Joint Training of Cascaded CNN for Face Detection

Hongwei Qin[1,2] Junjie Yan[3,4] Xiu Li[1,2] Xiaolin Hu[3]
[1]Grad. School at Shenzhen, Tsinghua University [2]Dept. of Automation, Tsinghua University
[3]Dept. of Computer Science and Technology, Tsinghua University [4]SenseTime
{qhw12@mails., li.xiu@sz., xlhu@}tsinghua.edu.cn yanjunjie@outlook.com

Abstract

Cascade has been widely used in face detection, where classifier with low computation cost can be firstly used to shrink most of the background while keeping the recall. The cascade in detection is popularized by seminal Viola-Jones framework and then widely used in other pipelines, such as DPM and CNN. However, to our best knowledge, most of the previous detection methods use cascade in a greedy manner, where previous stages in cascade are fixed when training a new stage. So optimizations of different CNNs are isolated. In this paper, we propose joint training to achieve end-to-end optimization for CNN cascade. We show that the back propagation algorithm used in training CNN can be naturally used in training CNN cascade. We present how jointly training can be conducted on naive CNN cascade and more sophisticated region proposal network (RPN) and fast R-CNN. Experiments on face detection benchmarks verify the advantages of the joint training.

1. Introduction

Face detection plays an important role in face based image analysis and is one of the fundamental problems in computer vision. The performances of various face based applications, from face identification and verification to face clustering, tagging and retrieval, rely on accurate and efficient face detection. Recent works in face detection focus on faces in uncontrolled setting, which is challenging due to the variations in subject level (e.g., a face can have many different poses), category level (e.g., adult and baby) and image level (e.g., illumination and cluttered background).

Given a novel image I, the face detector is expected to return a bounding box configuration $B = (b_i, c_i)_{\{N\}}$, where the b_i and c_i specify the localization and confidence of a face. The number of detected faces N always vary in different images. Considering that the b_i can possibly appear in any scale and position, the face detection problem has a output space of size $\frac{(w*h)^2}{2}$, where w and h denote width and height respectively. Considering that it can be $\frac{(500*350)^2}{2} \approx 10^{10}$ for a typical 500×350 image, it is actually impossible to evaluate them all at a acceptable cost. Actually, only a few of them correspond to faces and most of the configurations in the output space belongs to the background.

The previous face detection research can be seen as a history of more efficiently sampling the output space to a solvable scale and more effectively evaluating per configuration. One natural idea to achieve this is using cascade, where classifier with low computation cost can be firstly used to shrink background while keeping the faces. The pioneering work [27] popularized this, which combined classifiers in different stages, to allow background regions quickly discarded while spending more computation on promising face-like regions. The cascade made efficient detection possible and was widely used in subsequent works. For example, two other detection pipelines DPM [6] and CNN [16] can both use cascade for acceleration.

Despite the efficiency in testing, the cascade based detectors are always trained greedily. In a typical training procedure, when training a new stage in the cascade, previous stages are fixed. The relationship of different stages lies in that each stage is trained with the *hard* training samples which pass through previous stages. It makes the greedily trained cascade not end-to-end optimal with respect to the final detection score. It leads to performance drop when compared with non-cascade methods. For example, the cascade version of DPM [6] does not as accurate as the original version [7].

In this paper, we show that in CNN based cascade detection, other than enjoying the advantages in efficiency as traditional cascade, different stages in the cascade can be jointly trained to achieve better performance. We show that the back propagation algorithm used in training CNN can be naturally used in training CNN cascade. Joint training can be conducted on naive CNN cascade and more sophisticated cascade such as region proposal network (RPN) and fast R-CNN. We show that the jointly trained cascade CNN as well as the jointly trained RPN and fast R-CNN can achieve lead-

ing performance on face detection.

The rest of the paper is organized as follows. Section 2 reviews the related work. Analysis of jointly training is presented in section 3. Then we present how to jointly train naive CNN cascade in section 4 and how to jointly train RPN and Fast RCNN in section 5. Section 6 shows the experimental results and analysis and section 7 concludes the paper.

2. Related Work

Numerous works have been proposed for face detection and some of them have been delivered to real applications. Similar to many other computer vision tasks, leading algorithms in face detection are based on convolutional neural network in the 1990s, then based on hand-craft feature and model, and recently based on convolutional neural network again. In this part, we briefly review the three kinds of methods and refer more detailed survey to [33, 37, 35].

2.1. Early CNN based methods

Face detection, as well as MNIST OCR recognition, are two tasks where CNN based approach achieve success in 1990s. In [26], CNN is used in a sliding window manner to traverse different locations and scales and classify faces from the background. In [22], CNN is used for frontal face detection and shows quite good performance. In [23], CNNs trained on faces from different poses are used for rotation invariant face detection. These methods are quite similar to modern CNN methods and get relatively good performance on easy datasets.

