MTCNN

Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks

A

by Yumin Cho

Paper

K. Zhang, Z. Zhang, Z. Li, Y.J.I.S.P.L. Qiao. Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks, IEEE Signal Process Lett. 23(10):1499-1503, 2016.

Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks

Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, *Senior Member, IEEE*, and Yu Qiao, *Senior Member, IEEE*

***Abstract*—Face detection and alignment in unconstrained environment are challenging due to various poses, illuminations and occlusions. Recent studies show that deep learning approaches can achieve impressive performance on these two tasks. In this paper, we propose a deep cascaded multi-task framework which exploits the inherent correlation between them to boost up their performance. In particular, our framework adopts a cascaded structure with three stages of carefully designed deep convolutional networks that predict face and landmark location in a coarse-to-fine manner. In addition, in the learning process, we propose a new online hard sample mining strategy that can improve the performance automatically without manual sample selection. Our method achieves superior accuracy over the state-of-the-art techniques on the challenging FDDB and WIDER FACE benchmark for face detection, and AFLW benchmark for face alignment, while keeps real time performance.**

***Index Terms*—Face detection, face alignment, cascaded convolutional neural network**

I. Introduction

FACE detection and alignment are essential to many face applications, such as face recognition and facial expression analysis. However, the large visual variations of faces, such as occlusions, large pose variations and extreme lightings, impose great challenges for these tasks in real world applications.

The cascade face detector proposed by Viola and Jones [2] utilizes Haar-Like features and AdaBoost to train cascaded classifiers, which achieve good performance with real-time efficiency. However, quite a few works [1, 3, 4] indicate that this detector may degrade significantly in real-world applications with larger visual variations of human faces even with more advanced features and classifiers. Besides the cascade structure, [5, 6, 7] introduce deformable part models (DPM) for face detection and achieve remarkable performance. However, they need high computational expense and may usually require expensive annotation in the training stage. Recently, convolutional neural networks (CNNs) achieve remarkable progresses in a variety of computer vision tasks, such as image classification [9] and face recognition [10]. Inspired by the good performance of CNNs in computer vision tasks, some of the CNNs based face detection approaches have been proposed in recent years. Yang *et al.* [11] train deep convolution neural networks for facial attribute recognition to obtain high response in face regions which further yield candidate windows of faces. However, due to its complex CNN structure, this approach is time costly in practice. Li *et al.* [19] use cascaded CNNs for face detection, but it requires bounding box calibration from face detection with extra computational expense and ignores the inherent correlation between facial landmarks localization and bounding box regression.

Face alignment also attracts extensive interests. Regression-based methods [12, 13, 16] and template fitting approaches [14, 15, 7] are two popular categories. Recently, Zhang *et al.* [22] proposed to use facial attribute recognition as an auxiliary task to enhance face alignment performance using deep convolutional neural network.

However, most of the available face detection and face alignment methods ignore the inherent correlation between these two tasks. Though there exist several works attempt to jointly solve them, there are still limitations in these works. For example, Chen *et al.* [18] jointly conduct alignment and detection with random forest using features of pixel value difference. But, the handcraft features used limits its performance. Zhang *et al.* [20] use multi-task CNN to improve the accuracy of multi-view face detection, but the detection accuracy is limited by the initial detection windows produced by a weak face detector.

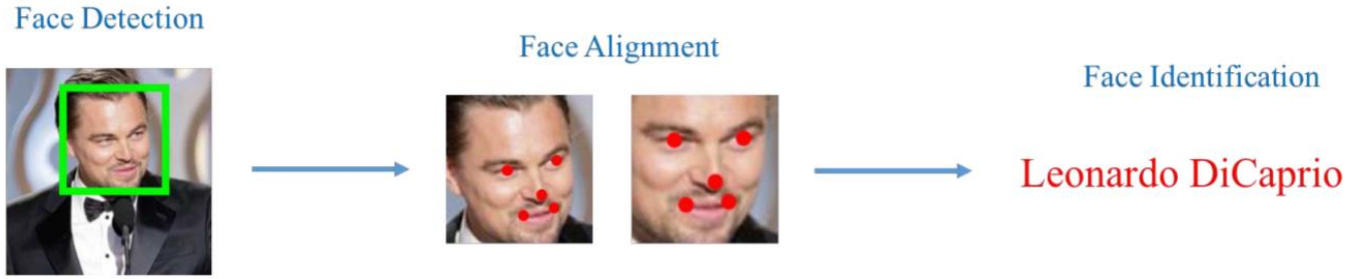
On the other hand, in the training process, mining hard samples in training is critical to strengthen the power of detector. However, traditional hard sample mining usually performs an offline manner, which significantly increases the manual operations. It is desirable to design an online hard sample mining method for face detection and alignment, which is adaptive to the current training process automatically.

In this paper, we propose a new framework to integrate these two tasks using unified cascaded CNNs by multi-task learning. The proposed CNNs consist of three stages. In the first stage, it produces candidate windows quickly through a shallow CNN. Then, it refines the windows to reject a large number of non-faces windows through a more complex CNN. Finally, it uses a more powerful CNN to refine the result and output facial landmarks positions. Thanks to this multi-task learning framework, the performance of the algorithm can be notably improved. The major contributions of this paper are summarized as follows: (1) We propose a new cascaded CNNs based framework for joint face detection and alignment, and carefully

K.-P. Zhang, Z.-F. Li and Y. Q. are with the Multimedia Laboratory, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. E-mail: kp.zhang@siat.ac.cn; zhifeng.li@siat.ac.cn; yu.qiao@siat.ac.cn

Z.-P. Zhang is with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong. E-mail: zz013@ie.cuhk.edu.hk

Face Detection vs Face Alignment



Problem

Difficult to apply in real-world and high computations

Cascade face detector may degrade significantly in real-world applications

Deformable Part Models(**DPM**) require expensive annotation in the training stage

Separate the face detection and face alignment

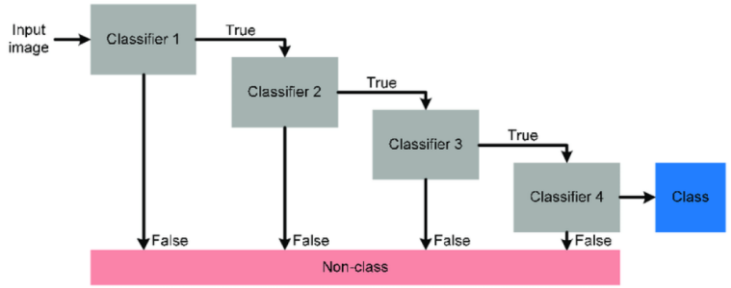
Most of the available face detection and face alignment methods **ignore the inherent correlation** between these two tasks

Traditional hard sample mining

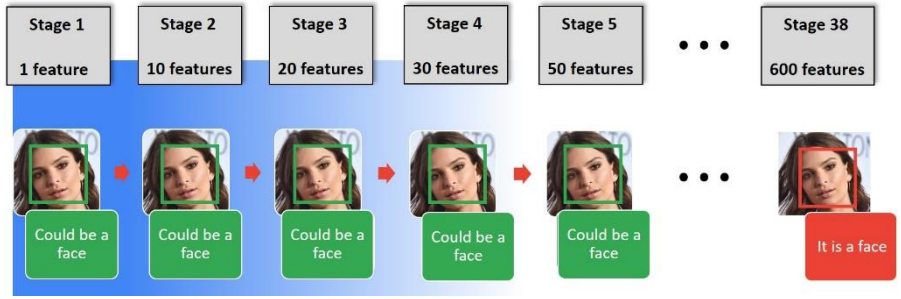
It performs an **offline manner**, which significantly increases the manual operations.

