Joint Face Detection and Alignment using Multi-Task Cascaded Convolutional Networks Report

Instructor: Qinru Qiu

Team Member:
Chao Yang 461352435
Chien Chih Kao 611189677
Mengjie Shi 457056896
Xiangyu Xie 284550652

Content

Introduction

Biometrics Technology

Biometrics is based on the behavioral and biological characteristics of individuals to automatically identify.[1] The main function of biometric system is to verify and authenticate.[2] Verification is usually related to database searching, and such procedures must be rigorous enough. Jain[3] summarizes seven factors that determine the applicability of the unique physical, behavioral, and chemical characteristics of a biometric system. These are:

1) Uniqueness, which should be sufficiently different across individuals in a population.

2) Measurability, defined as the possibility of using a device to acquire a biometric.

3) The universality of technology means that access is generally acceptable.

4) Acceptability, indicating that people are willing to use the system.

5) Performance should indicate the accuracy and repeatability under given constraints.

6) Permanence, indicating that biometrics are somewhat unchanged over time.

7) Circumvention, shows that the system does not respond to fake artifacts, and refuses to imitate behavioral traits.

Current biometrics technologies include fingerprint recognition, retinal recognition, iris recognition, gait recognition, vein recognition and face recognition.[3]

Why Face Recognition

For humans, facial images may be the most common biological features used for personal identification. Therefore, the detection and recognition of faces are the basic cognitive abilities that underlie our social interaction. Since birth, humans have experienced and participated in face-to-face interactions that help identify faces. Face recognition methods are often based on the location and shape of facial attributes such as the eyes, eyebrows, nose, lips, chin shapes, and their spatial relationship.[2] Compared with other recognition methods, face recognition has been widely used in research and application due to its direct, friendly and convenient features. It is easy for users to accept without any psychological obstacles. In addition, the results of face recognition can be further analyzed, and other information such as human gender, expression, age, etc., can be obtained, and the application prospect of face recognition can be expanded.[5]

Applications

One of the reasons for face recognition, which has attracted a great deal of research attention and sustainable development in recent 30 years, and has great potential in many government and commercial applications. With the advent of many new application areas, the number of face recognition systems and commercial enterprises has greatly increased, facial recognition technology has been further improved, which makes face payment possible.[6]

Facial recognition technology, as one of the non-invasive biometrics technologies, is getting closer to people's daily life. There is evidence that in 2000, the International Civil Aviation Organization recognized facial recognition as the most suitable biometrics for air travel.[7] We roughly group these scenarios into 10 categories as shown in Table 1.[6]

Category	**Exemplar application scenarios**
Face ID	Driver licenses, entitlement programs, immigration, national ID, passports, voter registration, welfare registration
Access control	Border-crossing control, facility access, vehicle access, smart kiosk and ATM, computer access, computer program access, computer network access, online program access, online transactions access, long distance learning access, online examinations access, online database access
Security	Terrorist alert, secure flight boarding systems, stadium audience scanning, computer security, computer application security, database security, file encryption, intranet security, Internet security, medical records, secure trading terminals
Surveillance	Advanced video surveillance, nuclear plant surveillance, park surveillance, neighborhood watch, power grid surveillance, CCTV control, portal control
Smart cards	Stored value security, user authentication
Law enforcement	Crime stopping and suspect alert, shoplifter recognition, suspect tracking and investigation, suspect background check, identifying cheats and casino undesirables, post-event analysis, welfare fraud, criminal face retrieval and recognition
Face databases	Face indexing and retrieval, automatic face labeling, face classification
Multimedia management	Face-based search, face-based video segmentation and summarization, event detection
Human computer interaction	Interactive gaming, proactive computing
Others	Antique photo verification, very low bit-rate image & video transmission, etc.

Table 1

Challenges

Faces are complex objects. Given any image, the goal of face detection and recognition is to determine if there are any faces in the image, and if so, to determine the position and extent of each face; then, the face needs to be identified. While this may seem like a trivial matter to humans, it is a very challenging task for any hardware system and has been a major research topic in machine vision for decades.[2] Now work on machine inspection and face recognition can be incorporated into a common term called Automatic Face Recognition (AFR). However, the ultimate goal is to mimic the human task of face detection and recognition. The key issue is understanding how to create face representations to enable robust face recognition that people show in their daily interactions.

Approaches

There are many ways to implement the face recognition includes geometric feature-based method, subspace-based face recognition, neural network-based face recognition, correlation-based method, matching pursuit-based methods, support vector machine approach and selected works on face classifiers.
Neural networks give a non-linear solution to the problem of face recognition. The advantage of neural networks in the linear classification is that they can reduce the misclassification between neighborhood categories. The basic idea is to consider the neural network of each pixel in the image.[2]

In this report, we will discuss neural network-based face recognition from the aspects of theory, structure, implementation and other details.

Architecture

3-Stage Architecture

The cascade face detector proposed by Viola and Jones[8] utilizes Haar-Like features and AdaBoost to train cascaded classifiers, which achieves good performance with real-time efficiency.
Recently, Convolutional Neural Networks (CNNs) have made significant advances in various computer vision tasks such as image classification[9] and face recognition[10]. Inspired by the notable success of deep learning methods in computer vision tasks, some studies use deep CNN for face detection.
Zhang[11] proposed a new framework for integrating these two tasks with a single, cascaded CNN through multitasking. The face detection architecture consists of three stages. In the first stage, it quickly generates candidate windows through a shallow CNN. In second stage, the network refined the window by denying a large number of non-face windows through a more complex CNN. Finally, in third stage it uses the more powerful CNN to refine the results again and output five facial marker positions. Due to this multitasking learning framework, the performance of the algorithm can be significantly improved. Figure 1 shows the overall 3-stage architecture.

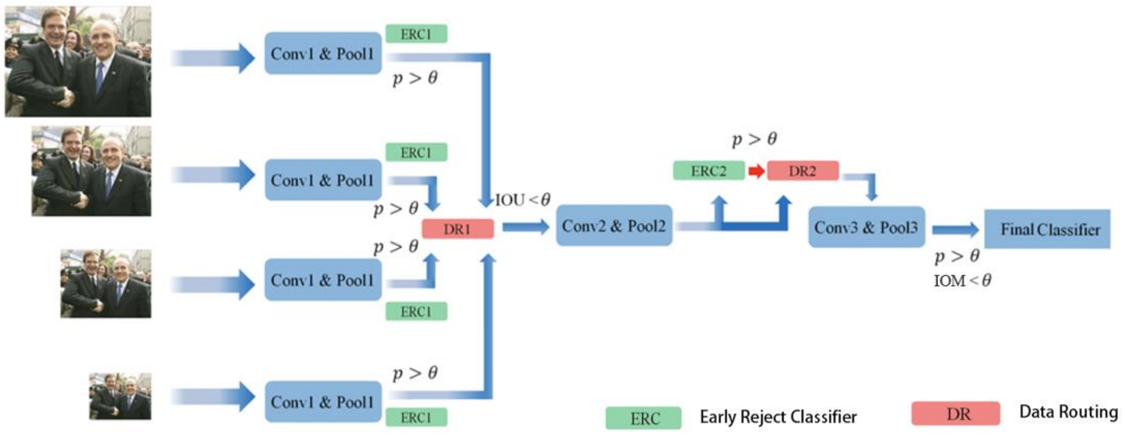


Figure 1

There are two parts in every stage: a neural network and several processes used to drop unnecessary bounding boxes. Zhang named these three neural networks as Proposal Net (P-Net), Refine Net (R-Net) and Output Net (O-Net). We will discuss the structures of three neural networks and box-dropping strategies in the further parts.

Proposal Net (CNN Structure, Input & Output)

P-Net(Proposal Net) is the Network used in the 1st stage. The P-Net is built by several Convolution Layers and max pooling Layers. The structure is shown in the Figure 2. The input size of the P-Net is the detection box through the pyramid detection which is 12x12 RGB image. The Pyramid structure will be explained in next section.