2.2. Hand-craft feature based methods

In [27], Viola and Jones proposed to use Haar feature, Adaboost based learning and cascade based inference for face detection. It shows advantage in speed when compared with methods (e.g., [22, 24, 19]) at the same period and quickly became very popular. Many subsequent works further improve performance through new local features [38, 31], new boosting algorithms [36, 11] and new cascade structures [1, 28]. The single model in Viola-Jones framework cannot handle faces from different poses, and in [17, 14, 11] the authors proposed efficient cascade structures to use multiple models for pose-invariant face detection. [2] uses additional landmarks annotations for better detection performance.

Besides Viola-Jones framework, methods based on structural models progressively achieve better performance and becomes more and more efficient, on challenging benchmarks such as AFW [39] and FDDB [13]. The seminal work deformable part model (DPM) [7] use the deformable parts on top of HOG feature to represent objects. [39, 30, 18, 29, 8] use supervised parts, more pose partition, better training or more efficient inference to achieve better performance.

2.3. Modern CNN based methods

In recent two years, CNN based methods show advantages in face detection. [32, 20] use boosting and DPM on top of CNN features. [5] fine-tune CNN model trained on 1000-way ImageNet classification task for face background classification task. [34] uses fully convolutional networks (FCN) to generate heat map of facial parts and then use the heat map to generate face proposals. [12] uses a unified end-to-end FCN framework to directly predict bounding boxes and object class confidences. These methods, however, are relatively slow even on a high-end GPU. In [16], six CNNs (three stages) are cascaded to efficiently reject backgrounds.

3. Cascaded Networks

Algorithms using cascaded stages are widely used in detection tasks. The advantage of cascaded stages lies in that they can handle unbalanced distribution of negative and positive samples. In the early stages, week classifiers can reject most false negatives. In the later stages, stronger classifiers can save computation with less proposals.

With the development of deep CNNs, multi-stage CNNs are getting popular. State-of-the-art object detection algorithms adopt multi-stage mechanisms. The first stage is a network for region proposal generation. The following one or more stages are networks for detection. Cascaded CNNs [16] and faster R-CNN [21] are such mechanisms.

However, previous methods are not jointly trained. They use greedy algorithms to optimize. Different from boosting methods, deep CNNs can naturally be jointly optimized. Recent CNNs are usually deep neural networks using back-propagation for optimization. So layers of different networks can be jointly optimized to share computation and information. For object detection tasks, we can design a single network that includes both region proposal generation and detection (maybe multi-stage) and optimize it jointly with back-propagation.

3.1. Cascaded CNNs

The cascaded CNN for face detection in [16] contains three stages. In each stage, they use one detection network and one calibration network. There are totally six CNNs. In practice, this makes the training process quite complicated. We have to carefully prepare the training samples for all the stages and optimize the networks one by one.

One natural question is how about we jointly train all the stages in one network?

Firstly, detection network and calibration network can share a multi-loss network used for both detection and

bounding-box regression. Multi-loss optimization has been proved effective in general objection detection [9, 21].

Secondly, if multi-resolution is used during training the later stages, as the authors did in [16], the network of the later stage contains the network of the previous one. So theoretically, the convolution layers can be shared by three stages. Meanwhile, shared convolutional layers results in smaller model size. In the joint training network, the model size is approximately the same as the final stage in separate cascaded CNNs.

Thirdly, in cascaded CNNs, the separate first stage used for generating proposals is only optimized by itself. In the joint network, it is jointly optimized by larger scale branches. In this way, each branch benefits from other branches. Together, the joint network is expected to achieve end-to-end optimization.

3.2. RPN + fast R-CNN

In faster R-CNN, the authors use one CNN called Region Proposal Network (RPN) for generating proposals, the other CNN called fast R-CNN [9] for detection.

In order to share convolutional layers between region proposal network and detection network, the typical training of RPN and fast R-CNN adopts four separate stages and uses alternating optimization. In the first step, RPN is initialized with an ImageNet pre-trained model and fine-tuned for region proposal. In the second step, fast R-CNN is initialized with the same pre-trained model and fine-tuned for detection. In the third step, RPN is initialized with fast R-CNN model from the second step and fine-tuned for region proposal. At this point, RPN and fast R-CNN share convolutional layers. Finally, fast R-CNN is fine-tuned from the second step fast R-CNN with proposals generated by the third step RPN.

The core idea is to let region proposal and later detection network share convolutional layers. However, in the final models, the convolutional layers are dominated by the second step fast R-CNN. The loss of RPN in the third step would not back-propagate to the convolutional layers.

RPN and the first stage FCN of cascaded CNNs are highly similar. They both use fully convolutional neural network to generate proposals. The input image can be of arbitrary size. The convolution computations are shared among proposals. They both use bounding-box (anchor) regression to refine the proposals. They both can handle various scales of proposals. The main difference lies in that FCN uses image pyramids to handle various scales, while RPN uses pre-set anchor scales to do that. Naturally FCN can better handle more scales than RPN, while RPN can save computation with only one input scale.

