Deep Convolutional Network Cascade for Facial Point Detection

Yi Sun[1] Xiaogang Wang[2,3] Xiaoou Tang[1,3]

[1]Department of Information Engineering, The Chinese University of Hong Kong

[2]Department of Electronic Engineering, The Chinese University of Hong Kong

[3]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences

sy011@ie.cuhk.edu.hk xgwang@ee.cuhk.edu.hk xtang@ie.cuhk.edu.hk

Abstract

We propose a new approach for estimation of the positions of facial keypoints with three-level carefully designed convolutional networks. At each level, the outputs of multiple networks are fused for robust and accurate estimation. Thanks to the deep structures of convolutional networks, global high-level features are extracted over the whole face region at the initialization stage, which help to locate high accuracy keypoints. There are two folds of advantage for this. First, the texture context information over the entire face is utilized to locate each keypoint. Second, since the networks are trained to predict all the keypoints simultaneously, the geometric constraints among keypoints are implicitly encoded. The method therefore can avoid local minimum caused by ambiguity and data corruption in difficult image samples due to occlusions, large pose variations, and extreme lightings. The networks at the following two levels are trained to locally refine initial predictions and their inputs are limited to small regions around the initial predictions. Several network structures critical for accurate and robust facial point detection are investigated. Extensive experiments show that our approach outperforms state-of-the-art methods in both detection accuracy and reliability[1].

1. Introduction

Facial keypoint detection is critical for face recognition and analysis, and has been studied extensively in recent years [3, 4, 5, 8, 9, 11, 20, 21, 23, 25, 26, 27, 28]. This problem is challenging when face images are taken with extreme poses, lightings, expressions, and occlusions, as shown in Figure 1. Existing approaches can be generally divided into two categories: classifying search windows [3, 4, 11, 20, 28] or directly predicting keypoint positions (or shape parameters) [5, 8, 9, 21, 25, 26]. For the first category, a classifier called component detector is trained for each keypoint and decision is made based on local regions. Since local features could be ambiguous or corrupted, multiple candidate regions which all look like the facial point or no suitable candidate region might be found. In that case, an optimal configuration of facial points is estimated with shape constraints [3, 4, 11, 20, 23, 28]. Compared with component detectors, directly predicting keypoint positions (or shape parameters) are more efficient since it does not need scanning. Regressors are often used as the predictor, based on local patches close to the facial point [9, 26], or the whole image region [5, 25]. Spatial constraints can also be added to regressors [25, 26].

Many approaches [5, 8, 11, 20, 21, 23, 25, 26] update the

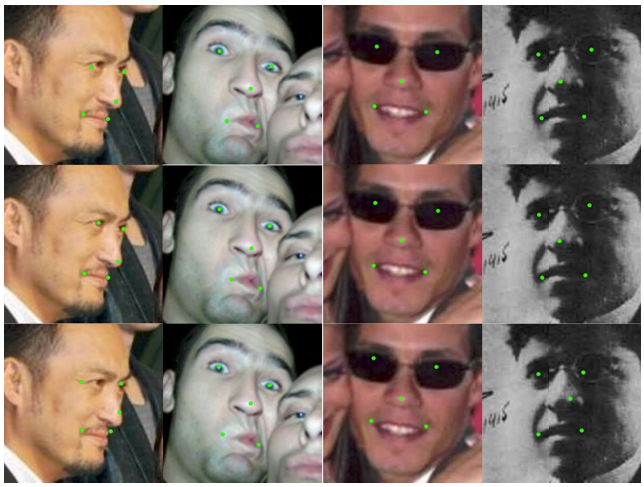


Figure 1: Examples of facial point detection. First row: initial detection with our first level of convolutional networks. It achieves good estimation with global context information even if some facial components are invisible or ambiguous in appearance. Second row: finely tuned results with our second and third levels of networks. The accuracy is improved. Third row: result from [5]. It is more restricted with shape templates learned from the training set and not accurate under some unusual poses and expressions.

[1]The dataset and code of this work can be found on the project webpage http://mmlab.ie.cuhk.edu.hk/CNN_FacePoint.htm

positions of facial points iteratively and good initializations are critical. The mean shape or shapes sampled from the training set is often used as the initialization, which may far from the target position, and the update may end with a local minimum. In addition, many approaches face the problem that the visual features extracted are not discriminative or not reliable enough to predict facial points, and context information becomes important. Most approaches employ shape constraints, which are relatively weak. It is desirable to directly extract texture context information over the whole face region, since they contain rich information. This requires much more powerful classifiers or regressors, since the visual complexity increases exponentially with the size of the image region.