CNN Cascaded

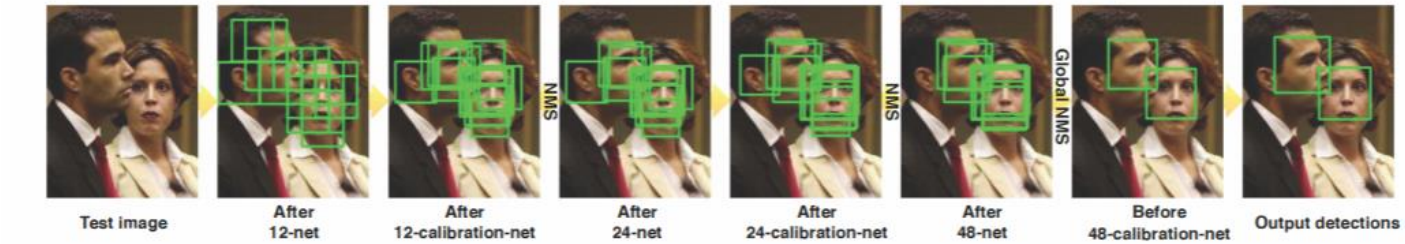
Cascade Classifier



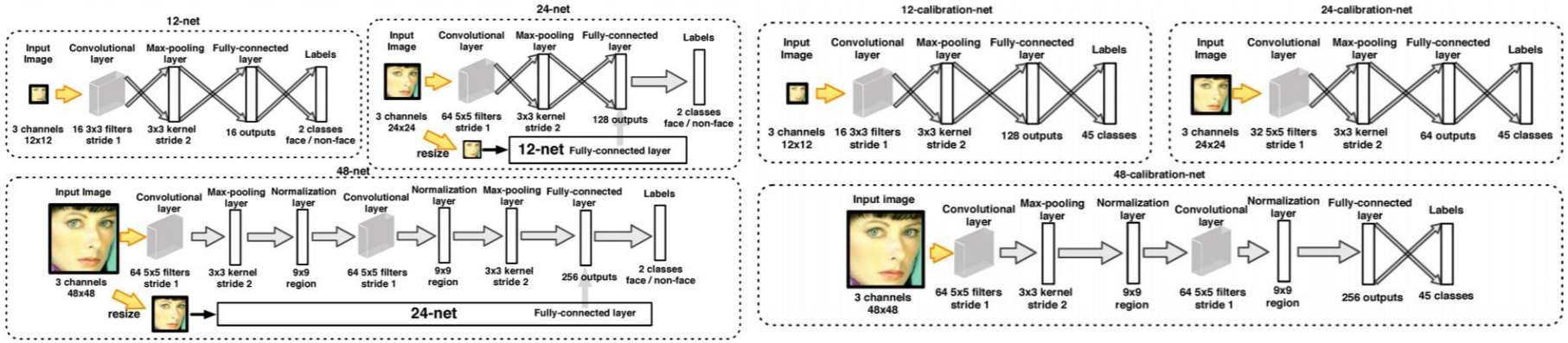
cf.) Haar Cascade (Viola- Jones algorithm)



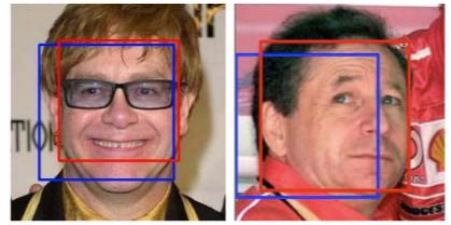
CNN Cascaded



Detection windows are reduced and calibrated **from stage to stage** in the detector.

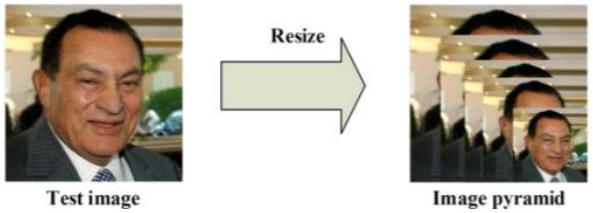


Bounding box calibration



Adjust its **size and location** to approach a potential face nearby

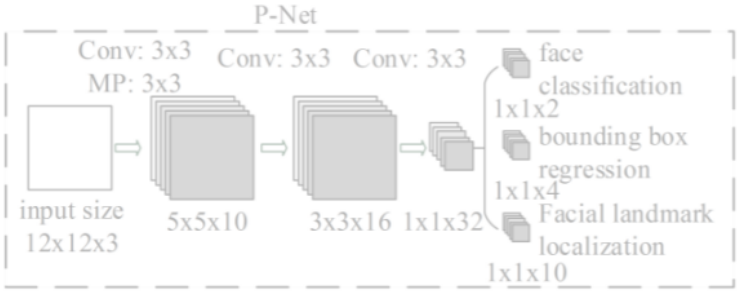
Approach: P-Net

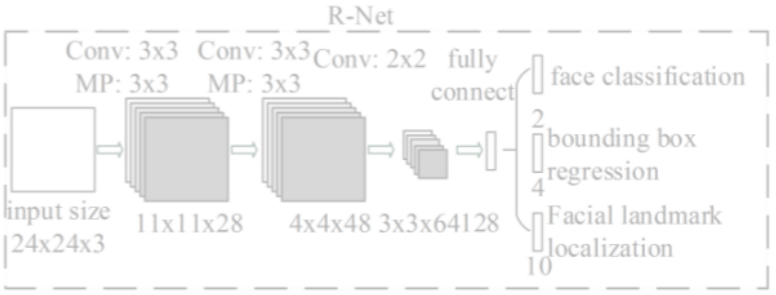


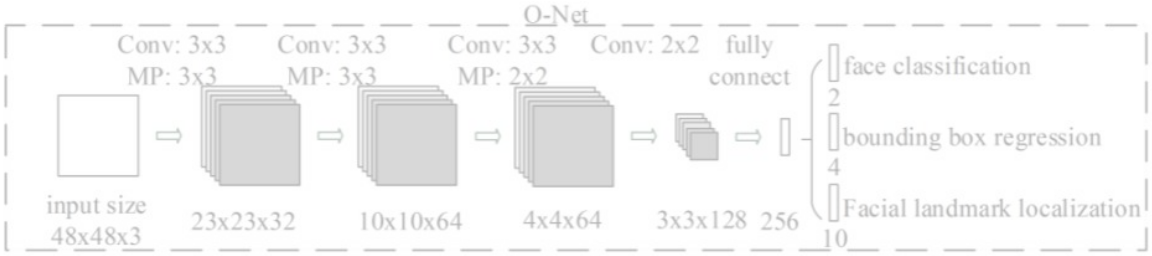
Stage 0. Image pyramid

Resize the images to different scales

Detect faces of all different sizes using 12x12 kernel





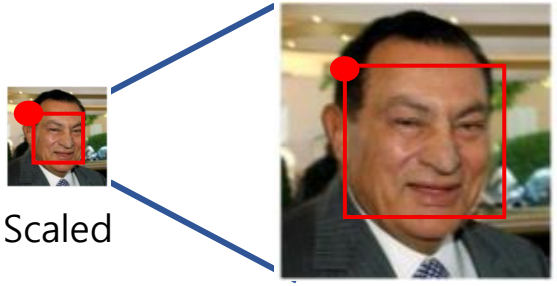


Face?