So the joint training of cascaded CNNs can apply to RPN and fast R-CNN.

4. Joint Training of Cascaded CNN

We design a joint training architecture to train the network once for all. We call this architecture FaceCraft.

Fig. 1 demonstrates this joint training architecture. During training, the network takes an image of size 48×48 as input, and outputs one joint loss of three branches. The three branches are called *x12*, *x24*, *x48* respectively, corresponding to the input size of each network. We use *ReLU* for non-linear layers and *drop-out* before classification or regression layer.

It is optimized through back-propagation. Compared to separate networks, the joint network also use threshold control layers to decide which proposals from up branches contribute to the loss of the down branches.

4.1. Training architecture

Branch *x12*: fully convolutional proposal network The proposal generation network is a fully convolutional network that has two sibling output layers. Input data is averagely pooled to 12×12. The final convolutional layer before output layer is of size $1 \times 16 \times 1 \times 1$. For the two output layers, one outputs a probability distribution (per feature map point) $p = (p_0, p_1)$, over face v.s. non-face. The other outputs bounding-box regression offsets, $t^k = (t^k_x, t^k_y, t^k_w, t^k_h)$, for each of the predicted face proposal.

Optimization We use a multi-task loss of classification and bounding-box regression to jointly optimize this branch. We use softmax loss for classification and smooth $L1$ loss defined in [9] for bounding-box regression:

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v), \quad (1)$$

where $L_{\text{cls}}(p, u) = -\log p_u$ is log loss for true class u.

We set $\lambda = 1$ in our experiment, this is appropriate for all three separate CNN networks.

For the regression offsets, we set the 4 coordinates defined in [10]:

$$t^*_{\text{x}} = (x^* - x_{\text{p}})/w_{\text{p}} \quad (2)$$
$$t^*_{\text{y}} = (y^* - y_{\text{p}})/h_{\text{p}} \quad (3)$$
$$t^*_{\text{w}} = \log(w^*/w_{\text{p}}) \quad (4)$$
$$t^*_{\text{h}} = \log(h^*/h_{\text{p}}), \quad (5)$$

where x and y denote the two coordinates of the box center, w and h denote box width and height respectively . The variables x_{p}, and x^* are for the proposal box and ground-truth box respectively. In this way, we can optimize the regression targets and regress the bounding-box from a proposal box to a nearby ground-truth box.

During training, the regression offsets are normalized to have zero mean and unit variance. This optimization method also applies to the other two branches.

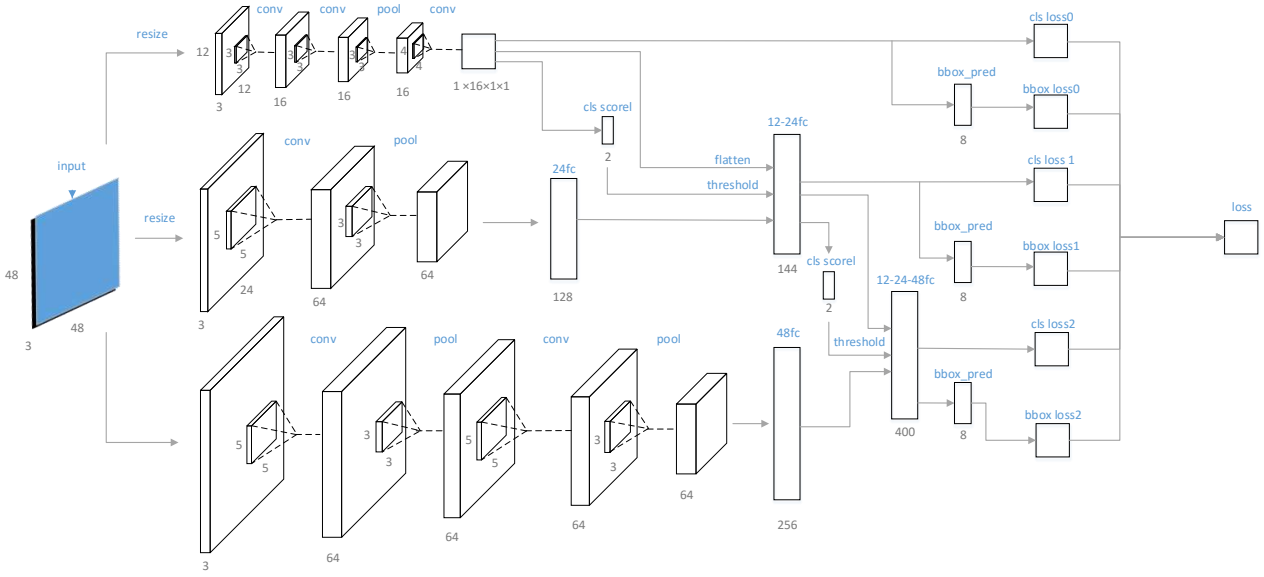


Figure 1. Joint training architecture. During training, the network takes an image of size 48×48 as input, and outputs one joint loss of three branches. The network is optimized through back-propagation. Compared to separate networks, the joint network also use threshold control layers, to decide which proposals from up branches contribute to the loss of the down branches.