To solve these problems, we propose a cascaded regression approach for facial point detection with three levels of convolutional networks. Different from existing approaches which roughly estimate the initial positions of facial points, our convolutional networks make accurate predictions at the first level, even on very challenging cases as shown in Figure 1. It effectively avoids the local minimum problem faced by other approaches. The convolutional networks take the full face as input to make the best use of texture context information, and extract global high-level features at higher layers of the deep structures, which can effectively predict keypoints even when low-level features from local regions are ambiguous or corrupted in challenging image examples. Our convolutional networks are trained to predict all the keypoints simultaneously and the constraints of keypoints are implicitly encoded.

The remaining two levels of convolutional networks refine the initial estimation of keypoints. Different from existing methods [5, 25, 26] which apply the same regressor at different cascade stages, we design different convolutional networks. The network structures at these two levels are shallower, since their tasks are low-level and their input is limited to small local regions around the initial positions. At each level, multiple convolutional networks are fused to improve the accuracy and reliability of estimation. Through detailed empirical investigation, we find that several factors regarding the network structures are critical for achieving good performance in facial point detection. Detailed experimental evaluations show that our approach outperforms state-of-the-art methods on both accuracies and reliability.

2. Related Work

Significant progress on facial keypoint detection has been achieved in recent years. Many used Adaboost [20], SVM [4, 28], or random forest [3] classifiers as component detectors and detection was based on local image features. Shape constraints are important to refine component detection results and much research has been focusing on this. The evidence given by local component detectors and the shape constraints can be balanced by optimizing designed objective functions [4, 28]. Liang *et al*. [20] trained a set of direction classifiers to guide the search of good shape. Amberg and Vetter [3] employed a branch and bound algorithm to efficiently find optimal configurations from a large number of candidates proposed by component detectors.

Among regression-based approaches, Dantone *et al*. [9] and Valstar *et al*. [26] predicted facial points from local patches with random forests and support vector regressors respectively. To resolve the uncertainties in predictions, Valstar *et al*. [26] modelled the spatial relations of facial points with Markov random field and Dantone *et al*. [9] fused many predictions from patches densely sampled within the face region. Patrick *et al*. [25] updated the parameters of an active appearance model with regressors. Cao *et al*. [5] used the whole face region as input and random ferns as the regressor. Shapes to be predicted were expressed as linear combinations of training shapes.

Convolutional networks and other deep models have been successfully used in vision tasks such as face detection and pose estimation [24], face parsing [22], image classification [6, 17], and scene parsing [10]. The research works on convolutional networks mainly focus on two aspects: network structures and feature learning algorithms. Coates *et al*. [7] analyzed the performance of single-layer networks with different filter strides, filter sizes, and the numbers of feature maps. Jarrett *et al*. [14] introduced strong nonlinearities after convolution, including absolute value rectification and local contrast normalization, and also compared different combinations of nonlinearities and pooling strategies. Not until recently has the potential of convolutional networks truly been discovered, when it becomes big (with hundreds of maps per layer) and deep (with up to five convolutional stages). By using large-scale convolutional networks, Ciresan *et al*. [6] significantly improved the state-of-the-art on some standard classification datasets. Even larger convolutional network was introduced in [17], and it significantly improved image classification accuracies on the ImageNet. Examples of recently proposed feature learning algorithms include convolutional sparse coding [16] and topographic independent component analysis [18].

3. Cascaded convolutional networks

In this paper, we focus on the structural design of individual networks and their combining strategies. Figure 2 is an overview of our approach. There are five facial points to be detected: *left eye center* (LE), *right eye center* (RE), *nose tip* (N), *left mouth corner* (LM), and *right mouth corner* (RM). We cascade three levels of convolutional networks to make coarse-to-fine prediction. At the first level, we employ three deep convolutional networks, F1, EN1, and NM1, whose input regions cover the whole face (F1), eyes and nose (EN1), nose and mouth (NM1). Each network si-

