Special Student Project

A Study of Face Segmentation, Recognition and Tracking

Author:
Zhenghao Chen

Supervisor:
Dr. Xiuying Wang
Dr. Jianlong Zhou

November 19, 2016

THE UNIVERSITY OF SYDNEY

Abstract

Serving as the one of most typical and problem that the 2D structure of face images normally has complicated dimensional for visual tasks. Researchers has been working in this vision problem for many years developing and evaluating a comprehensive model to accurately detect and recognize the human face in a state-of-the-art way. In this project, I would use three methods that EigenFace, Particle Swarm Optimization(PSO) and Convolution Neural Network(CNN) to solve this problem also after figuring out the relatively best method, I further develop an Android app to apply my face recognition method into face identification task in the real world. Overall a summary of face recognition methodology and implementation would be important step for further image classification task as an extension.

1 Introduction

Face Recognition is a very essential issue in the computer vision and artificial intelligence industry. The applications of that vary from security system to artificial robotics. The attempts for this problem started from very early that using basic Principal Component Analysis(PCA) that extract face 2D structure based on most outstanding eigen value of faces. Other methodologies including Support Vector Machine(SVM), Random Forest(RF) were also be implemented for this task. The biggest breakthrough recently is that Yann LeCun came up with Convolution Neural Net[2] that significantly enhanced the accuracy for human face[7] which provides a state-of-art way to address human face recognition. The follow-ups of this approach include Region-Based Convolution Neural Net[5] and so on. In this project, I attempted three ways to solve this problem. For one, the classical method Eigen Face based on PCA is going to be implemented, this method requires to train the large number of face images to compute the average of eigen values of each image in order to predict face of the test image . The second one is mainly for tracking that PSO, PSO is not able to distinguish the face from other object as it is not a classification method. However, once we select face as the target for PSO, PSO is able to update its particles to trace the face in real time therefore the PSO is mainly used for real-time video stream or web cam tracking human face. The last approach attempted is Convolution Neural Net. In this task, I do not just distinguish the human face. Instead, I try

to classify human face from other objects such as cat, dog etc. This method is very useful especially attempt it for comprehensive image. For instance, human riding horse, human with cats or dogs and so forth. Afterwards, an application that an human face classification app will be develop for mobile phone interface, this app use Convolution Neural Net as back-end and take Android as its interface to classify the human face to different person, by firstly use convolution neural net to identify the human face, and use certain distance method to compute the difference from training face to test face take the smallest difference value as target that identify the face to relative person. Overall this project has a review of different methods and have an extension of work.

2 Eigen Face for Human Face Identification

Eigenface[1] is one of the most significant approach to recognize the human face. The main approach to support this algorithm is Principle Component Analysis(PCA). In this section, I will have detail explain of PCA and how to implement it into eigen face.

2.1 Principal Component Analysis

To implement the Eigenface, PCA is needed to be applied to reduce the dimension of each entry in the training set. PCA is orthogonal linear transformation which is widely used to simplify the multidimensional data. It is an efficient method to find the most important elements and discard noisy and redundant data. In terms of linear algebra, PCA is to find a set of basis to describe the data dimension been generated and the new basis need to represent the relationship between original data (Jolliffe, 1986). Each faces contributes to generate each basis and these bases are contained in the covariance matrix. Hence, every image in the training set can be represented as a linear combination of each basis. These bases can then be used to calculate the weights of the training set and the test image to determine the distance. If the distance is in the range of the threshold, the test image will high likely to be a human face (Sirovich and Kirby, 1987).

2.2 Eigen Face

2.2.1 Data and Preprocessing

The train images used in this project are oxford VGG face data set [1] which provides 2622 celebrities in the dataset and the dataset is designed to have no overlap with the popular face recognition benchmarks such as Labeled Faces in the Wild (LFW) . Among these 259 faces including faces from both males and females. All train images have a resolution of 64*64 pixels, as multi-scale sliding window is required for this assignment, some pre-processing step has been done to generate the same faces but in a resolution of 40*40 pixels. When reading in the train images, the red channel of each pixel for a single train image is recorded as these images are already in the grayscale color mode. In addition, these images are stored in a matrix with the size of N^2 * M where N is the scale of the train images (either 40 or 64) and M is the number of input training images. The reason to do this is reducing the picture from 2D form to 1D matrix to ease the further calculation.