Not Found ☹ Found ☺ Not Found ☹ Found ☺

Standardize the coordinate system

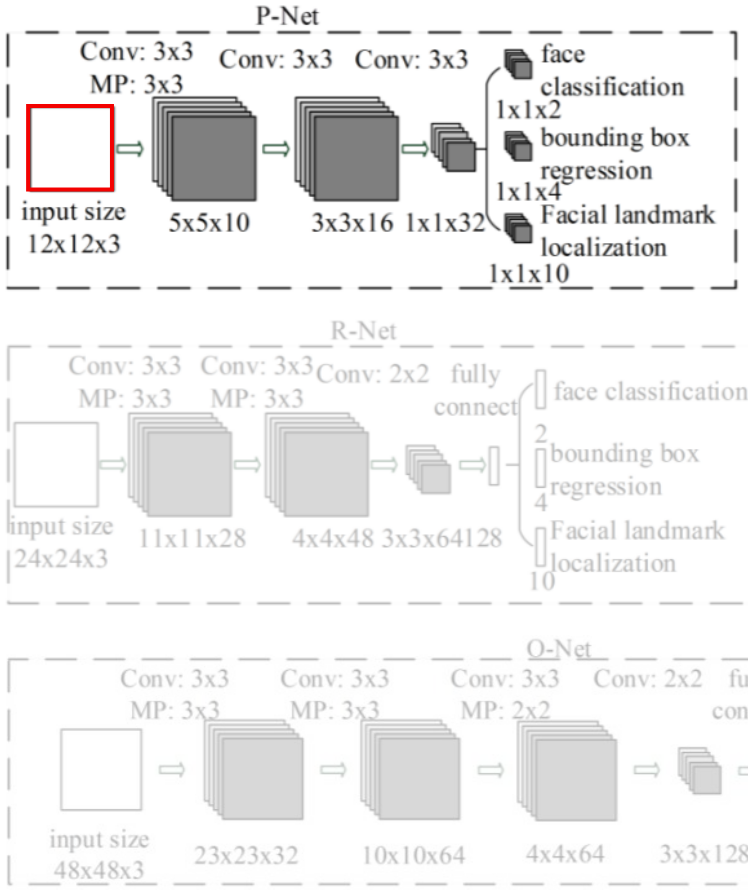
Convert all the coordinate systems to that of the actual 'un-scaled' image



Scaled

Un-scaled

Approach: P-Net

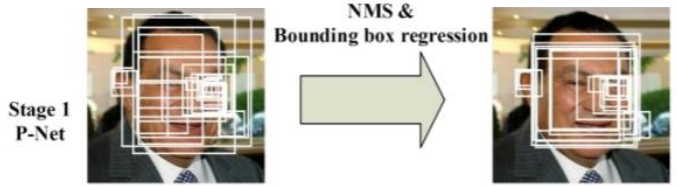


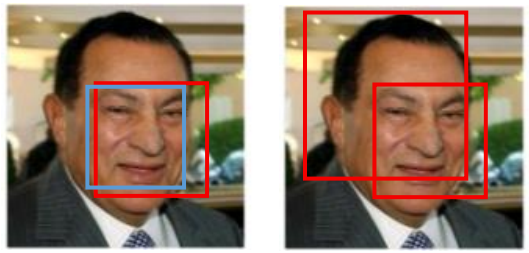
Stage 1. P-Net

Exploit a fully convolutional network

Obtain the **candidate windows** and their **bounding box regression vectors**

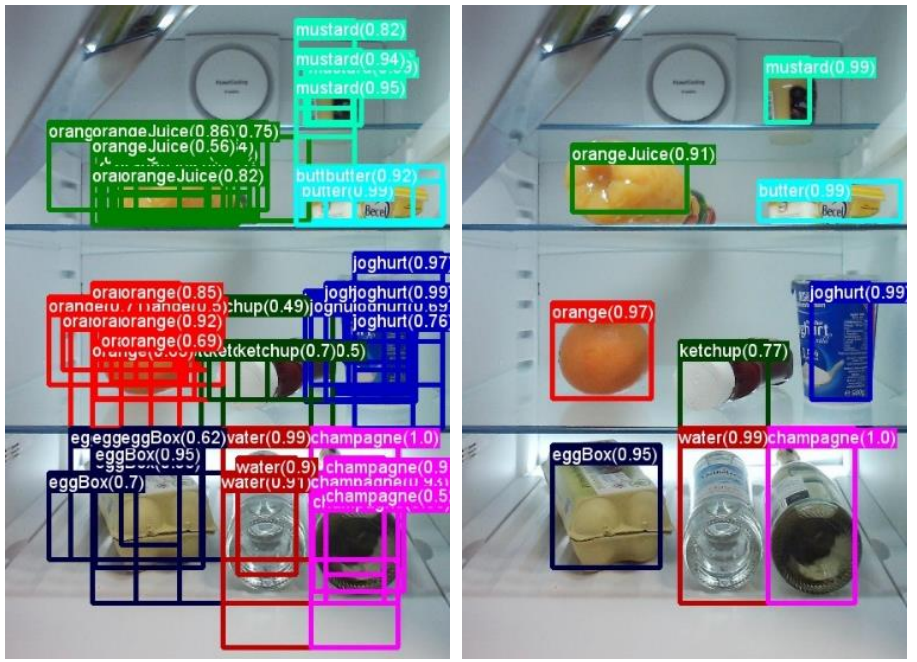
Employ non-maximum suppression(**NMS**) to merge highly overlapped candidates





Delete! Remain!

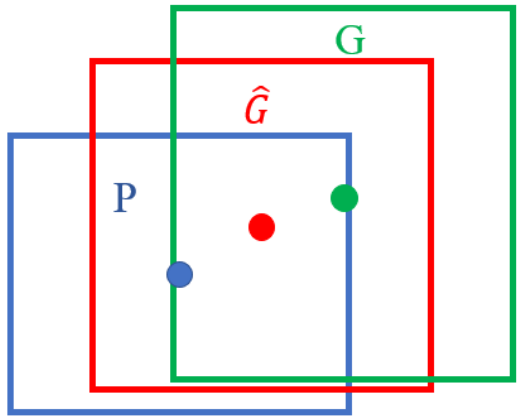
cf.) Non-Maximum Suppression



Bounding Box Regression

Refine the bounding box to reduce the localization error

Learn **scale-invariant** transformation between two centers and **log-scale** transformation between widths and heights



$$P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$$

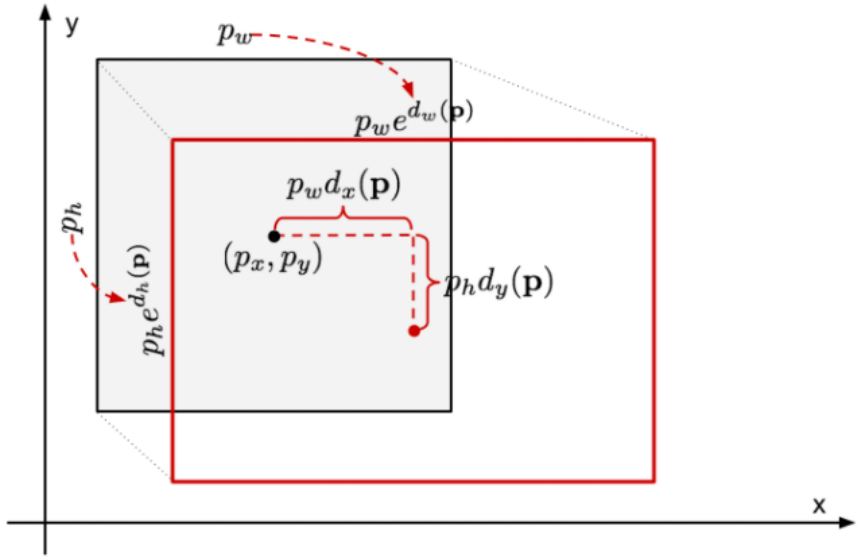
$$G = (G_x, G_y, G_w, G_h)$$

Bounding box correction functions **d**

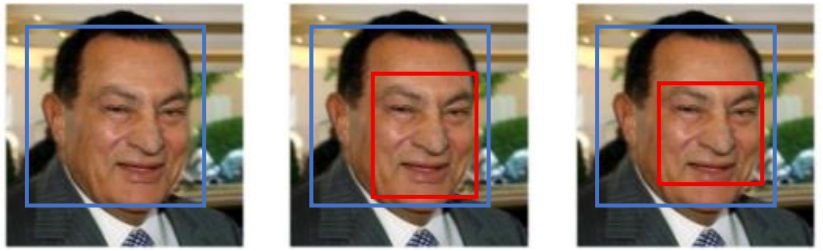
$$d_x(P),\ d_y(P),\ d_w(P),\ \text{and}\ d_h(P)$$