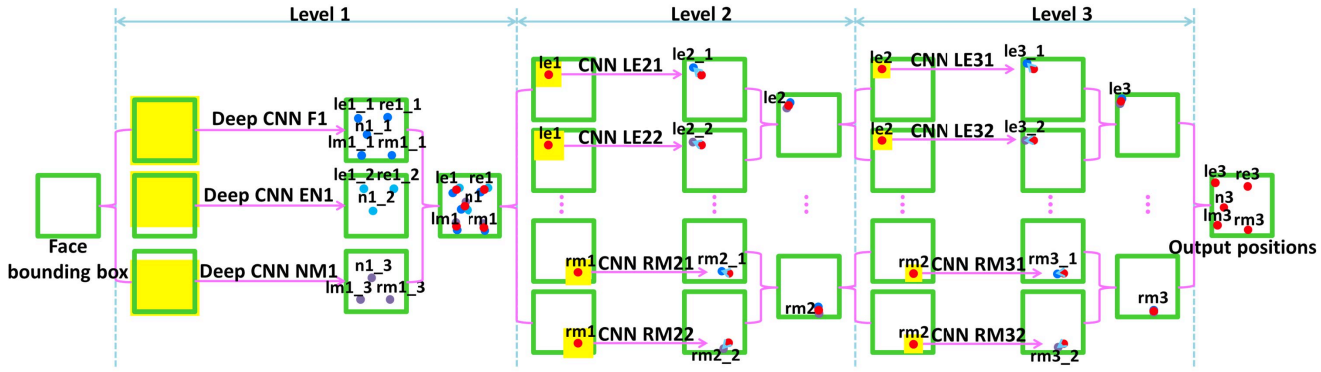


Figure 2: Three-level cascaded convolutional networks. The input is the face region returned by a face detector. The three networks at level 1 are denoted as F1, EN1, and NM1. Networks at level 2 are denoted as LE21, LE22, RE21, RE22, N21, N22, LM21, LM22, RM21, and RM22. Both LE21 and LE22 predict the left eye center, and so forth. Networks at level 3 are denoted as LE31, LE32, RE31, RE32, N31, N32, LM31, LM32, RM31, and RM32. Green square is the face bounding box given by the face detector. Yellow shaded areas are the input regions of networks. Red dots are the final predictions at each level. Dots in other colors are predictions given by individual networks.

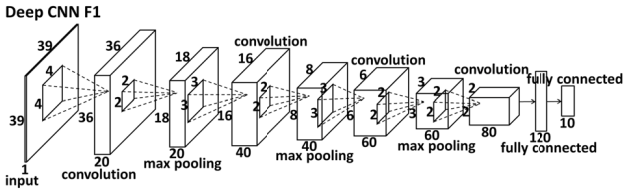


Figure 3: The structure of deep convolutional network F1. Sizes of input, convolution, and max pooling layers are illustrated by cuboids whose length, width, and height denote the number of maps, and the size of each map. Local receptive fields of neurons in different layers are illustrated by small squares in the cuboids.

multaneously predicts multiple facial points. For each facial point, the predictions of multiple networks are averaged to reduce the variance. Figure 3 illustrates the deep structure of F1, which contains four convolutional layers followed by max pooling, and two fully connected layers. EN1 and NM1 take the same deep structure, but with different sizes at each layer since the sizes of their input regions are different. Networks at the second and third levels take local patches centered at the predicted positions of facial points from previous levels as input and are only allowed to make small changes to previous predictions. The sizes of patches and search ranges keep reducing along the cascade. Predictions at the last two levels are strictly restricted because local appearance is sometimes ambiguous and unreliable. The predicted position of each point at the last two levels is given by the average of the two networks with different patch sizes. While networks at the first level aim to estimate keypoint positions robustly with few large errors, networks at the last two levels are designed to achieve high accuracy. All the networks at the last two levels share a common shallower structure since their tasks are low-level.

3.1. Network structure selection

We analyze three important factors on the choice of network structures. The discussions are limited to networks at the first level, which are the hardest to train. First, convolutional networks at the first level should be deep. Predicting keypoints from large input regions is a high-level task. Deeper structures help to form high-level features, which are global while features extracted by neurons at lower layers are local due to local receptive fields. By combining spatially nearby features extracted at lower layers, neurons at higher layers can extract features from larger regions. Moreover, high-level features are highly non-linear. Adding additional layers increases the non-linearity from input to output, and makes it possible to represent the relationship between input and output.

Second, for neurons in the convolutional layers, absolute value rectification after the hyperbolic tangent activation function (see details in Section 4) can effectively improve the performance. This modification over traditional convolutional networks was proposed in [14], where improvement on Caltech-101 was observed. Our empirical study shows that it is also effective in our application.

Third, locally sharing weights of neurons on the same map improves the performance. Traditional convolutional networks share weights of all the neurons on the same map based on two considerations. First, it assumes that the same features may appear everywhere in an image. So filters useful in one place should also be useful in others. Second, weight sharing helps to prevent gradient diffusion when back-propagating through many layers, since gradients of shared weights are aggregated, which makes supervised learning on deep structures easier. However, globally

sharing weights does not work well on images with fixed spatial layout, such as faces. For example, while eyes and mouth may share low-level features (*e.g.* edges), they are very different at high-level. So for networks whose inputs contain different semantic regions, locally sharing weights at high layers is more effective for learning different high-level features, *e.g.*, eyes, nose, and mouth. The idea of locally sharing weights was originally proposed for convolutional deep belief net for face recognition [12].