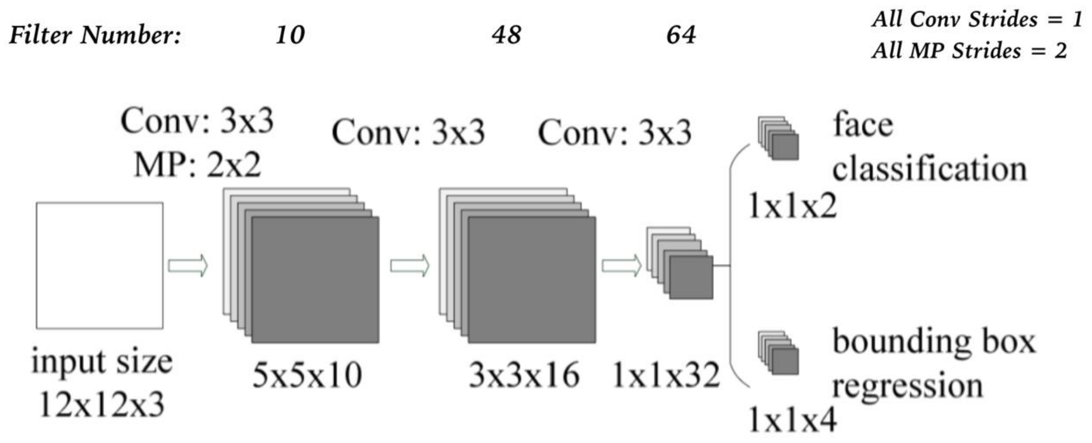


Figure 2

The Output is a size 2 list. The Output[1] is a N*4 array, the N means the number of the bounding box generating by the pyramid structure. The 4 elements are the bounding box regression parameters which are used to adjust the bounding position. The first 2 are for the width adjustment and the second 2 are for the height adjustment. The Output[2] is N*2 array which contain face classification score. The first element is the negative score and the second one is the positive score.

Refine Net (CNN Structure, Input & Output)

The R-Net(Refine Net) is the Network used in the second stage. For the input, we resize the boxed generated by previous stage to 24x24x3. The architecture is shown as the Figure 3 below.

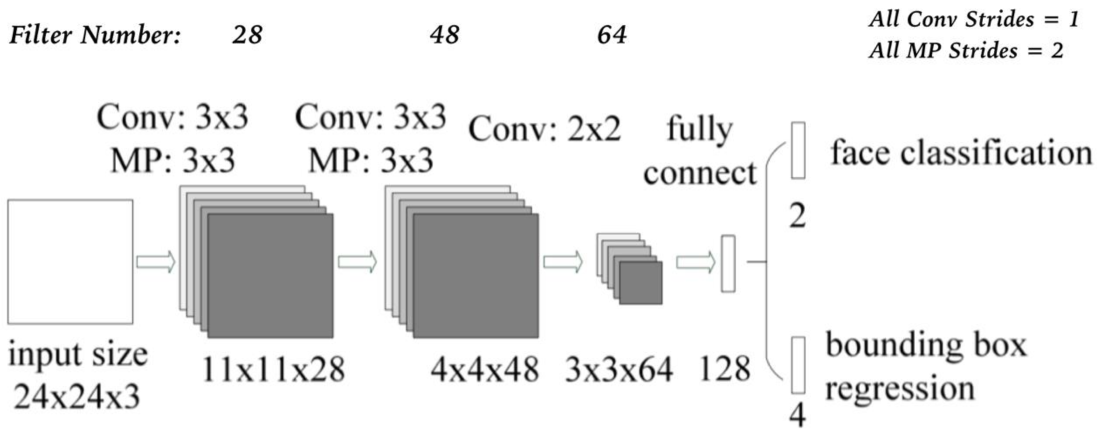


Figure 3

For the Output, it use two fully connected layer to generate face classification score and bounding box regression parameters. The Output is also a size 2 list, the content is same as the P-Net in previous stage.

Output Net (CNN Structure, Input & Output)

The O-Net(Output Net) is the network in the third stage. The structure of the O-Net is shown in the Figure 4 below. The input is 48x48x3 boxes generated by the previous stage. The architecture of O-Net is similar to the R-Net, the difference between these two networks is the output. Besides the face classification The O-Net output generate the landmark localization parameters. The localization parameters are used to adjust the five face joints point on the detected face.

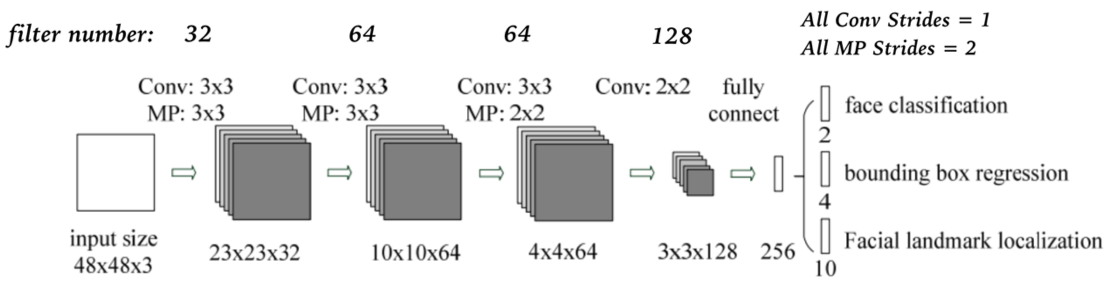


Figure 4

The output is size 3 list, the output[0] store 10 facial landmark localization parameters. The first five elements are for the width adjustment of the five face joint point. The Second five elements are for the height adjustment.

Train and Loss function

Training data

The training data is composed by four different kinds of data: Negative, Positive, Part Face and Landmark face. The ratio of the these four data sets is 3:1:1:2. To define which kind of the set the data is, we use the Intersection over Union (IOU) to do the judgement. The IOU will be explained in later section. For the Negative dataset, the IOU is less than 0.3 to any ground truth face, for the Positive data set, the IOU is above 0.65 to the ground truth face, for the Part face, the IOU is between 0.4 and 0.65 to the ground truth, and for the Landmark face, the face is labeled 5 landmark position.

Training and Loss function

The training in this architecture can be divided into three part, face classification, box regression and landmark localization.

Face classification:

For the face classification training, we cross entropy loss function, the loss function is often used for the classification model. The function is shown below

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

where P_i the probability produced by the network that indicates sample x_i being a face. The notation $y_i^{det} \in \{0,1\}$ denotes the ground-truth label. [11]

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:

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

The formula above basically means the distance between the regression target obtained from the network $\hat{y}_i^{box}$ and the ground-truth coordinate y_i^{box}. There are four coordinates, including left top, height and width.[11]

Facial Landmark localization training:

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

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

where $\hat{y}_i^{landmark}$ is the facial landmark's coordinates obtained from the network and $y_i^{landmark}$ is the ground-truth coordinate for the i-th sample. There are five facial landmarks, including left eye, right eye, nose, left mouth corner, and right mouth corner. [11]

Multi-Source Training

Since we employ different tasks in each CNN, 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 (Three loss function above) 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:

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

where N is the number of training samples and α_j denotes on the task importance. We use (α_{det}= 1, α_{box}= 0.5, $\alpha_{landmark}$ = 0.5) in P-Net and R-Net, while (α_{det}= 1, α_{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 these CNNs. [11]

Face Detection

Pyramid Structure

At the beginning, we use pyramid structure to resize the input image into different scales shown in Figure 5. And the scales would be computed by scale = 0.6 * 0.709^n.