$$\begin{aligned}\hat{g}_x &= p_w d_x(\mathbf{p}) + p_x \\ \hat{g}_y &= p_h d_y(\mathbf{p}) + p_y \\ \hat{g}_w &= p_w \exp(d_w(\mathbf{p})) \\ \hat{g}_h &= p_h \exp(d_h(\mathbf{p}))\end{aligned} \quad \Rightarrow \quad \begin{aligned} t_x &= (g_x - p_x)/p_w \\ t_y &= (g_y - p_y)/p_h \\ t_w &= \log(g_w/p_w) \\ t_h &= \log(g_h/p_h)\end{aligned}$$

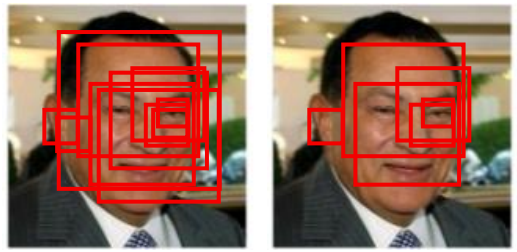
$$\mathcal{L}_{\text{reg}} = \sum_{i \in \{x,y,w,h\}} (t_i - d_i(\mathbf{p}))^2 + \lambda \|\mathbf{w}\|^2$$



BB Regression



NMS

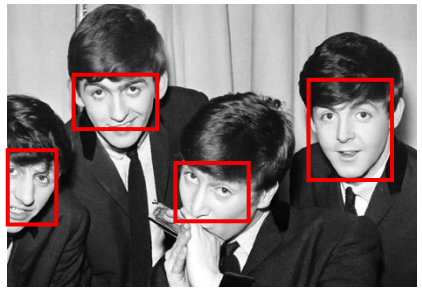


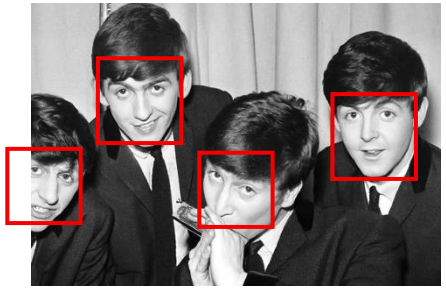
Stage 2 R-Net ?

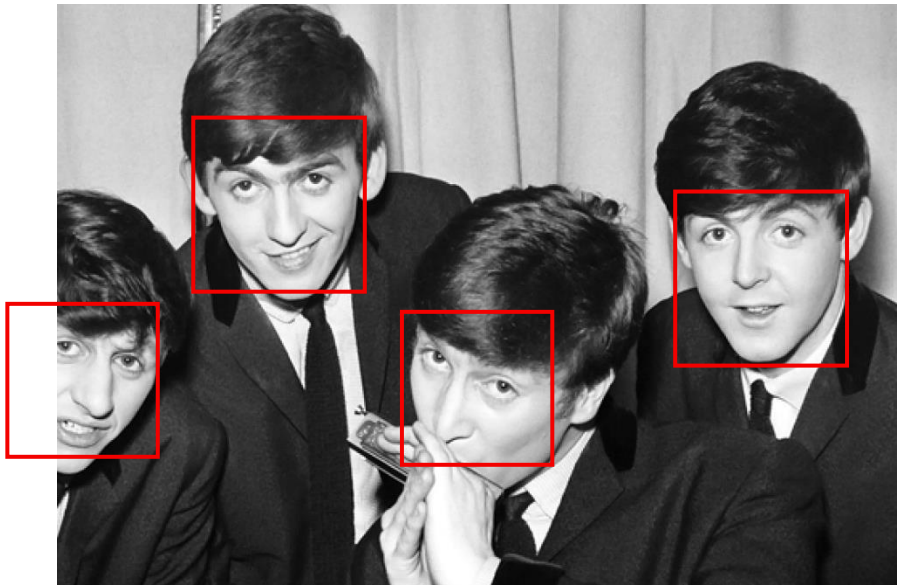
Padding

Zero padding

If the bounding box is **out of bounds**, copy the portion of the image in the bounding box to the new array and fill in everything else with a **0**



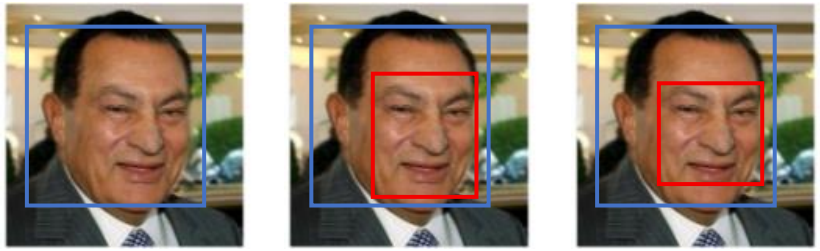




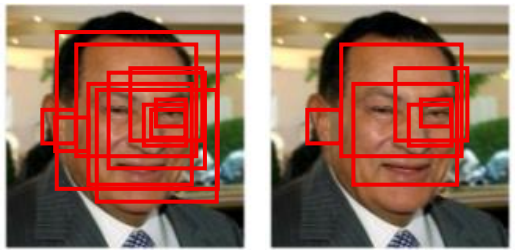
```
@staticmethod
def __pad(total_boxes, w, h):
    # compute the padding coordinates (pad the bounding boxes to square)
    tmpw = (total_boxes[:, 2] - total_boxes[:, 0] + 1).astype(np.int32)
    tmph = (total_boxes[:, 3] - total_boxes[:, 1] + 1).astype(np.int32)
    numbox = total_boxes.shape[0]

    dx = np.ones(numbox, dtype=np.int32)
    dy = np.ones(numbox, dtype=np.int32)
    edx = tmpw.copy().astype(np.int32)
    edy = tmph.copy().astype(np.int32)

    x = total_boxes[:, 0].copy().astype(np.int32)
    y = total_boxes[:, 1].copy().astype(np.int32)
    ex = total_boxes[:, 2].copy().astype(np.int32)
    ey = total_boxes[:, 3].copy().astype(np.int32)
```

```
    tmp = np.where(ex > w)
    edx.flat[tmp] = np.expand_dims(-ex[tmp] + w + tmpw[tmp], 1)
    ex[tmp] = w

    tmp = np.where(ey > h)
    edy.flat[tmp] = np.expand_dims(-ey[tmp] + h + tmph[tmp], 1)
    ey[tmp] = h

    tmp = np.where(x < 1)
    dx.flat[tmp] = np.expand_dims(2 - x[tmp], 1)
    x[tmp] = 1

    tmp = np.where(y < 1)
    dy.flat[tmp] = np.expand_dims(2 - y[tmp], 1)
    y[tmp] = 1

    return dy, edy, dx, edx, y, ey, x, ex, tmpw, tmph
```