3.2. Multi-level regression

We find several effective ways to combine multiple convolutional networks. The first is multi-level regression. The face bounding box is the only prior knowledge for networks at the first level. The relative position of a facial point to the bounding box could vary in a large range due to large pose variations and the instability of face detectors. So the input regions of networks at the first level should be large in order to cover many possible predictions. But large input region is the major cause of inaccuracy because irrelevant areas included may degrade the final output of the network. The outputs of networks at the first level provide a strong prior for the following detections, *i.e.*, the true position of a facial point should lie within a small region around the prediction at the first level. So the second level detection can be done within a small region, where the disruption from other areas is reduced significantly, and this process repeats. However, without context information, appearance of local regions is ambiguous and the prediction is unreliable. To avoid drifting, we should not cascade too many levels or trust the following levels too much. These networks are only allowed to adjust the initial prediction in a very small range.

To further improve detection accuracy and reliability, we propose to jointly predict the position of each point with multiple networks at each level. These networks differ in input regions. The final predicted position of a facial point can be formally expressed as

$$x = \frac{x_1^{(1)} + \cdots + x_{l_1}^{(1)}}{l_1} + \sum_{i=2}^{n} \frac{\Delta x_1^{(i)} + \cdots + \Delta x_{l_i}^{(i)}}{l_i} \quad (1)$$

for an n-level cascade with l_i predictions at level i. Note that predictions at the first level are absolute positions while predictions at the following levels are adjustments.

4. Implementation details

The input layer is denoted by $I(h, w)$, where h and w are the height and width of the input region. The input is 2D since color information is not used. Convolutional layer is denoted by $CR(s, n, p, q)$, if absolute value rectification is used, otherwise $C(s, n, p, q)$. s is the side length of the square convolution kernels (or filters). n is the number of maps in the convolutional layer. p and q are weight sharing parameters. Each map in the convolutional layer is evenly divided into p by q regions, and weights are locally shared in each region. Traditional convolutional network can be viewed as a special case by setting $p = q = 1$. Filter stride is 1 pixel in both directions by default. Let (h, w, m) be the size of the previous layer, *i.e.*, m maps and each map of size h by w. Then the operation taken by $C(s, n, p, q)$ is

$$y_{i,j}^{(t)} = \tanh\left(\sum_{r=0}^{m-1}\sum_{k=0}^{s-1}\sum_{l=0}^{s-1} x_{i+k,j+l}^{(r)} \cdot w_{k,l}^{(r,u,v,t)} + b^{(u,v,t)}\right),$$

for $i = \Delta h \cdot u, \ldots, \Delta h \cdot u + \Delta h - 1$, $j = \Delta w \cdot v, \ldots, \Delta w \cdot v + \Delta w - 1$, $t = 0, \ldots, n-1$. $\Delta h = \frac{h-s+1}{p}$, $\Delta w = \frac{w-s+1}{q}$, $u = 0, \ldots, p-1$, and $v = 0, \ldots, q-1$. x and y are the outputs of the previous and current layers. w is weight and b is bias. The m maps in the previous layer are correlated with m s by s kernels. The resulting maps, together with a bias, are accumulated and passed the $\tanh$ nonlinearity, forming one of the n map in the convolutional layer. For different output maps and different regions in the maps, the set of kernels and the bias are different. $CR(s, n, p, q)$ is similar but has an additional abs operation after $\tanh$.

Pooling layer is denoted by $P(s)$. s is the side length of square pooling regions. Max pooling is used and the pooling regions are not overlapped. Pooling results are multiplied with a gain coefficient (g) and shifted by a bias (b), followed by a $\tanh$ non-linearity. The gain and bias coefficients are shared in a similar way as weights at the previous convolutional layer. $P(s)$ is formulated as

$$y_{i,j}^{(t)} = \tanh\left(g^{(u,v,t)} \cdot \max_{0 \le k,l < s}\left\{x_{i \cdot s+k, j \cdot s+l}^{(t)}\right\} + b^{(u,v,t)}\right).$$

Fully connected layer is denoted by $F(n)$ with function $y_j = \tanh\left(\sum_{i=0}^{m-1} x_i \cdot w_{i,j} + b_j\right)$, for $j = 0, \ldots, n-1$, where n and m are the numbers of neurons at the current layer and previous layer.

Structures. Networks at the first level are deep convolutional networks with four convolutional stages, absolute value rectification, and locally shared weights. Networks at the second and third levels share a common shallower structure. Since they are designed to extract local features, deep structures and locally sharing weights are unnecessary. Details of the network structures are summarized in Table 1.