Branch *x12-x24* hard negative sample mining *x12-x24* first averagely pools the input data to size 24×24, then outputs a 128 dimensional fc layer called *24fc*. *24fc* is concatenated with 1×16 fc layer flattened from $1 \times 16 \times 1 \times 1$ of *x12*. We call the new fc layer *12-24fc*.

We choose a classification score threshold for the score threshold layer. Only the passed proposals contribute to the loss of final layers. In our experiments, 0.1 is an appropriate threshold. This threshold is similar to that in separate cascaded networks.

x12-x24 also outputs classification loss and bounding-box regression loss.

Branch *x12-x24-x48*: harder negative mining *x12-x24-x48* outputs a 256 dimensional fc layer called *48fc*. *48fc* is concatenated with *12-24fc*.

As in *x12-x24*, we choose a classification score threshold for the score threshold layer. Only the passed proposals contribute to the loss of final layers. In our experiments, 0.003 is an appropriate threshold. This threshold is similar to that in separate cascaded networks.

x12-x24-x48 also outputs classification loss and bounding-box regression loss.

Joint loss Each branch has a face v.s. non-face classification loss and a bounding-box regression loss. Adding them with loss weights, we get the joint loss function:

$$L_{\text{joint}} = \lambda_1 L_{\text{x12}} + \lambda_2 L_{\text{x24}} + \lambda_3 L_{\text{x48}}, \qquad (6)$$

where L_{x12}, L_{x24} and L_{x48} denote different losses of three branches. The loss of each branch is calculated by Equation 1. λ_1, λ_2 and λ_3 are loss weights of the three branches.

4.2. Implementation details

Training data To prepare training data, we first use sliding windows on each training image to get face candidates. The positive samples are chosen from the candidates that have intersection over union (IoU) overlap with any ground-truth bounding box of larger than 0.8. The negative samples are sampled from the face candidates that have a maximum IoU with ground-truth in the interval $[0, 0.5)$. The samples are cropped and resized to network input size. To apply data augmentation, each sample is horizontally flipped. The ultimate ratio of positive samples of the whole training data is about 5%. The input patches are mean removed with mean image from ImageNet [3]. No other pre-processing is used.

Training procedure Each training image is first built into image pyramids with interval of 5. The smallest pyramid is $1/2^5$ of the original image. We prepare face proposals by sliding windows with stride 8 over training images. Positive samples are chosen from face proposals whose maximum IoU with ground-truth is larger than 0.8. Negative samples are chosen from the proposals that have a maximum IoU with ground-truth in the interval $[0, 0.5)$. For the sample ratio, we keep a very low positive sample ratio during stage one. Because this can decrease false positives , which also accelerates the following negative mining stages. In our ex-

periments, setting the ratio of positive samples as 5% is appropriate. The *x12* branch threshold is set as 0.1, *x12-x24* branch threshold is set as 0.003. They are set empirically. Within appropriate threshold range, the training procedure is quite robust. The principle is to make the threshold as high as possible while keeping the recall, so as to reject as many proposals as possible in the earlier stages. Alternatively, we can fix the proposal number, which is exactly what we did in the joint training of RPN and fast R-CNN. During forward, only face proposals that have *x12* branch scores higher than 0.1 contribute to *x12-x24* branch. Only face proposals that have *x12-x24* branch scores higher score than 0.003 contribute to *x12-x24-x48* branch.

We decrease the positive sample thresholds when training the three stages. So in the later stages, we can train the networks with *harder* samples. This in turn results in stronger models for face v.s. non-face classification.

To make it converge eaisily, we train seperate networks and initialize the joint network with trained weights.

SGD hyper-parameters. We set global learning rate 0.001. After a number of iterations, we lower the learning rate to 0.0001 to train more iterations. The specific iteration number is related to the number of training samples. Generally, 5 to 10 epochs would be appropriate. Following standard practice, we use a momentum term with weight 0.9 and weight decay factor of 0.0005.