BB Regression

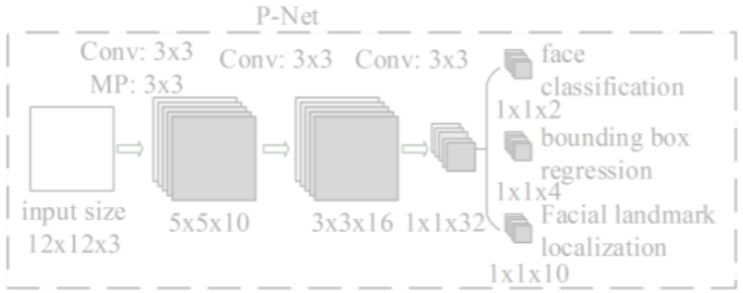


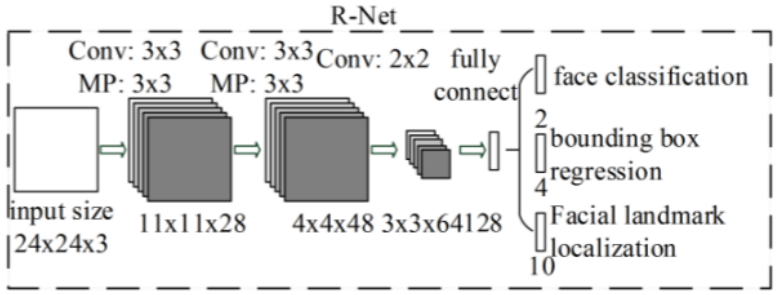
NMS

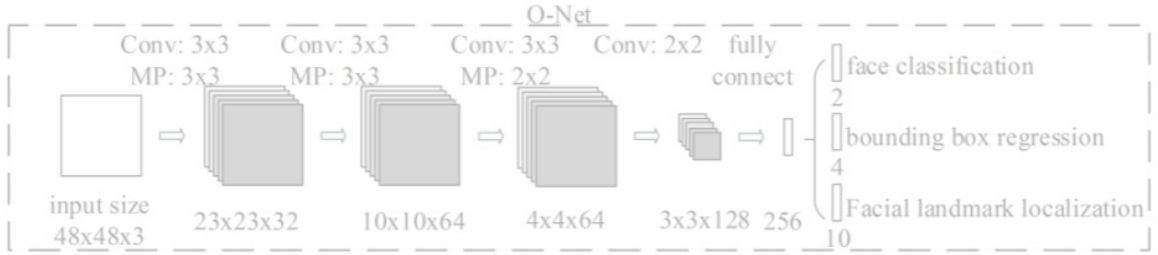


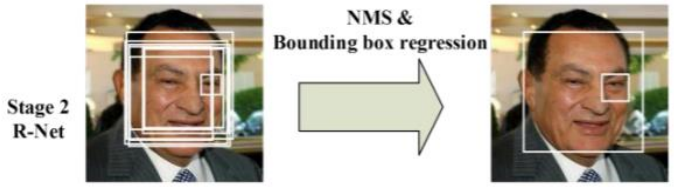
Stage 2 R-Net ?

Approach: R-Net









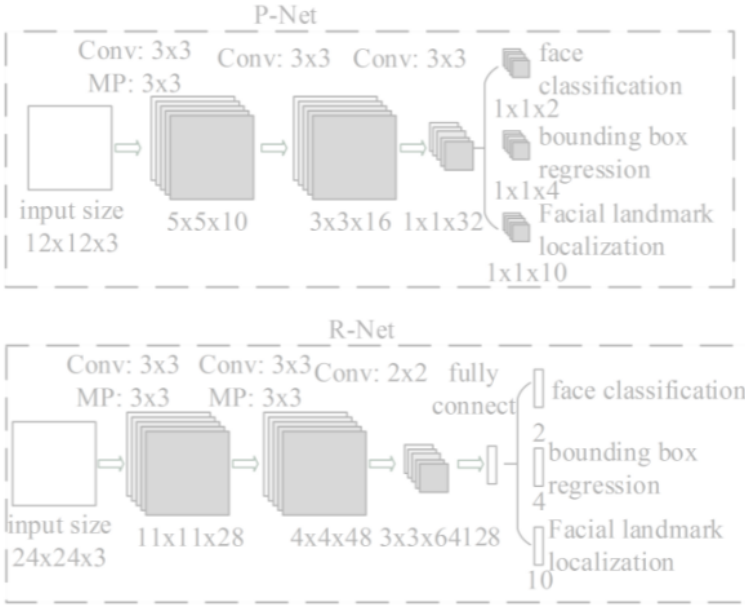
Stage 2 R-Net

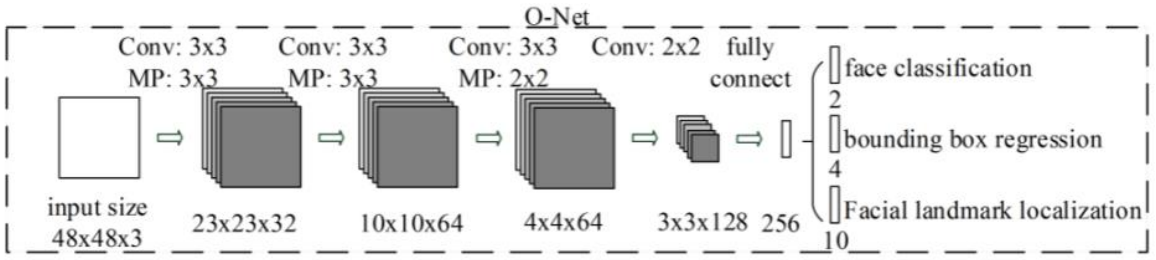
Refine those candidates

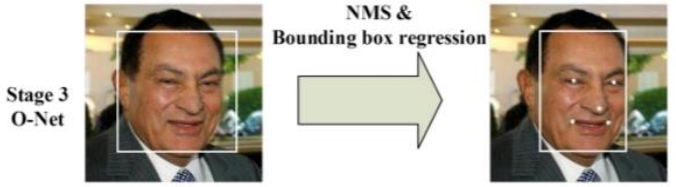
Reject large number of **false candidates**

Perform **calibration** with bounding box regression and **NMS** candidate merge

Approach: O-Net







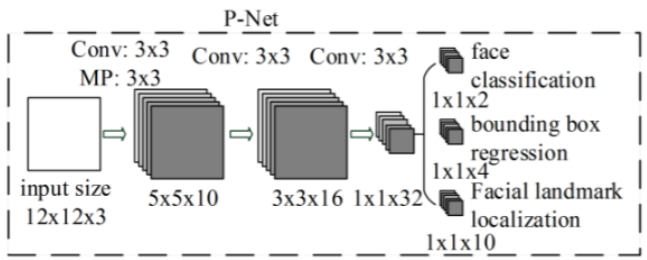
Stage 3. O-Net

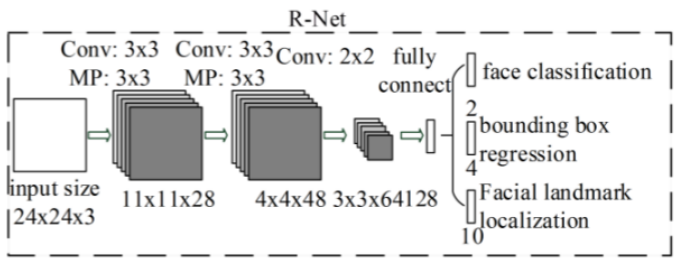
Produce final bounding box and facial landmarks position