Input ranges. F1 takes the whole face as input and outputs the positions of all the five points. EN1 takes the top and middle part of face as input and outputs positions of two eye centers and nose tip. NM1 takes the middle and bottom part of face as input and outputs positions of nose tip and two mouth corners. All the networks at the second and third levels take small squares centered at the positions predicted by the previous level as input and output an incremental

	layer 0	layer 1	layer 2	layer 3	layer 4	layer 5	layer 6	layer 7	layer 8	layer 9
S0	I(39,39)	CR(4,20,2,2)	P(2)	CR(3,40,2,2)	P(2)	CR(3,60,3,3)	P(2)	CR(2,80,2,2)	F(120)	F(10)
S1	I(31,39)	CR(4,20,1,1)	P(2)	CR(3,40,2,2)	P(2)	CR(3,60,2,3)	P(2)	CR(2,80,1,2)	F(100)	F(6)
S2	I(15,15)	CR(4,20,1,1)	P(2)	CR(3,40,1,1)	P(2)	F(60)	F(2)			
S3	I(39,39)	CR(4,20,2,2)	P(2)	CR(3,40,2,2)	P(2)	CR(3,60,3,3)	P(2)	F(120)	F(10)	
S4	I(39,39)	CR(4,20,2,2)	P(2)	CR(3,40,2,2)	P(2)	F(120)	F(10)			
S5	I(39,39)	CR(4,20,2,2)	P(2)	F(120)	F(10)					
S6	I(39,39)	C(4,20,2,2)	P(2)	C(3,40,2,2)	P(2)	C(3,60,3,3)	P(2)	C(2,80,2,2)	F(120)	F(10)
S7	I(39,39)	CR(4,20,1,1)	P(2)	CR(3,40,1,1)	P(2)	CR(3,60,1,1)	P(2)	CR(2,80,1,1)	F(120)	F(10)

Table 1: Summary of network structures. F1 adopts S0. Both EN1 and NM1 adopt S1. All the networks at the second and third levels share S2. To investigate different designs of network structures, we also compare different structures S3-S7 for F1 in experiments.

	net	left	right	top	bottom
L1	F1	-0.05	+1.05	-0.05	+1.05
	EN1	-0.05	+1.05	-0.04	+0.84
	NM1	-0.05	+1.05	+0.18	+1.05
L2	*21	-0.16	+0.16	-0.16	+0.16
	*22	-0.18	+0.18	-0.18	+0.18
L3	*31	-0.11	+0.11	-0.11	+0.11
	*32	-0.12	+0.12	-0.12	+0.12

Table 2: Summary of network input ranges, which are described by left, right, top, and bottom boundary positions. For networks at level 1 (L1), the four boundary positions are relative to the normalized face bounding box with boundary positions $(0, 1, 0, 1)$. For networks at level 2 (L2) and level 3 (L3), the four boundary positions are relative to the predicted facial point position.

prediction. At each of the two levels, we use two regions of different sizes to predict each point. Regions at the third level are smaller than the second level. The precise input ranges of all the networks are listed in Table 2.

Training. At the first level, we take training patches according to the face bounding box, and augment them by small translation and rotation. At the following levels, we take training patches centered at positions randomly shifted from the ground truth position. The maximum shift in both horizontal and vertical directions is 0.05 at the second level, and 0.02 at the third level, where the distances are normalized with the face bounding box. Networks at the third level aim at more subtle adjustment to previous predictions than those at the second level. Learnable network parameters include the weight w, the gain g, and the bias b, which are initialized by small random numbers and learned by stochastic gradient descent. Levenberg-Marquardt method [19] is used to estimate the neurons' learning rate individually. Training continues until converge.

5. Experiments

We first investigate different designs of network and cascade structures with a training set and a validation set collected by ourselves. Then we compare with the state-of-the-art methods and commercial software on two public test sets without changing the training set. Our training and validation sets have no overlap with the two public test sets.

5.1. Investigate network and cascade structures

We created a dataset with $13,466$ face images, among which $5,590$ images are from LFW [2] [13] and the remaining $7,876$ images are downloaded from the web. Each face is labeled with the positions of five keypoints. We randomly select $10,000$ images for training and the remaining $3,466$ images for validation. Performance is measured with the average detection error and the failure rate of each facial point. They indicate the accuracy and reliability of an algorithm. The detection error is measured as

$$err = \sqrt{(x - x')^2 + (y - y')^2} \,/\, l \,, \qquad (2)$$

where (x, y) and (x', y') are the ground truth and the detected position, and l is the width of the bounding box returned by our face detector. If an error is larger than 5%, it is counted as failure. Note that the bi-ocular distance is more commonly used as the detection error normalizer, but it has problem on faces with large pose variations, since bi-ocular distance of near-profile faces is much shorter than that of frontal faces. Its drawback is also noticed in [28]. So we use the width of the face bounding box instead to validate our algorithm in Section 5.1 and will switch to bi-ocular distance for fair comparison in Section 5.2. Some exemplar images from our validation set and our detection results are shown in Figure 8a.

