window.

4.2 Experiment Results

Results on Three Public Datasets. We respectively evaluate our “*Face Hunter*” only (expressed using the red curve) and “CNN-Refined *FaceHunter*” (expressed using the blue curve) on AFW [4], FDDB [17] and Pascal Faces dataset [18]. The PR curve of our approach on the AFW [4] and Pascal Faces [18] dataset and the Discrete ROC curve on the FDDB [17] dataset are shown in Figs. 7, 8 and 9 respectively. Our *FaceHunter* detector can already reach top performance. Then our CNN-Refined structure can slightly improve the performance. The average precision of our detector can be enhanced by more than 1 percent after our whole CNN-Refine structure being applied to the detector. Figure 10 shows some examples of our detection results. As we can see, our detector can detect faces with different poses, in severe occlusions and cluttered background, as well as blurred face images.

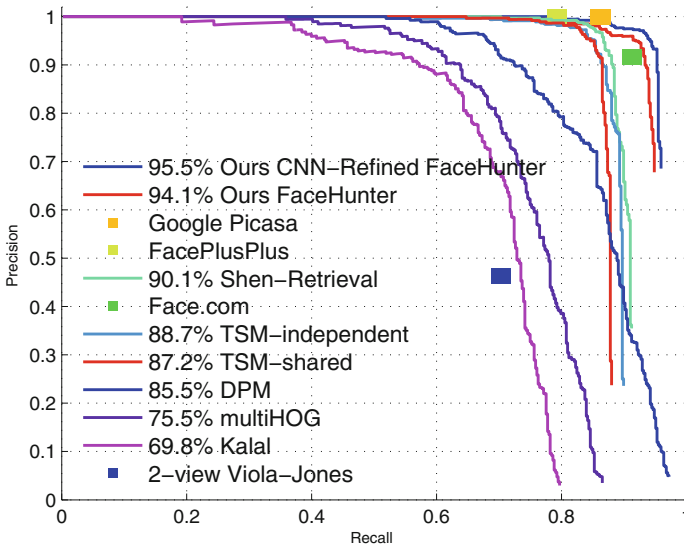


Fig. 7. PR Curve on AFW.

Evaluation about Our CNN-Refine Structure. Is the CNN-Refine structure only suitable for our *FaceHunter* detector? The answer is no. We evaluate it on two datasets for some other detectors and achieve exciting results. Figure 11 shows the evaluation about our CNN-Refine structure that is applied to two different algorithms (i.e. Structured Models [18] and OpenCV [1]) on two different datasets. Our CNN can also improve the performance of them.

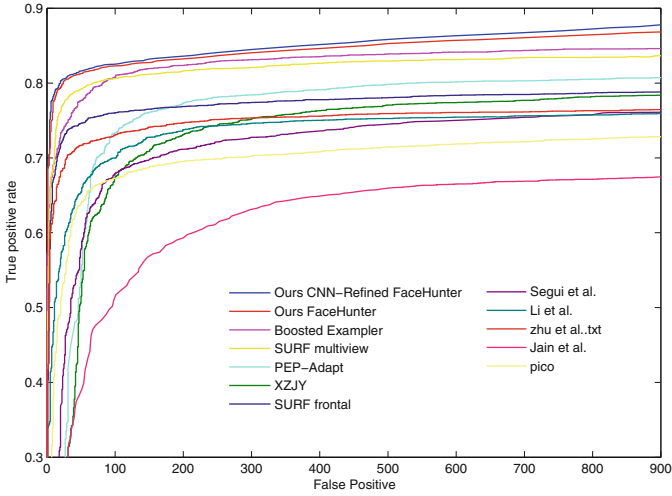


Fig. 8. Discrete ROC on FDDB.

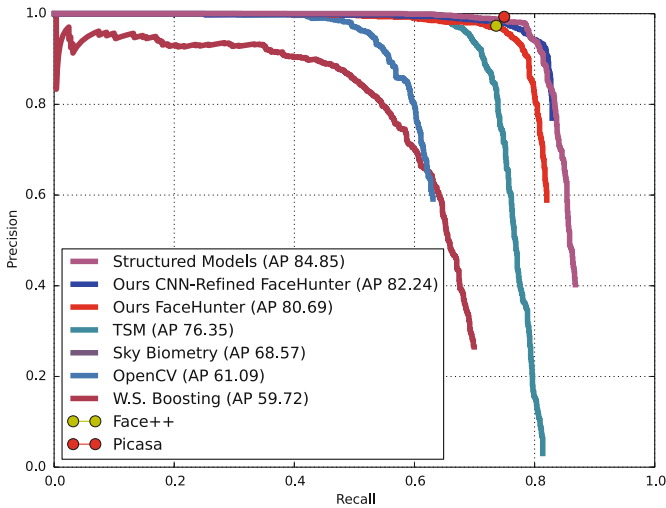


Fig. 9. PR Curve on Pascal Faces.