4.3. Testing pipeline

The testing pipeline contains three separate CNNs. Given an input test image, the fully convolution network outputs a feature map response. Each point of the feature map gives a face v.s. non-face classification score and bounding box regression targets (1×4). The regression targets are used for bounding-box refinement. Each point is corresponding to a bounding-box, whose up-left vertex is the point. After the first stage, we keep the boxes whose scores are higher than the pre-set threshold. Usually Non-maximum suppression (NMS) is applied to reject the highly overlapped boxes. However, NMS is quite time consuming. We use a novel algorithm to reject the highly overlapped boxes. Inspired by max pooling, we apply *max pooling* on the output feature map. Different from max pooling, the resolution of the feature map stays unchanged. In detail, we choose a max pooling kernel k. In the $k \times k$ area, only the point with the highest score is kept, while the others are set to zero. The stride of the stage one network is 2. So, when $k = 2$, the result is comparative to NMS with threshold set as 0.5.

The remaining boxes are fed to the second stage. We crop each box region from the original image and resize them to 24×24. For each region, this network outputs a score and corresponding bounding-box regression targets. As in the first stage, we reject the boxes whose score is lower than the pre-set threshold. Then we use the regression targets to refine the boxes.

After the second stage, in average only dozens of boxes are remaining. We apply similar procedure as the second stage. The box regions are cropped and resized to 48×48 for input. After score threshold operation, the passed boxes are refined by bounding box regression targets which is the output of the third stage.

Finally NMS is applied to all the remaining boxes to get the final detection results. In practice, the test image is first built into image pyramids to handle different face scales.

5. Joint Training of RPN and Fast R-CNN

5.1. Training architecture

Considering the disadvantages of RPN + Fast R-CNN training procedure, we design an alternative joint training architecture for RPN and fast R-CNN. The architecture is shown in Fig. 2. In our architecture, we use only one joint network. In this network, RPN and fast R-CNN share convolutional layers. The output of the last shared convolutional layer is fed into two sibling branches. One region proposal branch for generating candidate proposals with scores. The other is Region of Interest (*RoI*) pooling branch for detection with final scores.

5.2. Training process

The training is a two-step procedure ($2\times$ faster than the original four-step one). We train RPN as in separate training first, then fine-tune the joint network from RPN model. The proposals used for fast R-CNN branch are generated by doing RPN test on the training images. During training the joint network, the inputs consist of original RPN inputs plus proposals generated by trained RPN model. The RPN generated proposals serve as the input *RoI*s of fast R-CNN in the joint network. The loss fusion is similar with the joint training of cascaded CNNs in section 4.1.

Except for cascade of RPN and fast R-CNN, there are more cascaded CNNs out there. Joint training maybe one possible solution for end-to-end optimization and convolutional layer sharing.

6. Experiments

We carry out experiments on face detection dataset to evaluate our joint training pipeline.

6.1. Datasets

In training joint cascaded CNNs, we use Annotated Facial Landmarks in the Wild (AFLW) [15] and our dataset called $3R$. $3R$ contains about 26000 images that have faces and 27000 images that have no faces. $3R$ is collected from

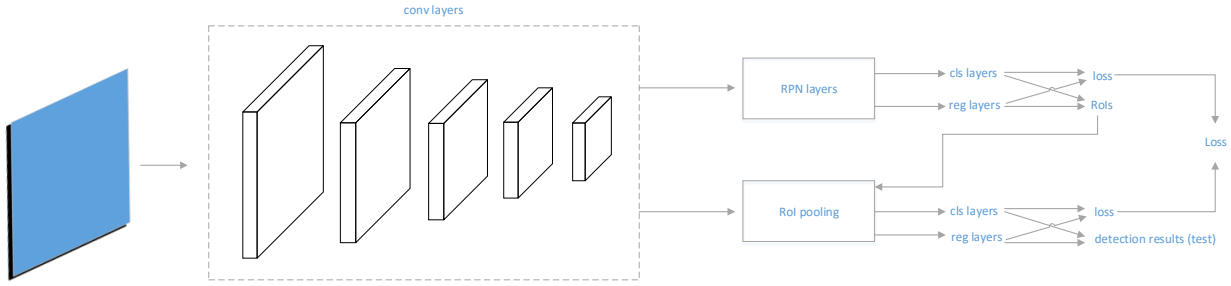
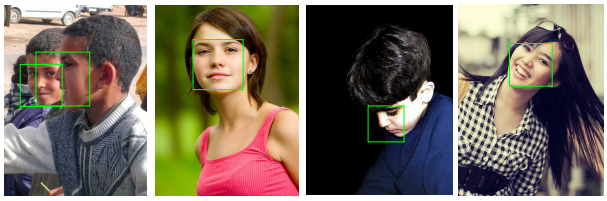


Figure 2. In our architecture, we use only one joint network. In this network, RPN and fast R-CNN share convolutional layers. The output of the last shared convolutional layer is fed into two sibling branches. One region proposal branch for generating candidate proposals with scores. The other is *RoI* pooling branch for detection with final scores.