Network structure. We use network F1 as an example to investigate how network depth, absolute value rectification, and the weight sharing scheme influence the performance. Six different network structures summarized in Table 1 are studied and their performance is compared in Figure 4. S0, S3-S5 all have absolute value rectification and lo-

[2] In LFW each person has many similar images. To save annotation effort, we only select one image per person. After removing a few images where our face detector failed, we get 5590 images from LFW.

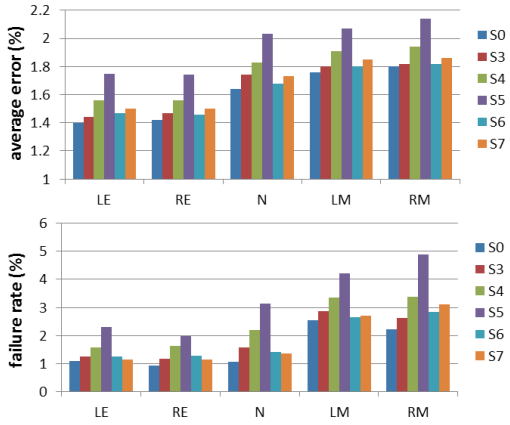


Figure 4: Average detection errors and failure rates of convolutional network F1 with different structures.

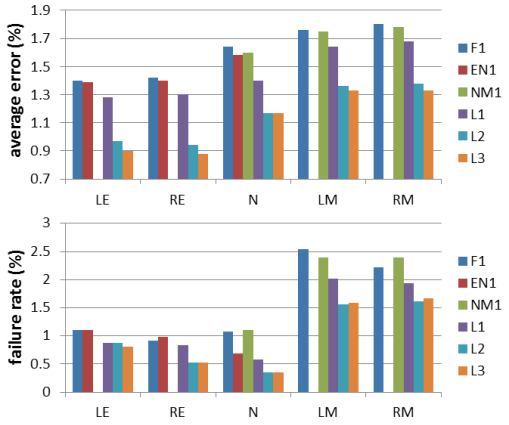


Figure 5: Average detection errors and failure rates of the three networks, F1, EN1, and NM1, at level 1, their combination denoted as L1, the first two cascaded levels (L2), and the three cascaded levels (L3). LE is not predicted by NM1, so the corresponding result is missed. It also applies to other keypoints and networks.

cally shared weights, but with different depths. The results show that the performance can be significantly improved by including more layers. Since the input face region is in size of 39×39 and the network keeps downsampling images as going up to the top layers, S0 has reached the maximum number of possible layers. S6 and S7 have the same layers as S0. But S6 does not adopt absolute value rectification and S7 globally shares weights in all its convolutional layers. It is observed that both absolute value rectification and locally sharing weights are effective in facial point detection. We also find that locally sharing weights in higher layers is more important, while only locally sharing weights in lower layers deteriorates the performance, which coincides with our conjecture that high-level features are less likely to be shared than low-level features.

Multi-level prediction. Detection errors can be effectively reduced by multi-level cascaded prediction and fusion of multiple predictions at the same level. Figure 5 compares the performance of the three networks at level 1, and cascaded prediction with different numbers of levels. Detection errors and failures are greatly reduced at the second level. At the third level, the average detection errors are slightly reduced while the failure rates almost remain the same.

5.2. Comparison with other methods

We compare with state-of-the-art methods and latest commercial software on two public datasets, BioID [15] and LFPW [4]. BioID contains face images collected in lab conditions while LFPW contains face images from the web. On both test sets, we use the model trained on the dataset described in Section 5.1. To be consistent with most previous works, we used the bi-ocular distance to normalize detection errors and redefine the failure rate as the proportion of cases whose normalized errors are larger than 10%. The results are summarized in Figure 6.

BioID has $1,521$ images of 23 subjects. All the faces are frontal, with moderate variations on illumination and expression. We compare with Component based Discriminative Search [20], Boosted Regression with Markov Networks [26], and two latest commercial software, Luxand Face SDK [1] and Microsoft Research Face SDK [2]. Since Microsoft Research face SDK does not detect eye centers and nose tip, we only compare mouth corners with it. Our method reduces the detection errors significantly, and our failure rate approaches to zero. Figure 8b shows some of our detection results on BioID.