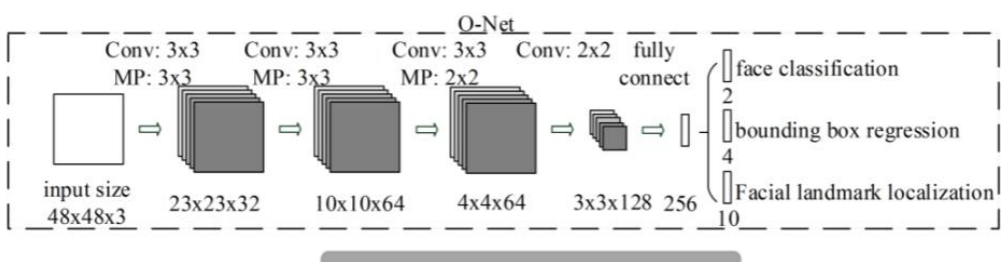
Aim to describe the face in more details

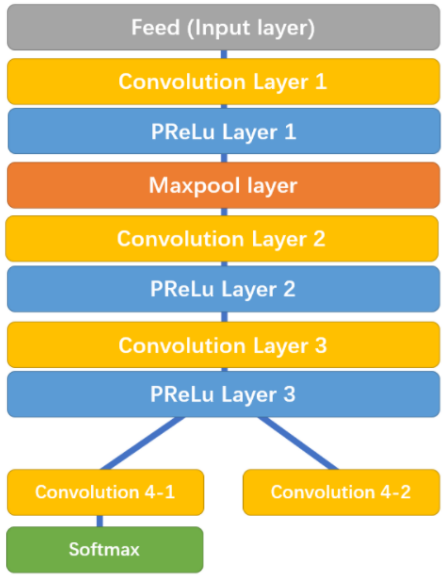
Output five **facial landmarks' positions**

Approach:

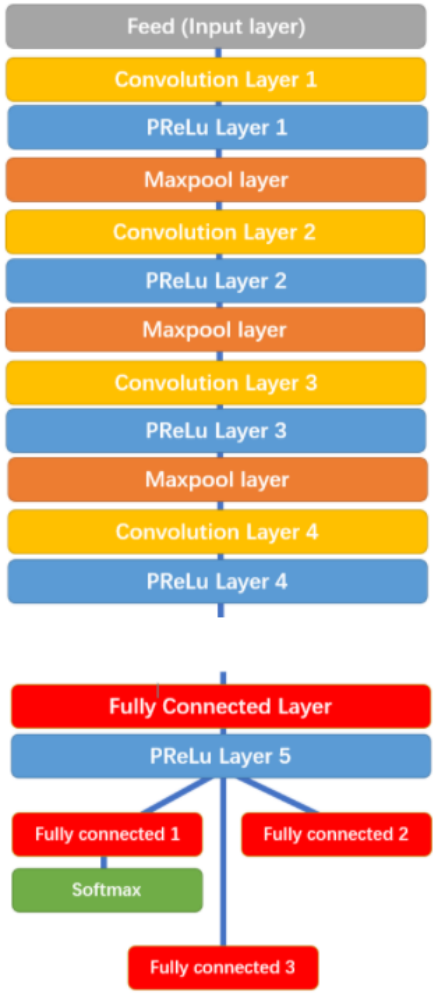

P-Net
Conv: 3x3
MP: 3x3
Conv: 3x3
Conv: 3x3
face classification
1x1x2
bounding box regression
1x1x4
Facial landmark localization
1x1x10
input size
12x12x3
5x5x10
3x3x16
1x1x32


R-Net
Conv: 3x3
MP: 3x3
Conv: 3x3
MP: 3x3
Conv: 2x2
fully connect
face classification
2
bounding box regression
4
Facial landmark localization
10
input size
24x24x3
11x11x28
4x4x48
3x3x64
128


O-Net
Conv: 3x3
MP: 3x3
Conv: 3x3
MP: 3x3
Conv: 3x3
MP: 2x2
Conv: 2x2
fully connect
face classification
2
bounding box regression
4
Facial landmark localization
10
input size
48x48x3
23x23x32
10x10x64
4x4x64
3x3x128
256


Feed (Input layer)
Convolution Layer 1
PReLu Layer 1
Maxpool layer
Convolution Layer 2
PReLu Layer 2
Convolution Layer 3
PReLu Layer 3
Convolution 4-1
Convolution 4-2
Softmax


Feed (Input layer)
Convolution Layer 1
PReLu Layer 1
Maxpool layer
Convolution Layer 2
PReLu Layer 2
Maxpool layer
Convolution Layer 3
PReLu Layer 3
Fully Connected 1
PReLu Layer 4
Fully connected 1
Fully connected 2
Softmax


Feed (Input layer)
Convolution Layer 1
PReLu Layer 1
Maxpool layer
Convolution Layer 2
PReLu Layer 2
Maxpool layer
Convolution Layer 3
PReLu Layer 3
Maxpool layer
Convolution Layer 4
PReLu Layer 4
Fully Connected Layer
PReLu Layer 5
Fully connected 1
Fully connected 2
Softmax
Fully connected 3

Training

We leverage three tasks to train our CNN detectors: face/non-face classification, bounding box regression, and facial landmark localization.

1) Face classification: The learning objective is formulated as a two-class classification problem. For each sample x_i, we use the cross-entropy loss:

$$L_i^{det} = -(y_i^{det} \log(p_i) + (1 - y_i^{det})(1 - \log(p_i))) \quad (1)$$

where p_i is the probability produced by the network that indicates a sample being a face. The notation $y_i^{det} \in \{0,1\}$ denotes the ground-truth label.

2) Bounding box regression: For each candidate window, we predict the offset between it and the nearest ground truth (i.e., the bounding boxes' left top, height, and width). The learning objective is formulated as a regression problem, and we employ the Euclidean loss for each sample x_i:

$$L_i^{box} = \left\| \hat{y}_i^{box} - y_i^{box} \right\|_2^2 \quad (2)$$

where $\hat{y}_i^{box}$ regression target obtained from the network and y_i^{box} is the ground-truth coordinate. There are four coordinates, including left top, height and width, and thus $y_i^{box} \in \mathbb{R}^4$.

3) Facial landmark localization: Similar to the bounding box regression task, facial landmark detection is formulated as a regression problem and we minimize the Euclidean loss:

$$L_i^{landmark} = \left\| \hat{y}_i^{landmark} - y_i^{landmark} \right\|_2^2 \quad (3)$$

where $\hat{y}_i^{landmark}$ is the facial landmark's coordinate obtained from the network and $y_i^{landmark}$ is the ground-truth coordinate. There are five facial landmarks, including left eye, right eye, nose, left mouth corner, and right mouth corner, and thus $y_i^{landmark} \in \mathbb{R}^{10}$.

Loss function

Face classification

$$L_i^{det} = -(y_i^{det} \log(p_i) + (1 - y_i^{det})(1 - \log(p_i)))$$

Cross-entropy loss

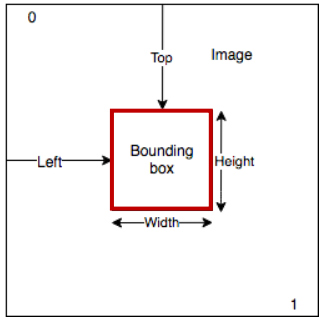
$$y_i^{det} \in \{0,1\}$$

Bounding box regression

Euclidean loss

$$L_i^{box} = \left\| \hat{y}_i^{box} - y_i^{box} \right\|_2^2$$

$$y_i^{box} \in \mathbb{R}^4$$

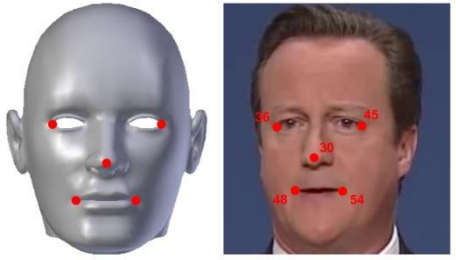


Facial landmark localization

Euclidean loss

$$L_i^{landmark} = \left\| \hat{y}_i^{landmark} - y_i^{landmark} \right\|_2^2$$

$$y_i^{landmark} \in \mathbb{R}^{10}$$



Training

4) Multi-source training: Since we employ different tasks in each CNNs, there are different types of training images in the learning process, such as face, non-face and partially aligned face. In this case, some of the loss functions (i.e., Eq. (1)-(3)) are not used. For example, for the sample of background region, we only compute L_i^{det}, and the other two losses are set as 0. This can be implemented directly with a sample type indicator. Then the overall learning target can be formulated as:

$$\min \sum_{i=1}^{N} \sum_{j \in \{det, box, landmark\}} \alpha_j \beta_i^j L_i^j \qquad (4)$$

where N is the number of training samples. α_j denotes on the task importance. We use ($\alpha_{det} = 1, \alpha_{box} = 0.5, \alpha_{landmark} = 0.5$) in P-Net and R-Net, while ($\alpha_{det} = 1, \alpha_{box} = 0.5, \alpha_{landmark} = 1$) in O-Net for more accurate facial landmarks localization. $\beta_i^j \in \{0,1\}$ is the sample type indicator. In this case, it is natural to employ stochastic gradient descent to train the CNNs.

⋮

5) Online Hard sample mining: Different from conducting traditional hard sample mining after original classifier had been trained, we do online hard sample mining in face classification task to be adaptive to the training process.

In particular, in each mini-batch, we sort the loss computed in the forward propagation phase from all samples and select the top 70% of them as hard samples. Then we only compute the gradient from the hard samples in the backward propagation phase. That means we ignore the easy samples that are less helpful to strengthen the detector while training. Experiments show that this strategy yields better performance without manual sample selection. Its effectiveness is demonstrated in the Section III.

Multi-source training

$\min \sum_{i=1}^{N} \sum_{j \in \{det, box, landmark\}} \alpha_j \beta_i^j L_i^j$

α_j = task importance

P-Net, R-Net = (1, 0.5, 0.5)

O-Net = (1, 0.5, 1)

$\beta_i^j \in \{0,1\}$ Whether the sample is selected or not

OHEM

Select top 70% as hard samples

B. The effectiveness of online hard sample mining

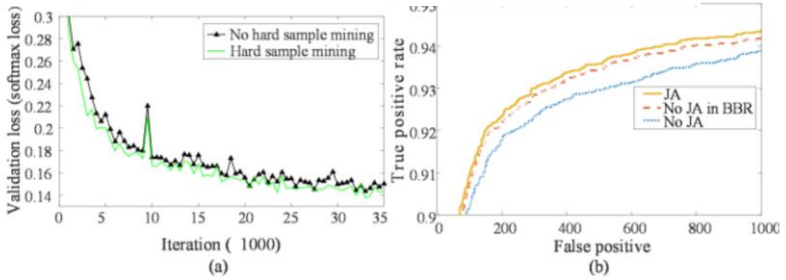


Fig. 3. (a) Validation loss of O-Net with and without hard sample mining. (b) "JA" denotes joint face alignment learning while "No JA" denotes do not joint it. "No JA in BBR" denotes do not joint it while training the CNN for bounding box regression.

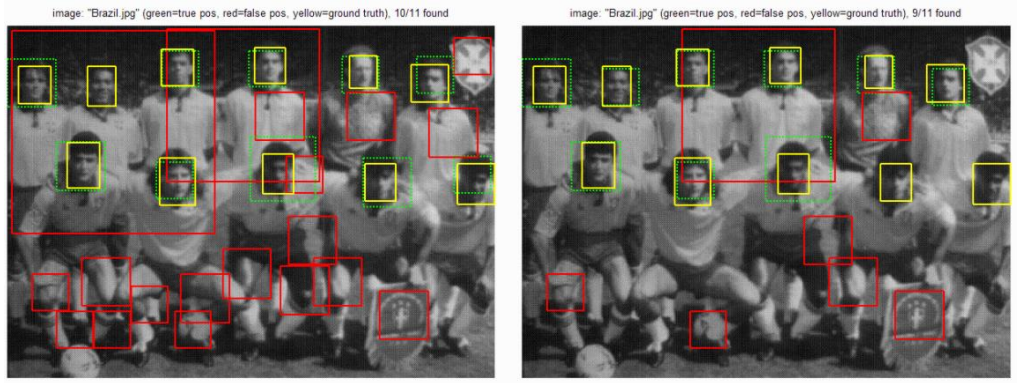
Online Hard Example Mining(OHEM)

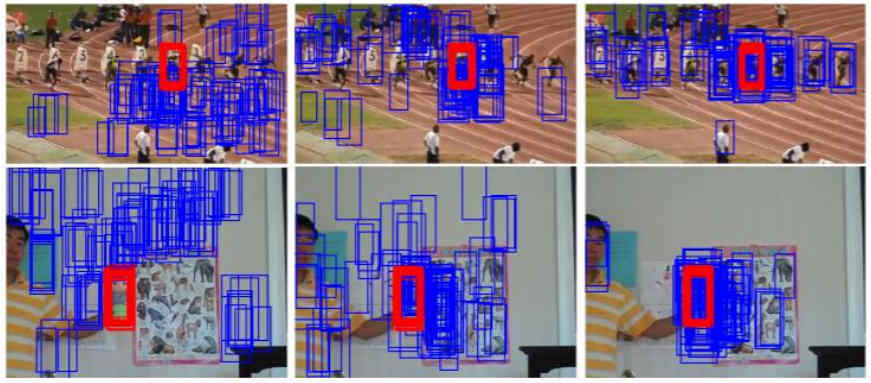
Problem

Hard sample mining

Identify the **false positives** in the training set and using those images as negative examples to re-train

= **Bootstrapping**





(a) 1st minibatch (b) 5th minibatch (c) 30th minibatch

Solution

Batch-wise hard sample mining

Performs **regular forward** propagation and computes loss and then find hard examples in the batch with **high loss values**.

Only back-propagates the loss over the selected examples

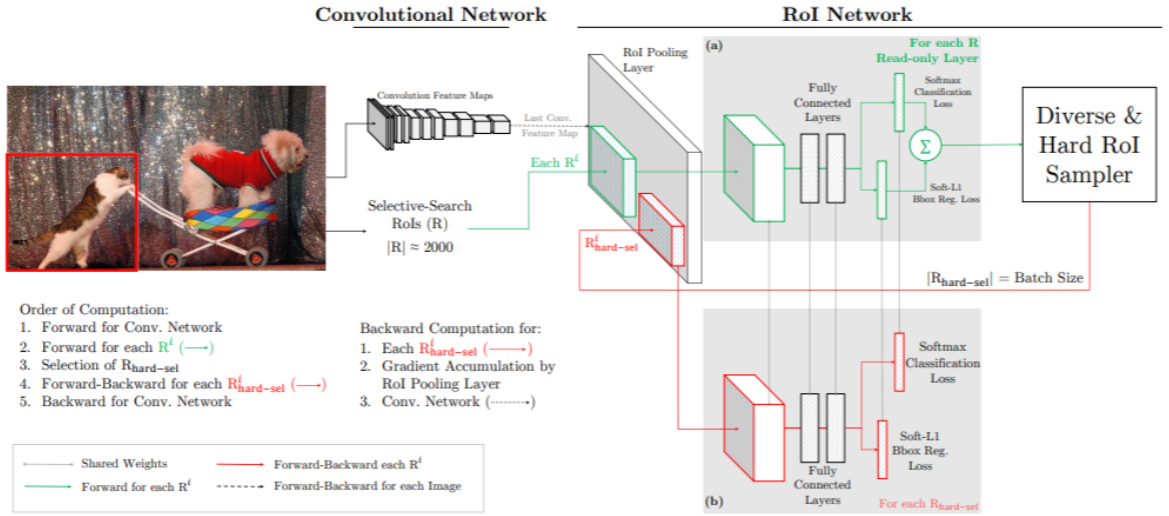


Figure 2: Architecture of the proposed training algorithm. Given an image, and selective search RoIs, the conv network computes a conv feature map. In (a), the *readonly* RoI network runs a forward pass on the feature map and all RoIs (shown in green arrows). Then the Hard RoI module uses these RoI losses to select B examples. In (b), these hard examples are used by the RoI network to compute forward and backward passes (shown in red arrows).