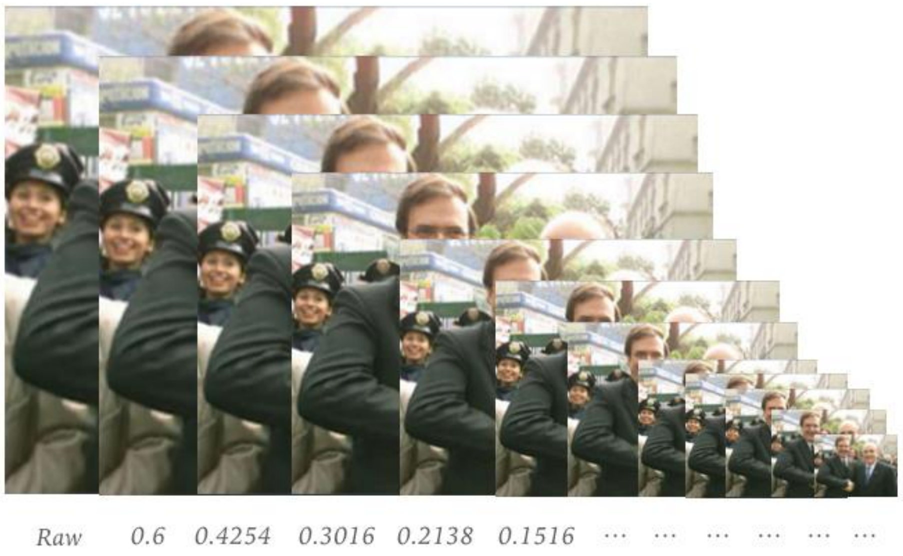


Figure 5

The reason to use pyramid structure is that the size of detection filter is 12*12*3. In order to obtain different size detection, in this model, we keep the size of detection filter and resize the input image. As the result, like in Figure 6, the ratio of detection filter to different scale images are different.

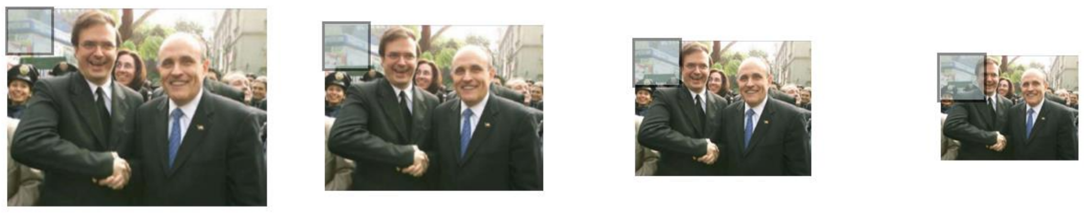


Figure 6

And after the detection process, we can obtain small size face in large scale image and large size face in small scale image like in Figure 7.

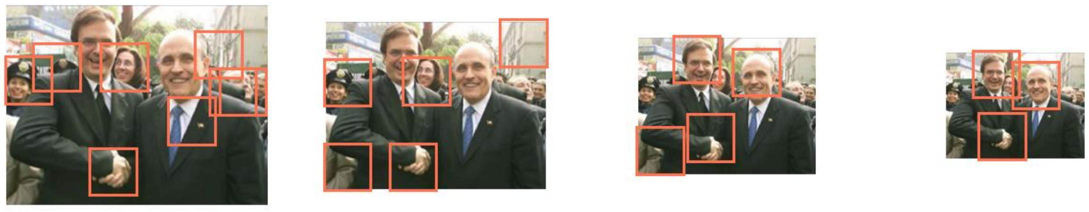
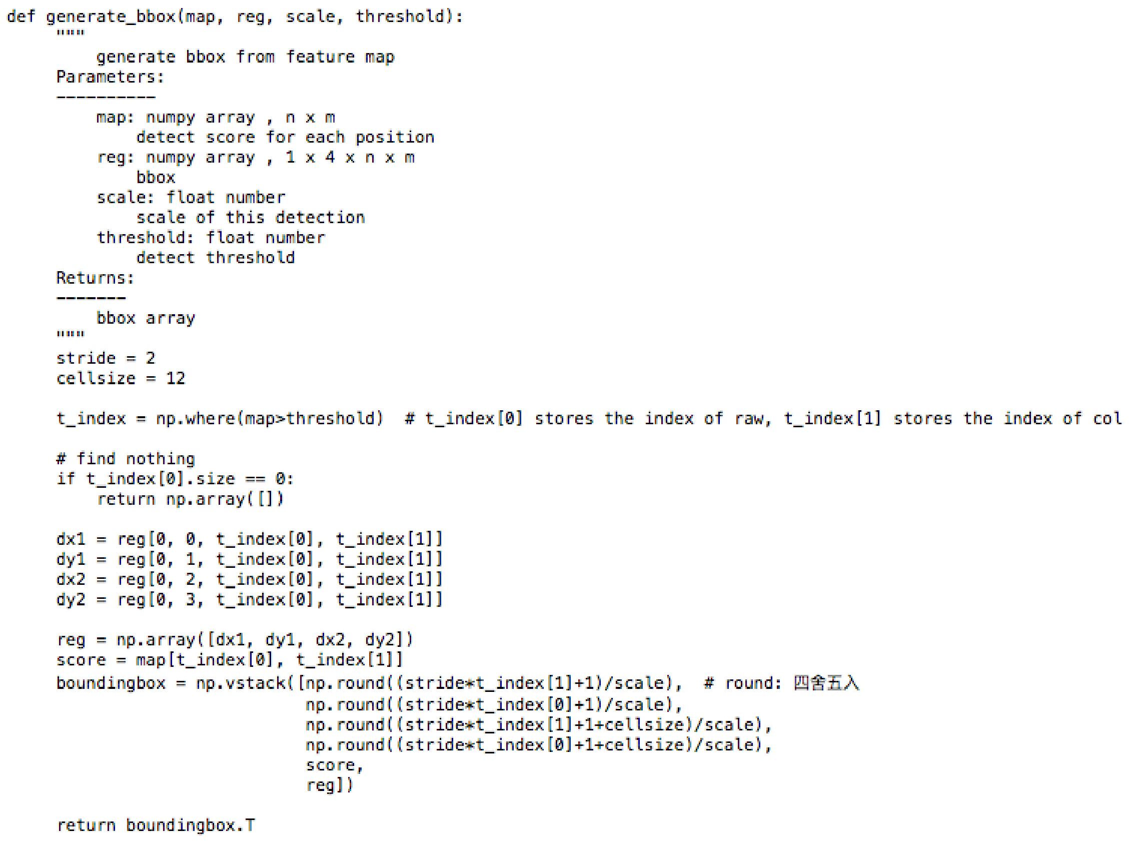


Figure 7

Generate Box

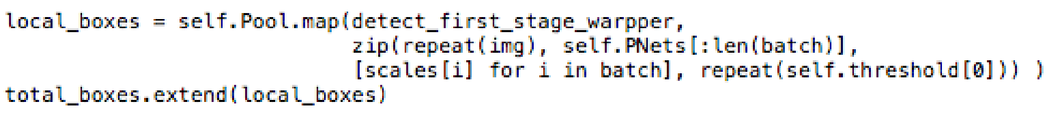
The code in Figure 8 shows the process of generating bounding box.



```
def generate_bbox(map, reg, scale, threshold):
    """
        generate bbox from feature map
    Parameters:
    ----------
        map: numpy array , n x m
            detect score for each position
        reg: numpy array , 1 x 4 x n x m
            bbox
        scale: float number
            scale of this detection
        threshold: float number
            detect threshold
    Returns:
    -------
        bbox array
    """
    stride = 2
    cellsize = 12

    t_index = np.where(map>threshold)  # t_index[0] stores the index of raw, t_index[1] stores the index of col

    # find nothing
    if t_index[0].size == 0:
        return np.array([])

    dx1 = reg[0, 0, t_index[0], t_index[1]]
    dy1 = reg[0, 1, t_index[0], t_index[1]]
    dx2 = reg[0, 2, t_index[0], t_index[1]]
    dy2 = reg[0, 3, t_index[0], t_index[1]]

    reg = np.array([dx1, dy1, dx2, dy2])
    score = map[t_index[0], t_index[1]]
    boundingbox = np.vstack([np.round((stride*t_index[1]+1)/scale),  # round: 四舍五入
                             np.round((stride*t_index[0]+1)/scale),
                             np.round((stride*t_index[1]+1+cellsize)/scale),
                             np.round((stride*t_index[0]+1+cellsize)/scale),
                             score,
                             reg])

    return boundingbox.T
```