(a) lateral (b) frontal (c) downward (d) upward

Figure 3. Face annotation examples.

online social network, the image on which is a reflection of the real world images in everyday life. To add negative samples, we also use images in PASCAL VOC2012 [4] that do not contain persons as background image. In total, the dataset contain 47211 images with 82987 faces and about 32000 background images. To avoid confusing circumstances when it is difficult to judge a patch is ground-truth or not, we add *ignore region*s in our training images. An *ignore region* is defined as a region where we do not sample negative samples.

To avoid annotation confusion, we do not annotate using face rectangles. Instead, each face is annotated by 21 facial landmarks. The landmarks are slightly different from those of AFLW official annotations, of which a face may be annotated with less than 21 landmarks. We design a transformation algorithm from facial landmarks to face rectangles. The face rectangles are square annotations. Face examples are shown in Fig. 3. We can see that nose is always in the center of the square annotations.

6.2. AFW results

We evaluate FaceCraft on Annotated Faces in the Wild (AFW) [39]. AFW contains 205 images collected from Flickr. The images contain cluttered backgrounds and various face viewpoints and appearances.

In ground-truth annotation, one specific problem of face detection different from general object detection is how to decide the face bounding box when a face is not frontal. Different rules in face annotations result in various ground-truth. So detectors trained with different training data may get mismatched results on test dataset annotated following different rules. This problem has been pointed out before [18, 16]. In our test results, this is also true. Examples of detection results are shown in Fig. 4. In our test results, non-frontal face bounding-box centred on the nose, which is consistent with our training ground-truth shown in Fig. 3. While in AFW ground-truth, nose is on the bounding-box edge.

In previous work [18, 16], the authors use refined detection results or manual evaluation to evaluate on AFW. We check all of our detection results and found that setting the evaluation IoU to 0.3 for all the methods can fairly evaluate the non-frontal faces with almost no impact on frontal faces. In this way, no post-processing or human interference is needed. Practically, 0.3 IoU is enough for following applications like face alignment.

When we judge a detector, the accuracy of bounding-box should be one of the indicators. For now, there is no such evaluation rule on AFW. One reason is that researches use various training data and various annotations. One possible solution maybe that we use a uniform transformation from facial landmarks to face rectangle annotations. Because facial landmarks can be accurately annotated.

The comparison of Precision-Recall curves generated by different methods is shown in Fig. 6. As we can see, the recall of our face detector is higher than all previous results, approaching 99.75%. The Average Precision (AP) is 98.22%, which is comparable with state-of-the-art methods.

6.3. FDDB results

Face Detection Data Set and Benchmark (FDDB) [13] contains 2845 images with 5171 face annotations.

We use the evaluation toolbox provided by [18], which also reports recall at 1000 false positives. This dataset use two kinds of evaluation method, discontinuous score and continuous score [13]. As pointed in section 6.2, conti-

Figure 4. Qualitative results of FaceCraft on AFW.

Figure 5. Qualitative results of FaceCraft on FDDB.

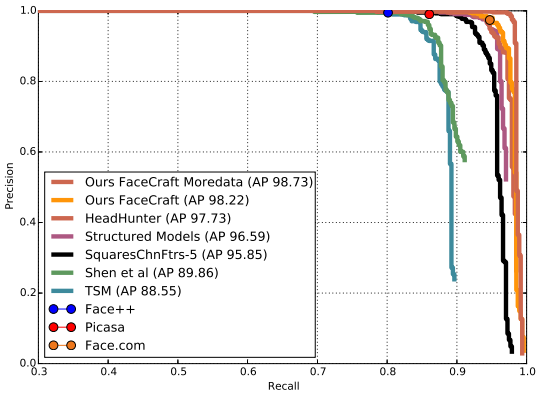


Figure 6. Precision-Recall Comparisons with state-of-the-art methods on AFW. The methods are HeadHunter [18], Structured Models [30], SquaresChnFtrs-5 [18], Shen *et al.* [25], TSM [39], Face.com, Face++ and Google Picasa.

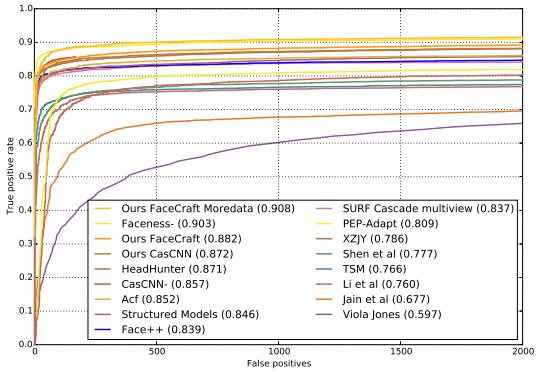


Figure 7. Comparison with state-of-the-art methods on FDDB.

nous score is largely influenced by annotations of training dataset. Previous works use many tricks to refine the detected face box to get a better score. To reduce comparison confusion, we only report discontinuous score.