4.3 Computational Complexity Analysis

The complexity of the convolutional feature computation in ours *FaceHunter* is $O(r \cdot s^2)$ at a scale s (5-scale version in this work), where r is the aspect ratio and assume that r is about $4/3$. In contrast, this complexity of R-CNN is $O(n \cdot 227^2)$ with the window number n (2000). In our single-scale version (e.g. $s=688$), this complexity is about $1/160$ of R-CNN's. Even for our 5-scale version, this complexity is about $1/24$ of R-CNN's. So our method is much faster than



Fig. 10. Examples of our detection results.

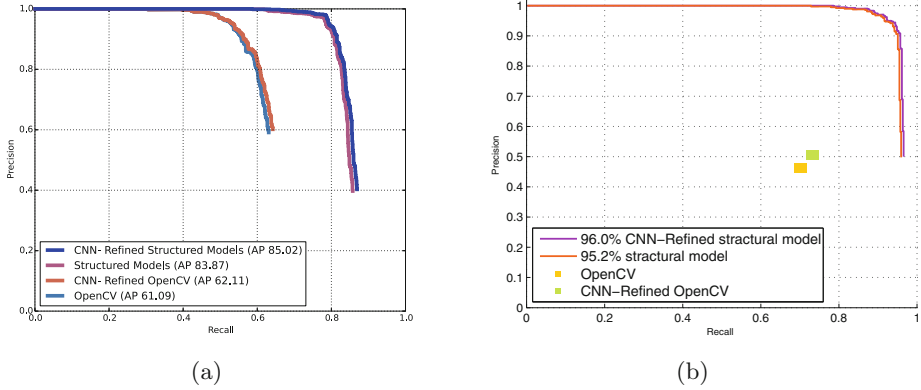


Fig. 11. (a). Results of CNN-Refined Structure Methods and OpenCV on Pascal Faces. (b). Results of CNN-Refined Structure Methods and OpenCV on AFW.

R-CNN. We evaluate the average time of 500 Pascal Faces [18] images using a single GeForce GTX Titan GPU (6 GB memory). Our 1-scale version takes only 0.05s per image and the 5-scale version takes 0.29s per image for convolutions. The GPU time of computing the 4,096-d fc7 features is 0.09s per image.

5 Conclusion

In this work we propose a hierarchical convolutional neural network (CNN) face detector that can achieve state-of-the-art performance. We introduce the spatial pyramid pooling (SPP) layer to remove the fixed-size constraint of the network. With SPP layer we can run the convolutional layers only once on the entire image, and then extract features by SPP on the feature maps. The bounding box regression approach that we have introduced to refine the detection results proves to have effective generalization. It is the first time that spatial pyramid pooling (SPP) is used in the task of face detection and our results show that it can achieve good performance on highly challenging datasets.

References

1. Viola, P., Jones, M.J.: Robust real-time face detection. *Int. J. Comput. Vis.* **57**(2), 137–154 (2004)
2. Li, J., Zhang, Y.: Learning surf cascade for fast and accurate object detection. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3468–3475. IEEE (2013)
3. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.* **32**(9), 1627–1645 (2010)
4. Zhu, X., Ramanan, D.: Face detection, pose estimation, and landmark localization in the wild. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2879–2886. IEEE (2012)
5. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.* **36**(4), 193–202 (1980)
6. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. *Neural Comput.* **1**(4), 541–551 (1989)
7. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Technical report, DTIC Document (1985)
8. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3642–3649. IEEE (2012)
9. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*, pp. 1097–1105 (2012)
10. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587. IEEE (2014)
11. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(8), 1915–1929 (2013)
12. Sun, Y., Wang, X., Tang, X.: Deep convolutional network cascade for facial point detection. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3476–3483. IEEE (2013)

13. Luo, P., Wang, X., Tang, X.: Hierarchical face parsing via deep learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2480–2487. IEEE (2012)
14. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition, arXiv preprint [arXiv:1406.4729](https://arxiv.org/abs/1406.4729)
15. Grauman, K., Darrell, T.: The pyramid match kernel: Discriminative classification with sets of image features. In: IEEE International Conference on ICCV, vol. 2, pp. 1458–1465. IEEE (2005)
16. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 2169–2178. IEEE (2006)
17. Jain, V., Learned-Miller, E.G.: Fddb: A benchmark for face detection in unconstrained settings, UMass Amherst Technical report
18. Yan, J., Zhang, X., Lei, Z., Li, S.Z.: Face detection by structural models. *Image Vis. Comput.* **32**(10), 790–799 (2014)
19. Kostinger, M., Wohlhart, P., Roth, P.M., Bischof, H.: Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In: IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp. 2144–2151. IEEE (2011)
20. Jia, Y.: Caffe: An open source convolutional architecture for fast feature embedding. <http://caffe.berkeleyvision.org>
21. Van de Sande, K.E., Uijlings, J.R., Gevers, T., Smeulders, A.W.: Segmentation as selective search for object recognition. In: 2011 IEEE International Conference on ICCV, pp. 1879–1886. IEEE (2011)
22. Yang, B., Yan, J., Lei, Z., Li, S.Z.: Aggregate channel features for multi-view face detection. In: 2014 IEEE International Joint Conference on Biometrics (IJCB), pp. 1–8. IEEE (2014)