Figure 8

The t_index in Figure 8 stores the order of bounding boxes and we can transfer it to the axis of left and upper boarders by formula *x1 = (index of raw * stride + 1) / scale* and *y1 = (index of col * stride + 1) / scale*. But for the right and lower boarders, we need add the width or height of detection filter. In our model, the size of detection filter is 12*12*3, so the x2 and y2 can be computed by *x2 = ((index of raw + 12) / stride + 1) / scale*, *y2 = ((index of col + 12) / stride + 1) / scale*. Then we need to group the regression value to adjust the boarders of bounding boxes.
The operations of "*divide scale*" in front formulas can recover the location and size of bounding boxes as before scaling. Finally, we merge all detections after generating process. The code segment has been given in Figure 9.



```
local_boxes = self.Pool.map(detect_first_stage_warpper,
                            zip(repeat(img), self.PNets[:len(batch)],
                            [scales[i] for i in batch], repeat(self.threshold[0])) )
total_boxes.extend(local_boxes)
```

Figure 9

Box Drop with Threshold and Non-Maximum Suppression (NMS)

We need drop unnecessary boxes after each neural network. Firstly, we drop the box with a score lower than the threshold. These boxes are considered as not containing any face. Furthermore, we need to drop the duplicated boxes by Non-Maximum Suppression strategy.

Non-maximum suppression (NMS) has been used as an intermediate step in many computer vision algorithms and is an integral part of many proposed approaches in detection, might it be edge, corner or object detection [12]. After we generate bounding boxes from each net, there will still exist some boxes which have high score and overlap with each other. It is necessarily to localize the concept of our interest, choosing serval representative boxes near the real location bounding regression. The process of NMS shows in Figure 10. For example, picture a is the output from R-Net, after the bounding boxes regression and NMS, we obtain the picture b.

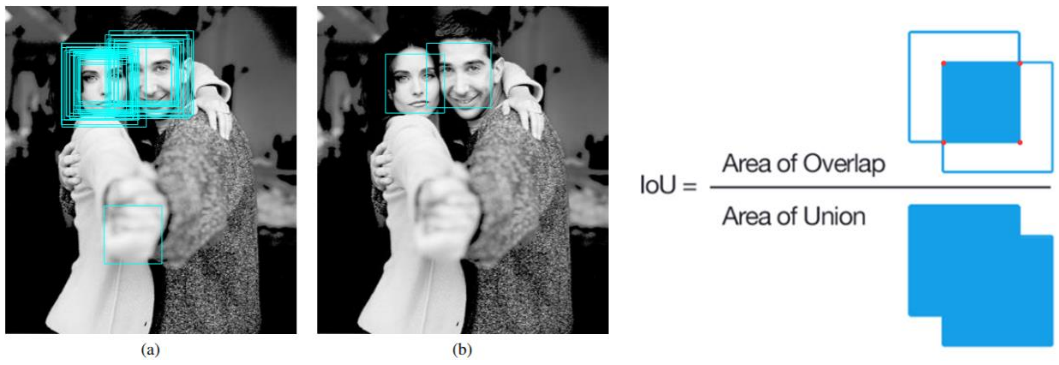
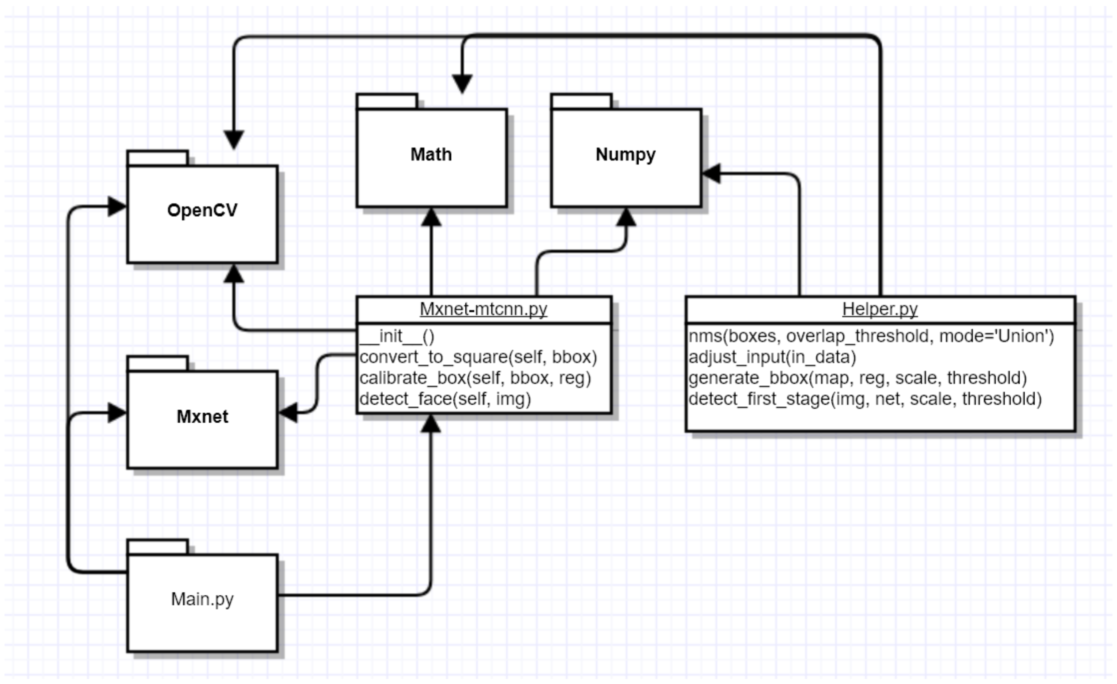


Figure 10[15]

Then let me talk about some details of the NMS process. The input of NMS function accepts boxes, threshold and model. To start of NMS, it selects the highest scoring box and suppose it indeed contains face object. Then, the highest scoring box calculate four red points in Figure 10[15] with each box to obtain the overlap area of them. It uses the format shows in Figure 10[15] to calculate the IOU of them and compare the IOU with threshold. The boxes that are too close to the selected window are suppressed. We pick the highest score boxes and select the next top-scoring one. This procedure is repeated until no more boxes remain. And when we do the NMS process in the last net, we use the IOM instead of IOU. We use their overlap area to divide by smaller area of them. We use this way to quickly reduce bigger portion of bounding boxes and generate precise box output for us. In generally, the NMS procedure involves defining a measure of similarity between boxes and setting a threshold for suppression.

Implementation (Structure of the Source Code)



Math
Numpy
OpenCV
Mxnet
Main.py
Mxnet-mtcnn.py
__init__()
convert_to_square(self, bbox)
calibrate_box(self, bbox, reg)
detect_face(self, img)
Helper.py
nms(boxes, overlap_threshold, mode='Union')
adjust_input(in_data)
generate_bbox(map, reg, scale, threshold)
detect_first_stage(img, net, scale, threshold)

Figure 11

As is shows on in Figure 11, our project mainly contains three python files, main file, Mxnet-mtcnn file and Helper file. Firstly, The main function is responsible for accepting input test image from user and invoke detect_face() function in Mxnet-mtcnn file to achieve the face detection operation. Secondly, The Mxnet-mtcnn file in the main file of our project. It imports Mxnet package to load the caffemodel from the root file and do the predict function. And it imports the Opencv package to preprocess the input images. In this file, it mainly uses the detect_face() function to do the three-stage face detection process.

In the detect_face() function, the first stage use the detect_first_stage_warpper() function in helper file to resize the input image in different scales and use P-Net to do the predict operation for the it. After the prediction, it uses the generate_bbox() function to drop the bounding boxes which has lower score than threshold and use the regression from the predict operation to adjust the bounding boxes. And then it use the nms() function in helper file to drop the overlap boxes which IOU is higher than 0.5. The second stage and the third stage is similar with the operation of the first stage in the code view.