The curve is shown in Fig. 7. As we can see, the joint training result get a recall of 88.2% (1000 false positives), which is comparative with the state-of-the-art. This is better than Cascaded CNN result (85.7%) reported in [16].

To make a fair comparison, we also show the result with the same networks but trained separately. The separately trained model get a recall of 87.2%, which is lower than Jointly trained model.

6.4. How larger dataset benefits?

In Faceness [34] the authors used more training data. To evaluate our method on larger training dataset, we conduct experiments with enlarged dataset $3R+$. In total, we use 108000 images with annotated faces. The experiments prove that our architecture can benefit from enlarged train-

ing data. Results are shown in Fig. 6 and Fig. 7. AP on AFW is 98.73%, recall on FDDB is 90.8% (1000 false positives).

6.5. Detection efficiency

CNN based methods have always been accused of its runtime efficiency. Recent CNN algorithms are getting faster on high-end GPUs. However, in most practical applications, especially mobile applications, they are not fast enough.

In our testing pipeline, the later two stages are more complicated, while the most time consuming stage is the first stage. By rejecting most of the face proposals in the first stage, we make the following two networks very efficient. The later two stages occupy about half of the whole computation.

Response map pooling In the first stage, after using score threshold, there are still nearly 1000 proposals in average. NMS would be time consuming, so we use *max pooling* on the final feature map. This is faster than NMS while achieving similar result.

Image patch resizing In the later two stages, we need to resize the image patches to the network input size. Actually when generating image pyramids, we've already resized the images to $1/2^k$ of the original size. To speed up the computation, for a passed face proposal after stage one, we can find the corresponding twice larger patch from the upper scale image pyramid. In this way, we can save the patch resizing time.

Our fast version method achieves 10 FPS on a single CPU core while keeping 87.3% (merely decreased compared to previous 88.2%) recall on FDDB. We test on VGA images, and detect multi-scale faces as small as 24×24. For specific circumstances, we can vary threshold and image pyramids number to accelerate. As comparison, the fast version of Faceness [34] uses outside methods to generate proposals, and runs 20 FPS on Titan Black GPU for VGA images, at the expense of recall rate decreased to 87%. Cascade CNNs [16] runs 14 FPS on CPU at the expense of recall rate, and it only scans for 80×80 faces.

6.6. Experiments of jointly trained faster R-CNN

For the convolutional layers, we use a network that is modified from ZF-net, in which we chop off the LRN layers. All settings in training separate and joint networks are the same, *e.g.*, loss weights, NMS thresholds, proposal numbers and learning rates. We train RPN first and fine-tune the joint network from RPN model. The proposals used for fast R-CNN branch are generated by doing RPN testing on the training images. During training the joint network, the joint inputs consist of original RPN inputs and proposals generated by trained RPN model. The joint network converges easily with improved performance. The experiments are conducted on $3R+$.

Table 1. Comparison of training methods of RPN + F-RCNN

Benchmark	Separate	Joint
AFW	97.0%	98.7%
FDDB	89.7%	91.2%

As shown in Table. 1, with our presented RPN + F-RCNN (fast R-CNN) joint training pipeline, the AP (average precision) on AFW is 98.7%, compared to the baseline result 97.0% trained with 4-stage training method proposed in [21]. On FDDB, the recall (1000 false positives) is 91.2% *v.s.* 89.7%. For the F-RCNN branch, the final joint training loss decreases 64% compared to separate training. In joint RPN + F-RCNN, the detection results mostly have much higher confidence scores than separate training results, which have lower confidence scores because of F-RCNN domination in convolution layers.

6.7. Disscussion

Except for the use of jointly trained cascaded CNNs for face detection, jointly trained RPN and fast R-CNN is also a promising method for fast and accurate face detection. RPN is very fast for generating good proposals with large pre-trained models in general object detection, while it is not fast enough for face detection. For face detection, we can design smaller RPN and train from scratch. The advantage is that the use of multi-scale anchors can replace image pyramids used in previous methods. However if trained with the faster R-CNN four-step procedure, the RPN convolutional layers would be dominated by fast R-CNN. If RPN is jointly trained with fast R-CNN, the whole network can get better performance. As the whole computation won't add too much compared with RPN only, fast face detection can be very promising.

7. Conclusion

In this paper, we have presented joint training as a novel way of training cascaded CNNs. By joint training, CNN cascade can achieve end-to-end optimization. We show that the back propagation algorithm used in training CNN can be naturally used in training CNN cascade. By jointly optimizing cascaded stages, the whole network get improved performance with smaller models for sharing convolutions. We evaluate joint training on face detection datasets. Our results achieve the state-of-the-art. Joint training can extend to general cascaded CNNs, and we show how to jointly train RPN and fast R-CNN as an example.