LFPW contains $1,432$ face images from the web. It is divided into $1,132$ training images and 300 test images. This dataset is intended to test facial point detection in unconstrained conditions, and faces show large variations on pose, illumination, and expression, and may contain occlusions. It shares only image URLs and some image links are no longer valid. We only download 781 training images and 249 test images. Since there is no overlap between our training/validation datasets and LFPW, we use both LFPW training and test images as our test images and compare with the four methods mentioned above. Our method again shows superior performance on faces in the wild conditions. Figure 8c shows some of our detection results on LFPW.

Belhumeur *et al*. [4] and Cao *et al*. [5] reported results on LFPW test images, and the latter defined the current state-of-the-art on these images. The result in [4] is on all the 300 LFPW test images. The result in [5] is on 249 of 300 LFPW test images due to the disappearing of image URLs. We also evaluate our algorithm on the 249 LFPW test images in order to compare with the two best methods on this dataset. Figure 7 shows the comparison results on average

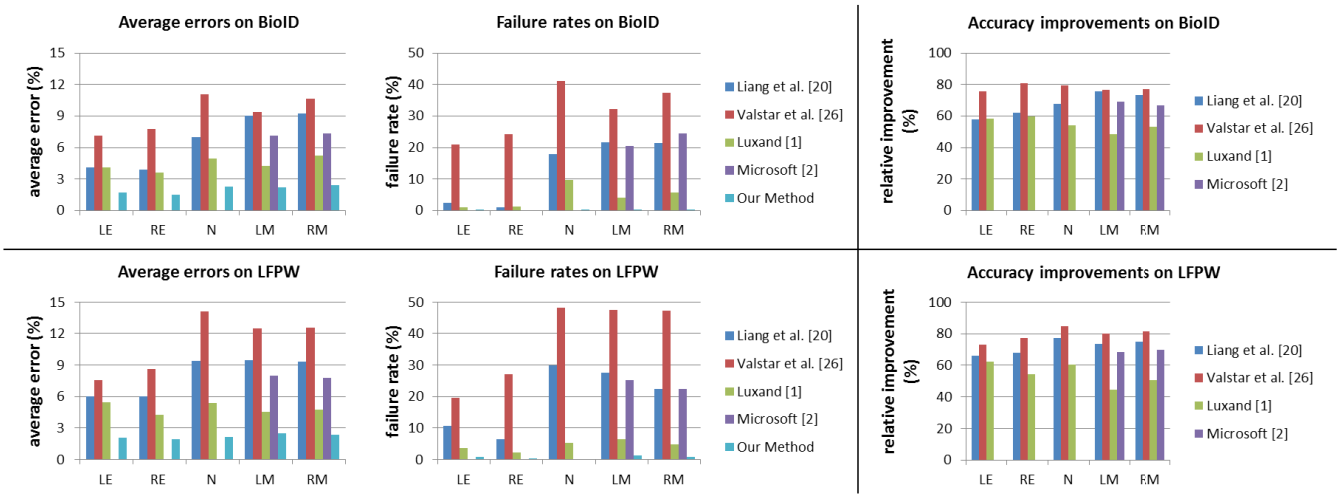


Figure 6: Comparison on BioID and LFPW. Since our failure rate approaches to zero on BioID, it may not be observable in the figure. Relative improvement $= \frac{\text{reduced average error}}{\text{average error of the method in comparison}}$. We achieved over 50% accuracy improvement on both datasets.

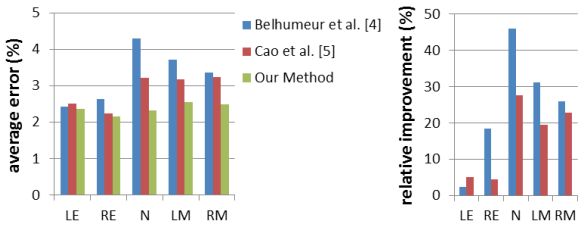


Figure 7: Compare with Belhumeur *et al*. [4] and Cao *et al*. [5] on LFPW test images.

errors and our relative accuracy improvements over the two methods. [4, 5] are very competitive methods, which perform significantly better than their contemporaries. Still, we improved their results with a large margin. More than 20% relative accuracy improvement is achieved for nose tip and two mouth corners. The C++ implementation of our algorithm takes 0.12 second to process one image on a 3.30GHz CPU [3]. The system can be easily parallelized since convolutional networks at each level are independent.