Analysis

Hyper Parameter & Threshold

	Reduce bounding box by threshold score	Reduce overlapped bounding box by NMS
1st stage	score < 0.6	IOU > 0.5
2nd stage	score < 0.7	IOU > 0.7
3rd stage	score < 0.8	IOU > 0.7

Table 2

As Table 2 shows above, we have two types of threshold in our project. The first type is to reduce the lower score bounding boxes which implement after the boxes generation. In order to generally reduce the lower score boxes, we set the three stages threshold as 0.6, 0.7 and 0.8. The second type threshold is to help us reduce the overlapped bounding boxes which implement in the NMS process. As you can see, the bigger the number, the more boxes that I can drop. That's why we can finally output the precise boxes.

Memory Requirement & Computation Complexity

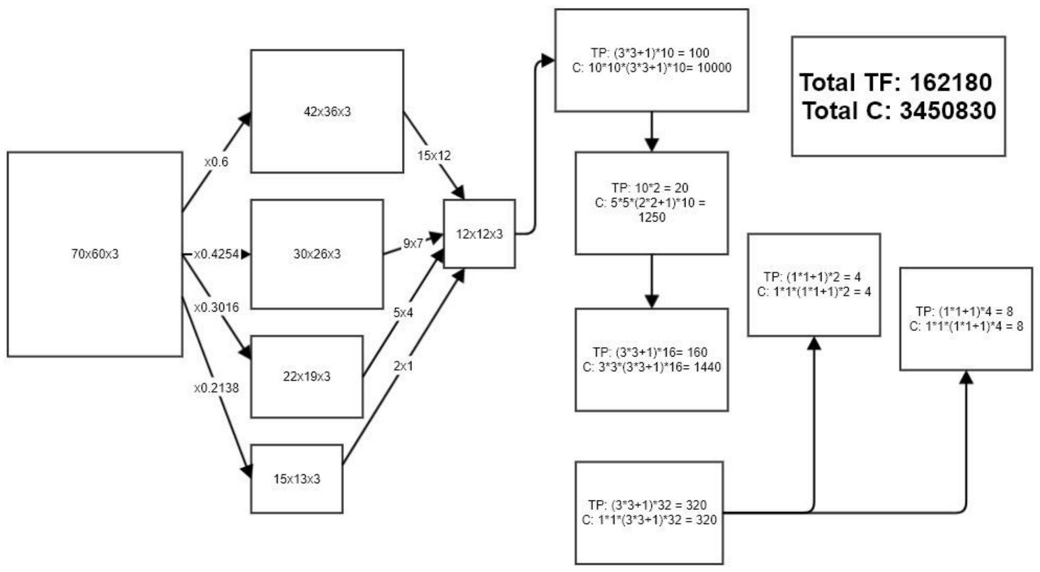


Figure 12

We use the total numbers of training parameters and connection to evaluate the memory and requirement and computation complexity of our project. As is shows in Figure 12, I use a 70x60 RGB face image as an example to calculate these two results. According to rescale process, it divides into four images which size is 42x36, 30x26, 22x19 and 15x3. After the filter of 12x12x3, P-Net receives 265 12x12x3 images as its input. The P-Net contains five convolution layers and one pooling layer. After the calculation of the whole process, the total train parameters are 162180 and the total connections are 3450830.

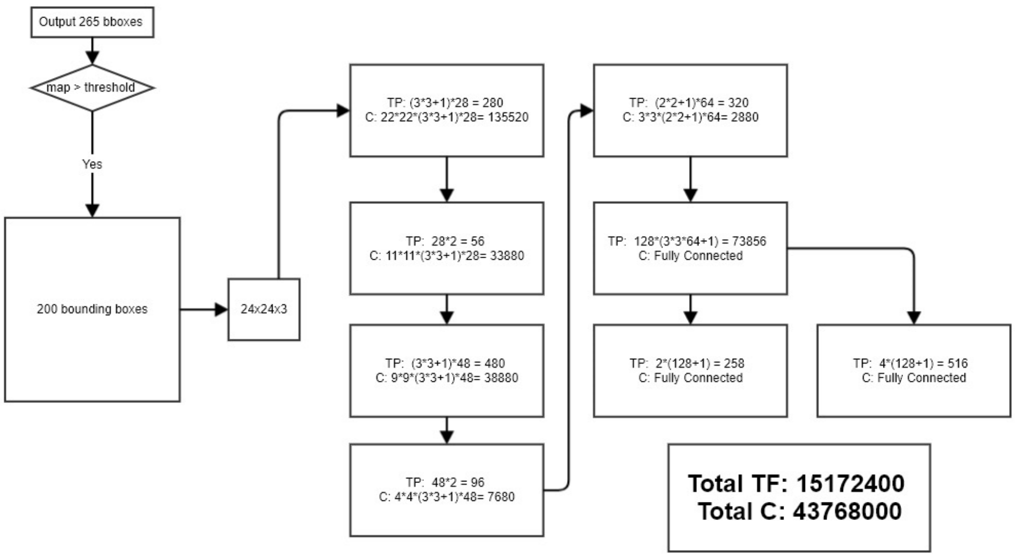


Figure 13

The Figure 13 shows the data process of R-Net. In our project, we do the bounding boxes regression and non-maximum suppression after each net. Thus, I suppose R-Net receives 200 bounding boxes after these processes. We use OpenCV to resize these boxes to the size of 24x24. The R-Net contains 3 convolution layers and 2 pooling layers and 3 fully connected layers. Because R-Net use fully connected layer at the end of its structure, the total training parameters is obviously bigger than P-Net which increase the computation complexity. And the filters in R-Net have more kernel size which create more connection between each layer. The total parameters is 15172400 and the total connection is 43768000.

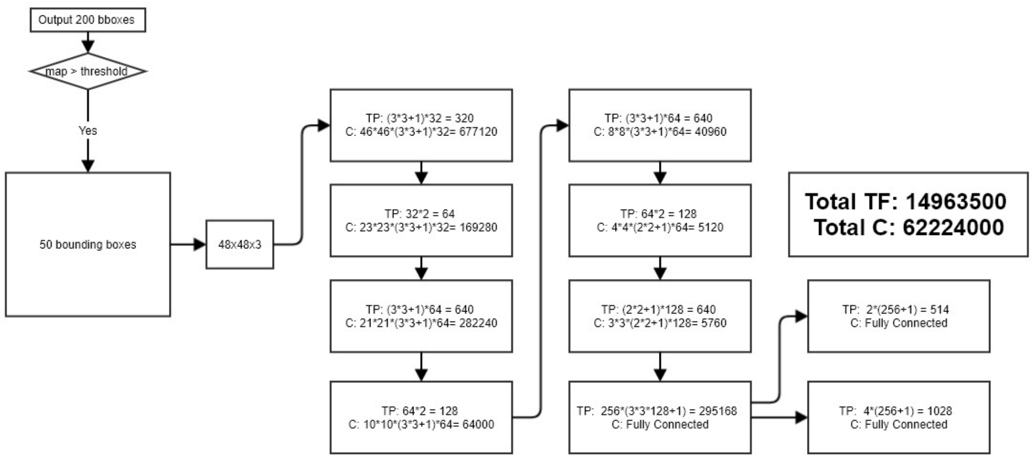


Figure 14

The dropping boxes process of R-Net is similar with P-Net. After this process, I suppose the number of input bounding boxes are 50 as shown in Figure 14. We need to generate the final output after O-Net which means each face will only have a precise box with it. That’s why O-Net is more complexity than P-Net and R-Net. It contains 4 convolution layers, 3 pooling layers and 3 fully connected layers. The total parameters in this net is 14963500 and the total connection is 62224000.

Improvement & Innovation

For the improvement part, we add an extra stage of a R net and one more bounding regression and NMS between the Stage 2 and 3 in the original 3-stage architecture. After this R net 2, we set a higher threshold than the R net 1 as 0.8.   Which can filer more incorrect box which also have high score in the original structure.