Acknowledgment

This work is partially done when the first author was an intern at SenseTime. This work was partly supported by National Natural Science Foundation of China (Grant No. 71171121), National 863 High Technology Research and Development Program of China (Grant No. 2012AA09A408), and Shenzhen Science and Technology Project (Grant No. JCYJ20151117173236192 and No. CXZZ20140902110505864).

References

[1] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 236–243. IEEE, 2005. 2

[2] D. Chen, S. Ren, Y. Wei, X. Cao, and J. Sun. Joint cascade face detection and alignment. In *Computer Vision–ECCV 2014*, pages 109–122. Springer, 2014. 2

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 4

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 6

[5] S. S. Farfade, M. Saberian, and L.-J. Li. Multi-view face detection using deep convolutional neural networks. *arXiv preprint arXiv:1502.02766*, 2015. 2

[6] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 2241–2248. IEEE, 2010. 1

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010. 1, 2

[8] G. Ghiasi and C. C. Fowlkes. Occlusion coherence: Detecting and localizing occluded faces. *arXiv preprint arXiv:1506.08347*, 2015. 2

[9] R. Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015. 3

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014. 3

[11] C. Huang, H. Ai, Y. Li, and S. Lao. High-performance rotation invariant multiview face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(4):671–686, 2007. 2

[12] L. Huang, Y. Yang, Y. Deng, and Y. Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015. 2

[13] V. Jain and E. G. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. *UMass Amherst Technical Report*, 2010. 2, 6

[14] M. Jones and P. Viola. Fast multi-view face detection. *Mitsubishi Electric Research Lab TR-20003-96*, 3:14, 2003. 2

[15] M. Köstinger, P. Wohlhart, P. M. Roth, and H. 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. 5

[16] H. Li, Z. Lin, X. Shen, J. Brandt, and G. 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. 1, 2, 3, 6, 7, 8

[17] S. Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *Computer VisionECCV 2002*, pages 67–81. Springer, 2002. 2

[18] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In *Computer Vision–ECCV 2014*, pages 720–735. Springer, 2014. 2, 6, 7

[19] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 130–136. IEEE, 1997. 2

[20] R. Ranjan, V. M. Patel, and R. Chellappa. A deep pyramid deformable part model for face detection. *arXiv preprint arXiv:1508.04389*, 2015. 2

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 2, 3, 8

[22] H. Rowley, S. Baluja, T. Kanade, et al. Neural network-based face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):23–38, 1998. 2

[23] H. Rowley, S. Baluja, T. Kanade, et al. Rotation invariant neural network-based face detection. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 38–44. IEEE, 1998. 2

[24] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 746–751. IEEE, 2000. 2

[25] X. Shen, Z. Lin, J. Brandt, and Y. Wu. Detecting and aligning faces by image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3460–3467. IEEE, 2013. 7

[26] R. Vaillant, C. Monrocq, and Y. Le Cun. Original approach for the localisation of objects in images. *IEE Proceedings-Vision, Image and Signal Processing*, 141(4):245–250, 1994. 2

[27] P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004. 1, 2

[28] R. Xiao, L. Zhu, and H.-J. Zhang. Boosting chain learning for object detection. In *Computer Vision, 2003. Proceed-*

ings. Ninth IEEE International Conference on, pages 709–715. IEEE, 2003. 2

[29] J. Yan, Z. Lei, L. Wen, and S. Z. Li. The fastest deformable part model for object detection. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2497–2504. IEEE, 2014. 2

[30] J. Yan, X. Zhang, Z. Lei, and S. Z. Li. Face detection by structural models. *Image and Vision Computing*, 32(10):790–799, 2014. 2, 7

[31] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Aggregate channel features for multi-view face detection. In *Biometrics (IJCB), 2014 IEEE International Joint Conference on*, pages 1–8. IEEE, 2014. 2

[32] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Convolutional channel features for pedestrian, face and edge detection. *arXiv preprint arXiv:1504.07339*, 2015. 2

[33] M.-H. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(1):34–58, 2002. 2

[34] S. Yang, P. Luo, C. C. Loy, and X. Tang. From facial parts responses to face detection: A deep learning approach. *arXiv preprint arXiv:1509.06451*, 2015. 2, 7, 8

[35] S. Zafeiriou, C. Zhang, and Z. Zhang. A survey on face detection in the wild: past, present and future. *Computer Vision and Image Understanding*, 2015. 2

[36] C. Zhang, J. C. Platt, and P. A. Viola. Multiple instance boosting for object detection. In *Advances in neural information processing systems*, pages 1417–1424, 2005. 2

[37] C. Zhang and Z. Zhang. A survey of recent advances in face detection. Technical report, Tech. rep., Microsoft Research, 2010. 2

[38] L. Zhang, R. Chu, S. Xiang, S. Liao, and S. Z. Li. Face detection based on multi-block lbp representation. In *Advances in biometrics*, pages 11–18. Springer, 2007. 2

[39] X. Zhu and D. 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. 2, 6, 7