6. Conclusion

We proposed an effective convolutional network cascade for facial point detection. Deep convolutional networks at the first level provide highly robust initial estimations, while shallower convolutional networks at the following two levels finely tune the initial prediction to achieve high accuracy. By exploring a few key features of the network structure, we achieve high performance convolutional networks with a relatively small scale. Our method significantly improves the prediction accuracy of state-of-the-art methods and latest commercial software.

[3]The time preparing the input for our algorithm (face detection and image resizing) is excluded.

References

[1] http://www.luxand.com/facesdk/. 6

[2] http://research.microsoft.com/en-us/projects/facesdk/. 6

[3] B. Amberg and T. Vetter. Optimal landmark detection using shape models and branch and bound. In *Proc. ICCV*, 2011. 1, 2

[4] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar. Localizing parts of faces using a consensus of exemplars. In *Proc. CVPR*, 2011. 1, 2, 6, 7

[5] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. In *Proc. CVPR*, 2012. 1, 2, 6, 7

[6] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Proc. CVPR*, 2012. 2

[7] A. Coates, A. Y. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. *Journal of Machine Learning Research*, 2011. 2

[8] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *Proc. ECCV*, 1998. 1

[9] M. Dantone, J. Gall, G. Fanelli, and L. J. V. Gool. Real-time facial feature detection using conditional regression forests. In *Proc. CVPR*, 2012. 1, 2

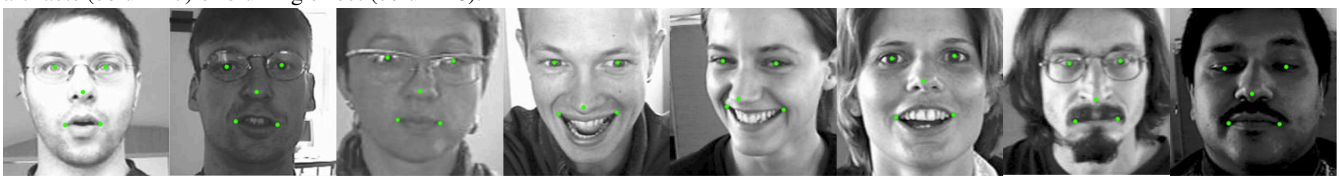
[10] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *PAMI*, 2013. 2

[11] L. Gu and T. Kanade. A generative shape regularization model for robust face alignment. In *Proc. ECCV*, 2008. 1

[12] G. B. Huang, H. Lee, and E. Learned-Miller. Learning hierarchical representations for face verification with convolutional deep belief networks. In *Proc. CVPR*, 2012. 4

[13] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, 2007. 5

(a) Validation images. Faces vary greatly on poses (column 1, 2, 3) and expressions (column 6), and may have occlusions (column 4, 5), artifacts (column 7) or blurring effect (column 8).



(b) BioID.

(c) LFPW. Faces vary greatly on poses (column 1, 2), expressions (column 5, 6) and illuminations (column 8), and may have occlusions (column 3, 4) or blurring effect (column 7).

Figure 8: Our results on validation images, BioID, and LFPW.

[14] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Proc. ICCV*, 2009. 2, 3

[15] O. Jesorsky, K. J. Kirchberg, and R. Frischholz. Robust face detection using the hausdorff distance. In *Proc. AVBPA*, 2001. 6

[16] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierarchies for visual recognition. In *Proc. NIPS*, 2010. 2

[17] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, 2012. 2

[18] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng. Building high-level features using large scale unsupervised learning. In *Proc. ICML*, 2012. 2

[19] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and M. K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998. 5

[20] L. Liang, R. Xiao, F. Wen, and J. Sun. Face alignment via a component-based discriminative search. In *Proc. ECCV*, 2008. 1, 2, 6

[21] X. Liu. Generic face alignment using boosted appearance model. In *Proc. CVPR*, 2007. 1

[22] P. Luo, X. Wang, and X. Tang. Hierarchical face parsing via deep learning. In *Proc. CVPR*, 2012. 2

[23] S. Milborrow and F. Nicolls. Locating facial features with an extended active shape model. In *Proc. ECCV*, 2008. 1

[24] M. Osadchy, Y. L. Cun, and M. L. Miller. Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research*, 2007. 2

[25] P. Sauer, T. Cootes, and C. Taylor. Accurate regression procedures for active appearance models. In *Proc. BMVC*, 2011. 1, 2

[26] M. Valstar, B. Martinez, X. Binefa, and M. Pantic. Facial point detection using boosted regression and graph models. In *Proc. CVPR*, 2010. 1, 2, 6

[27] H. Wu, X. Liu, and G. Doretto. Face alignment via boosted ranking model. In *Proc. CVPR*, 2008. 1

[28] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Proc. CVPR*, 2012. 1, 2, 5