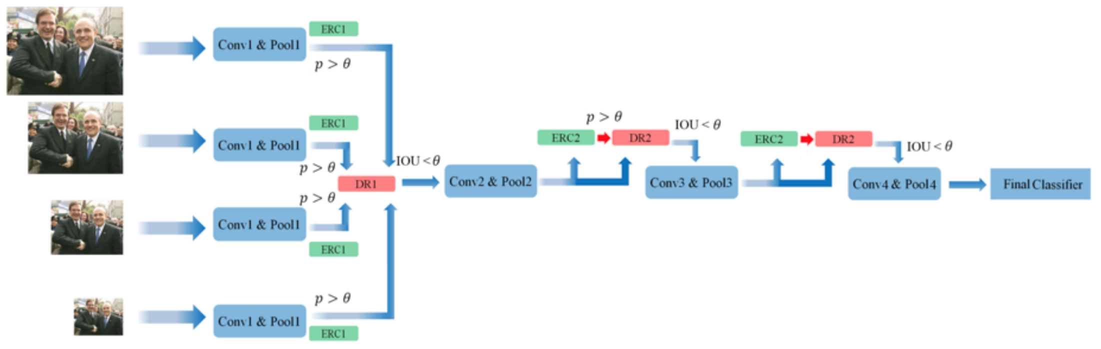


Figure 15

For example, we got a wrong box here in this Figure 16 Comparing the test result from original 3-Stage and improved 4-Stage. With our new architecture, we let high score box get higher score, and because of we have the 0.8 threshold after the R net 2, the wrong box has been filtered.

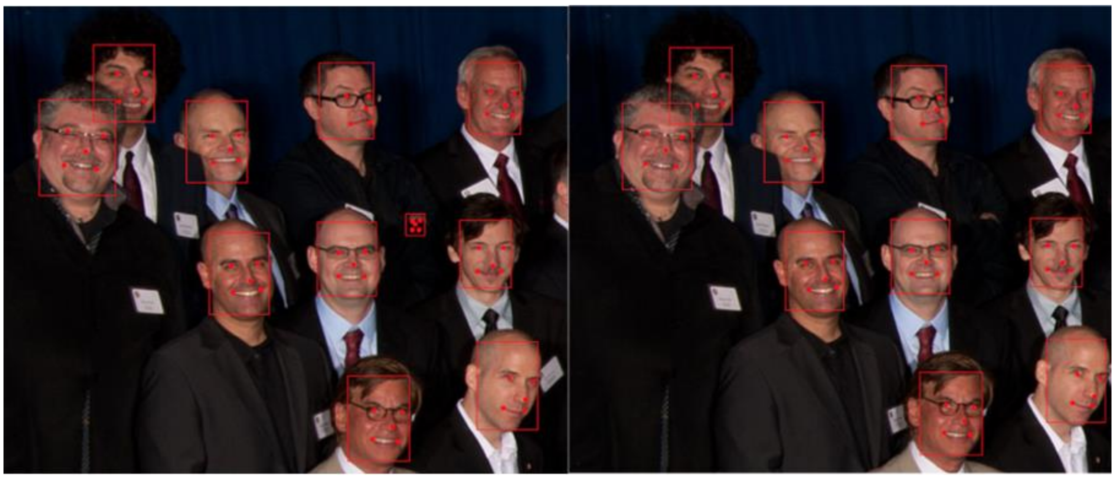


Figure 16 Comparing the test result from original 3-Stage and improved 4-Stage

For another example, we got two wrong boxes here in the up-right side of the Figure 17 Example 2 of Original Overall 3-Stage Test Result. With our new architecture, the wrong box has been filtered in the Figure 18 Example 2 of Improvement Overall 4-Stage Test Result.

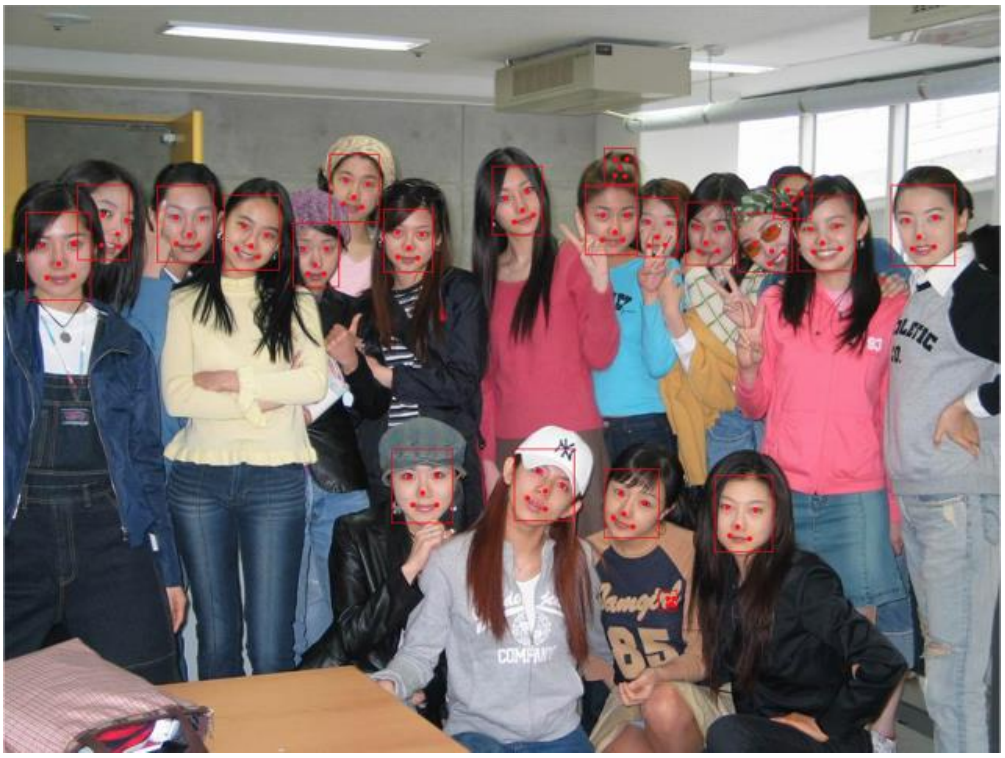


Figure 17 Example 2 of Original Overall 3-Stage Test Result

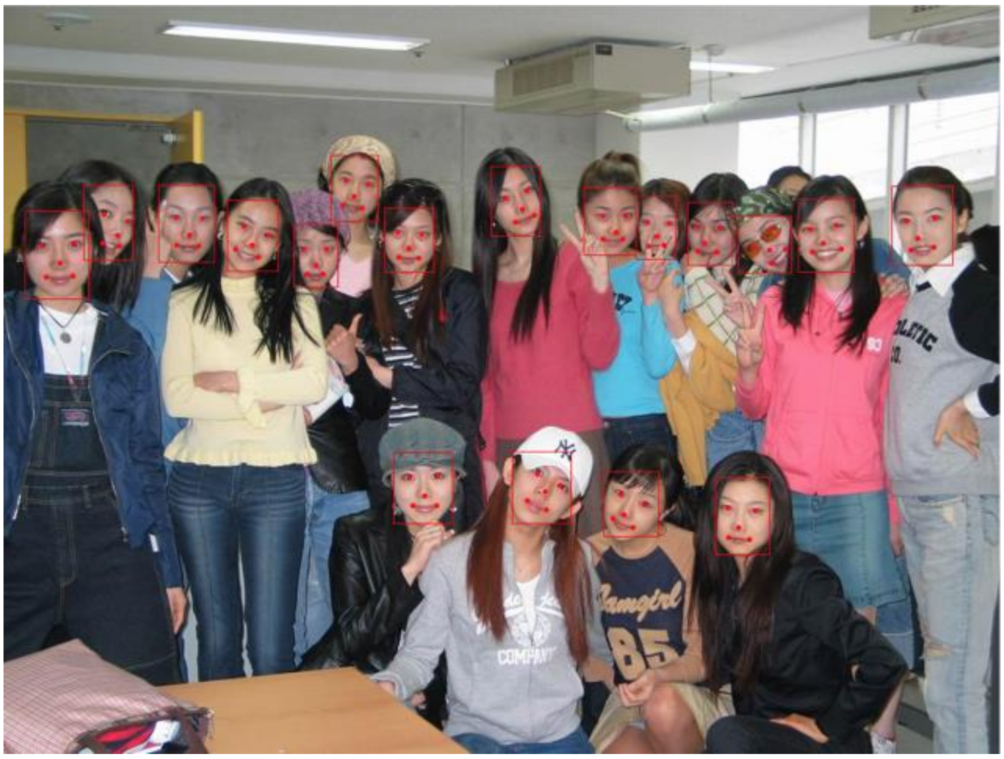


Figure 18 Example 2 of Improvement Overall 4-Stage Test Result

Our new model also can improve the accuracy in bad illumination. Figure 19 Comparing the test result from original 3-Stage and improved 4-Stage shows the comparison of our new improved model and the original doing the landmark localization when the input without good illumination.