Configure data sequentially in mini-batch

= **Online**

Experiments

Data annotation and training data

1. Negatives

IoU less than 0.3

2. Positives

IoU above 0.65

3. Part faces

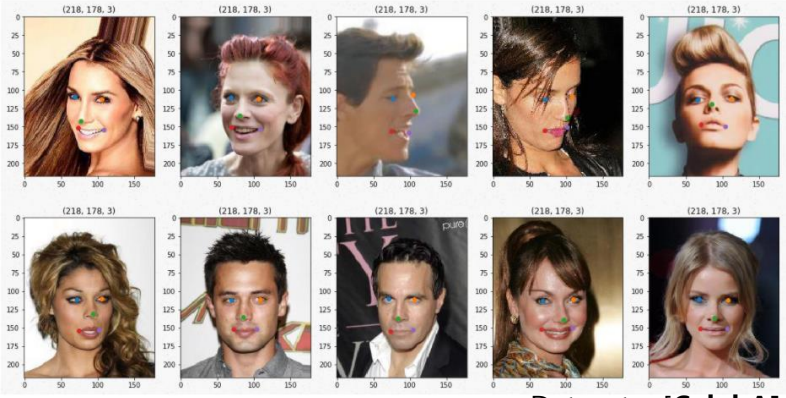
IoU between 0.4 and 0.65

→ **Face classification**

→ **Bounding box regression**

4. Landmark faces → **Landmark localization**

Faces labeled 5 landmarks' positions



Dataset - [**CelebA**]

Dataset - [**WIDER FACE**]

Results

Improvements to Cascaded CNN

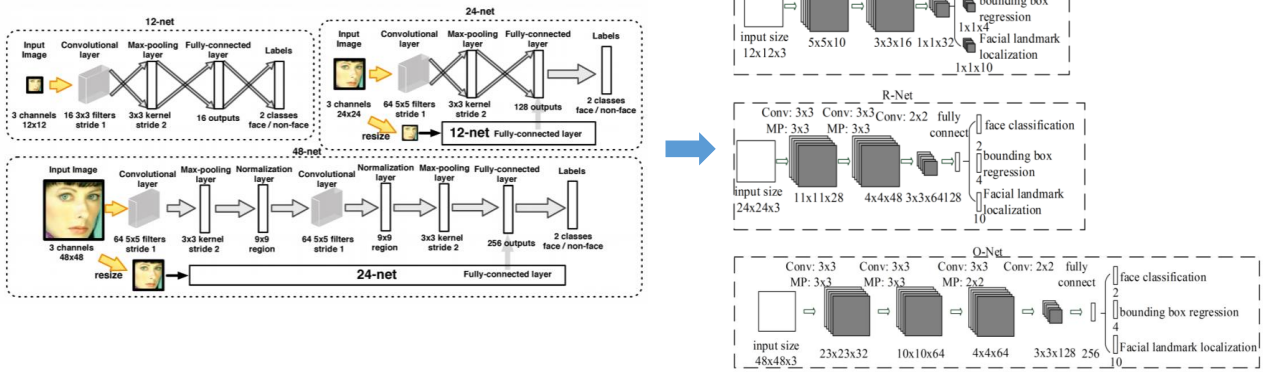
lack diversity of weights & binary classification task

Reduce the number of filters and change the 5×5 filter to a **3×3** filter

COMPARISON OF SPEED AND VALIDATION ACCURACY OF OUR CNNS AND PREVIOUS CNNS [19]

Group	CNN	300 Times Forward	Accuracy
Group1	12-Net [19]	0.038s	94.4%
Group1	P-Net	0.031s	94.6%
Group2	24-Net [19]	0.738s	95.1%
Group2	R-Net	0.458s	95.4%
Group3	48-Net [19]	3.577s	93.2%
Group3	O-Net	1.347s	95.4%

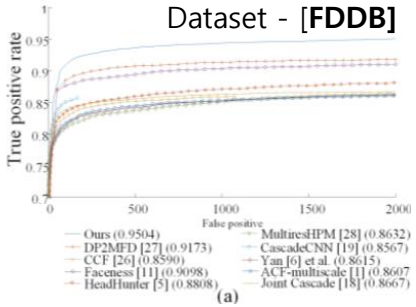
∴ Reduce the computing while increase the depth to get better performance



Evaluation to contribution of proposed model

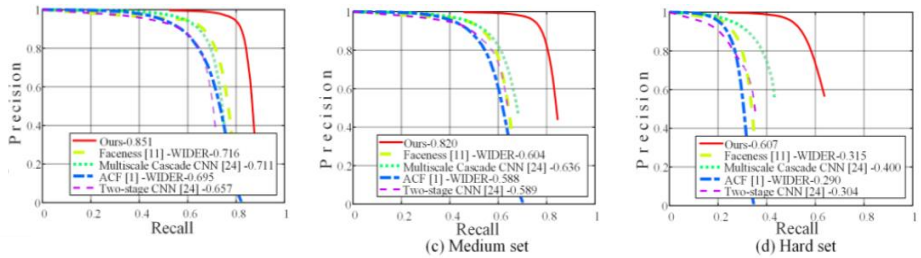
C. The effectiveness of joint detection and alignment

Dataset - [**FDDB**]

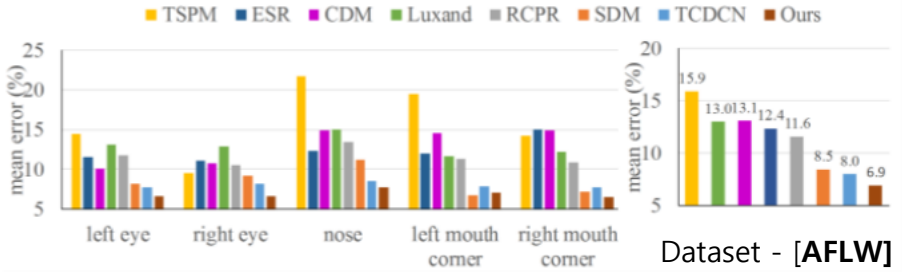


D. Evaluation on face detection

Dataset - [**WIDER FACE**]



E. Evaluation on face alignment



Dataset - [**AFLW**]

Conclusion

Contributions

We propose a new cascaded CNNs based framework for joint face detection and alignment, and carefully design lightweight CNN architecture for real time performance

We propose an effective method to conduct online hard sample mining to improve the performance.

Extensive experiments are conducted on challenging benchmarks, to show the significant performance improvement of the proposed approach compared to the state-of-the-art techniques in both face detection and face alignment tasks.

References

- **Haar Cascade**

http://datahacker.rs/008-how-to-detect-faces-eyes-and-smiles-using-haar-cascade-classifiers-with-opencv-in-python/

- **OHEM**

https://arxiv.org/abs/1604.03540

- **CNN Cascaded**

https://ieeexplore.ieee.org/document/7299170

- **R-CNN**

https://arxiv.org/abs/1311.2524

- **MTCNN**

https://towardsdatascience.com/how-does-a-face-detection-program-work-using-neural-networks-17896df8e6ff