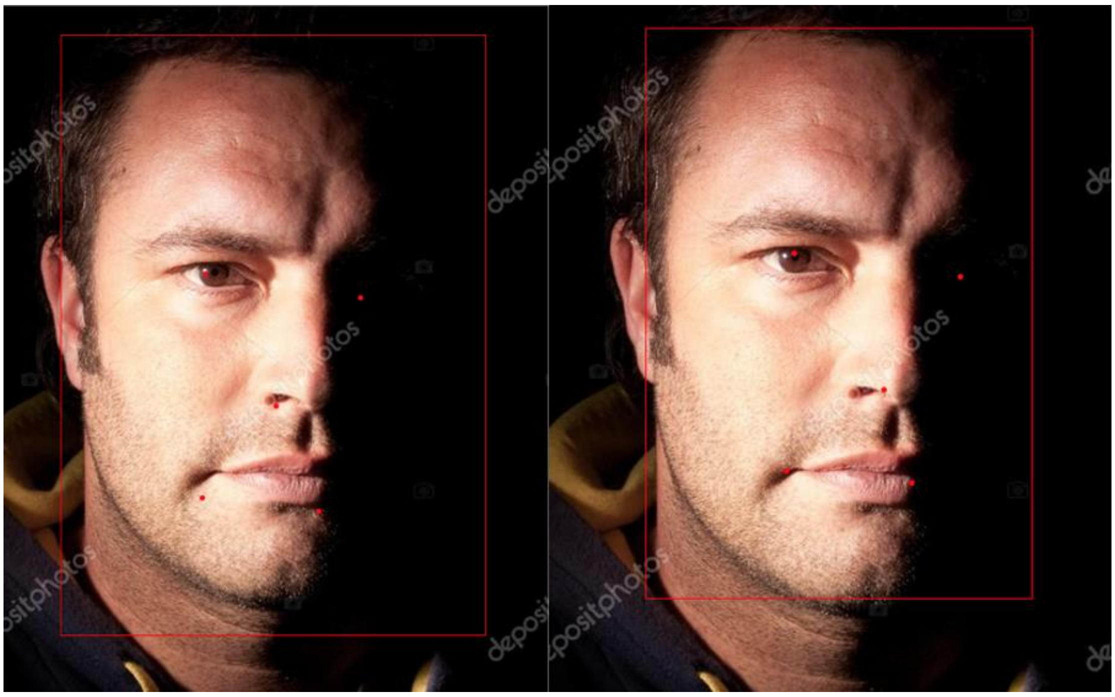


Figure 19 Comparing the test result from original 3-Stage and improved 4-Stage

From the Figure 15 we can see that the landmark has deviation in mouse and After our new R net 2, we have one more bounding box regression so let the adjustment of the box better. In this case, landmark localization could get better input from the bounding box regression so we get the more accurate landmark.

Demo

Here we demo how is the performance of our improved MTCCN.

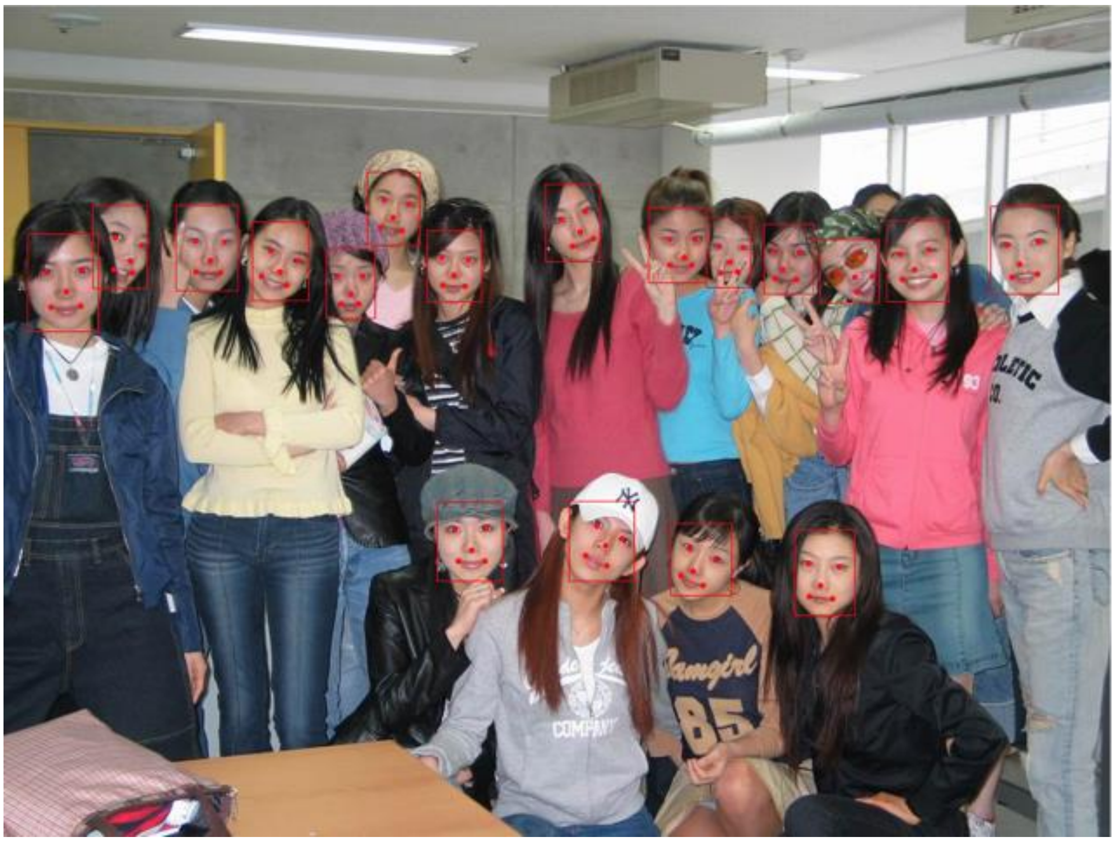


Figure 20 Demo of How MTCCN Doing the Face Detection

The result comes out within 0.053 second. Since the MTCCN is really fast, we can use it on other application. Such as Real Time Face Detection on camera.

Extend

We made a new application as real-time face detection on live camera. The computer's live camera supplying the different test data. The live demo has been showing in the Figure 21 Demo 1 of how MTCCN doing the real-time face detection and Figure 22 Demo 2 of how MTCCN doing the real-time face detection.

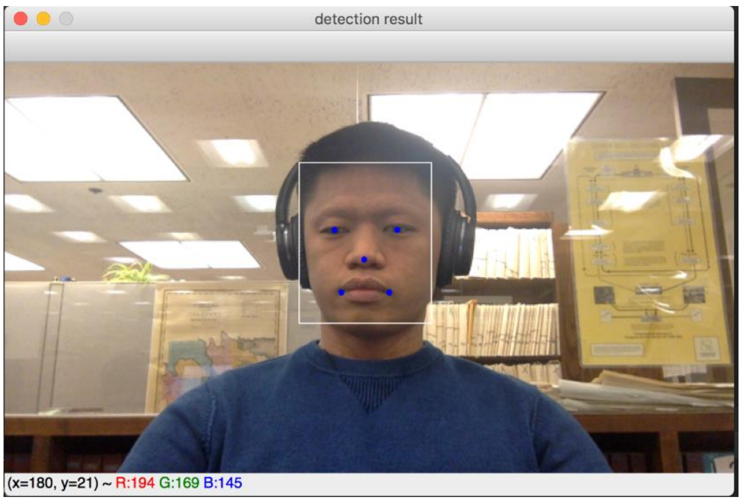


Figure 21 Demo 1 of how MTCCN doing the real-time face detection

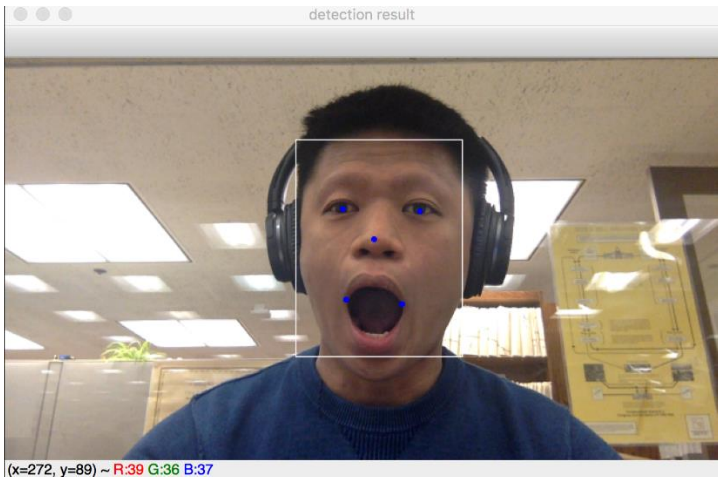
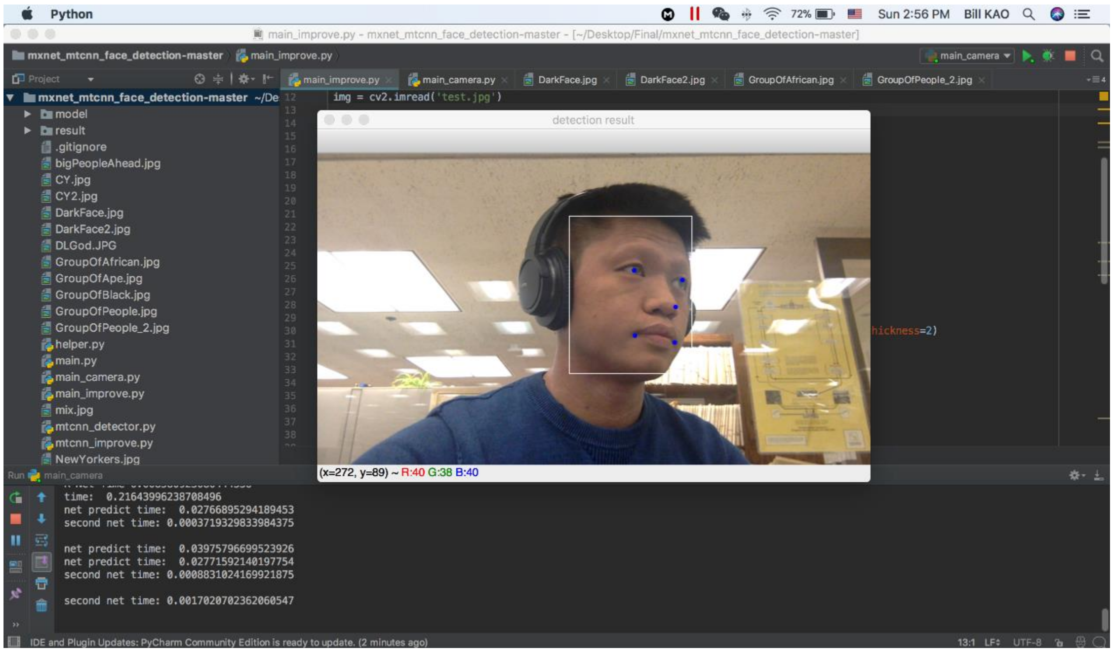


Figure 22 Demo 2 of how MTCCN doing the real-time face detection

Basically, we input every frame that camera received as a single image to the model and do the detection.

We can see that the MTCCN running really fast only with the CPU. So, we can see the multi thread in the first stage and the box reducing technic helps a lot for the performance.



```
img = cv2.imread('test.jpg')
```

```
time:  0.21643996238708496
net predict time:  0.027668952941894 53
second net time: 0.0003719329833984375

net predict time:  0.03975796699523926
net predict time:  0.02771592140197754
second net time: 0.0008831024169921875

second net time: 0.0017020702362060547
```

Figure 23 Demo 3 of How MTCCN Doing the Real-time Face Detection

The first is because, MTCCN has multiple thread to run the P net to train the model in the first step.

Secondly, in bounding box regression, it drops the box which have score below 0.6 in p net, 0.7 in r net 1, 0.8 in r net 2, 0.85 in o net. For another thing, it also drops the box which have 70% overlap which other box.

In this case, it’s significant Reduce the amount of box need to be computed.

Summary

The cascaded architecture is a 3 stage and contains P-Net, R-Net and O-Net in each stage. In every stage, we use threshold and NMS technique to do the bounding box reducing which lead to an improvement of the computation time. By doing so we can also achieve real time face detection based on a single CPU. For the improvement, we add an extra stage between the second and third stage and adjust the hypermeter of the threshold and NMS. By doing this, we achieve better accuracy for both bounding box and landmark localization and the detection efficiency is still acceptable.

Reference

[1] Pato, Joseph N., National Research Council (U.S.), Millett, Lynette I, Biometric Recognition: Challenges and Opportunities, pp. 1, 2010

[2] Datta, A., Datta, M., Banerjee, P. (2015). Face Detection and Recognition. New York: Chapman and Hall/CRC.

[3] A. K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. IEEE Tran. on Circuits and Systems for Video Technology, 14(1):20–26, 2004.

[4] Ajita Rattani, Fabio Roli, Eric Granger, “Adaptive Biometric Systems Recent Advances and Chanllenges”, pp. V, 2015

[5] Timo Ahonen, Student Member, IEEE, Abdenour Hadid, and Matti Pietika ̈inen, Senior Member, IEEE, “Face Description with Local Binary Patterns: Application to Face Recognition” IEEE Tran. on pattern Analysis and Machine Intelligence, Vol.28, No.12, December 2006

[6] Li, Stan Z., and Anil K. Jain (eds). Handbook of Face Recognition. Springer. © 2005. Books24x7

[7] Facelt-Hist. http://www.identix.com/company/comp_history.html

[8] P. Viola and M. J. Jones, “Robust real-time face detection. International journal of computer vision,” vol. 57, no. 2, pp. 137-154, 2004

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Image net classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097-1105.

[10] Y. Sun, Y. Chen, X. Wang, and X. Tang, “Deep learning face representation by joint identification-verification,” in Advances in Neural Information Processing Systems, 2014, pp. 1988-1996.

[11] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, Senior Member, IEEE, and Yu Qiao, Senior Member, IEEE, “Joint Face Detection and Alignment using Multi-Task Cascaded Convolutional Networks”, IEEE Signal Processing Letters Vol. 23, Issue 10, Oct. 2016

[12] Canny, J.: A computational approach to edge detection. TPAMI 8 (6) (1986) 679–698

[13] Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection.CVPR. (2005)

[14] Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. TPAMI 32 (9) (2010) 1627–1645

[15] C. K. Williams and W. Wojcikiewicz, “A principled approach to non-maximum suppression for object detection.”