The data set look like below:

Figure 1: Training images

[1] http://www.robots.ox.ac.uk

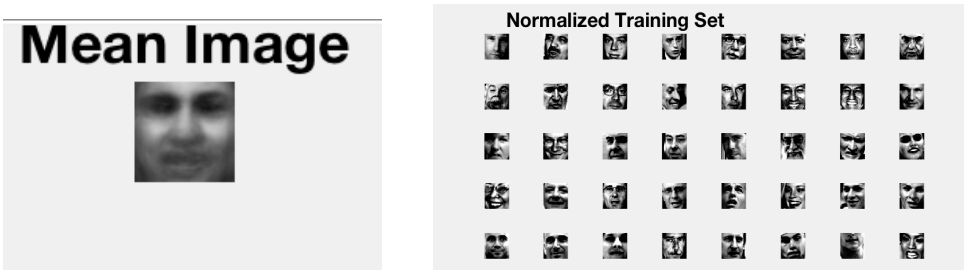
The structure how test images been stored as shown in the matrix below:

$$trainImage = \begin{bmatrix} a_1 & b_1 & c_1 & \dots & M_1 \\ a_2 & b_2 & c_2 & \dots & M_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N^2} & b_{N^2} & c_{N^2} & \dots & M_{N^2} \end{bmatrix}$$

2.2.2 Expect and Normalization

First, mean face has to be generated and been removed from each test entry because mean face is the common information shared by test images. The normalized train images can be generated by subtracting mean images from the original train images to have the distinguishing features kept.

$$mean = \frac{1}{M} \sum_{i=1}^{M} trainImages normImage = trainImages - meanImage$$



(a) mean face (b) normalized input images

Figure 2: Find mean image and normalized training images.

2.2.3 covariance matrix, eigen vectors and weight

Covariance matrix C = normImage · normImage^T where normImage is a N^2 * M matrix and normImage^T is a M * N^2 matrix. Hence the covariance matrix C is a N^2 * N^2 matrix. As N^2 * N^2 is a very high dimensional matrix, considering computational complexity, this matrix is too large to compute. So an alternative way is used during the implementation. Instead

of normImage $\cdot$ normImage^T, the normImage^T $\cdot$ normImage is considered in the real implementation.

Assume matrix A is the normImage matrix. If v is an eigenvector with nonzero eigenvalue of A^TA, then Av is an eigenvector with the same eigenvalue of AA^T. To prove the above proposition, the following equation can demonstatre the proving steps (Turk and Pentland, 1991).

$$(A^TA)Av = A(A^TA)v = A\lambda v = \lambda Av$$

Based on the proof above, eigen vectors u_i of the matrix AA^T can be calculated. u_i can be reverse converted to the matrix which have face like appearance, so u_i are also known as eigen faces. To make the processing easier, only the K most significant (has larger corresponding eigen values) eigen vectors have been picked for further calculation (Trivedi, 2009).

Each face in the training set now can be represented as a linear combination of the eigen vectors:

$$trainImage_i = \sum_{j=1}^{K} w_j u_j$$

Hence the weights can be represented as:

$$w_j = u_j^T trainImage_i$$

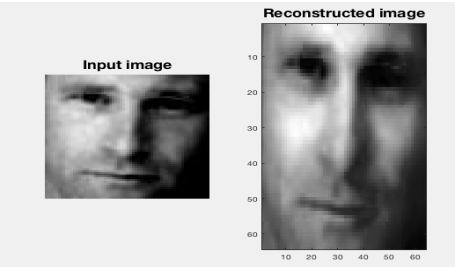
where w_j is a matrix with the size of K*1.

2.2.4 Test Result

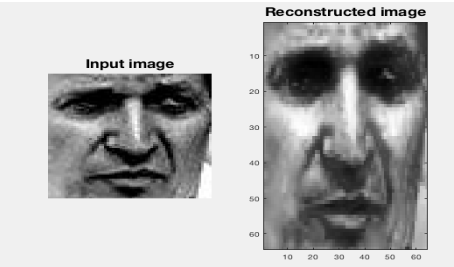
For an incoming test image, first step is to normalize it using the same step on the train images. Then calculate the weights as:

$$w_i = u_i^T normalizedTestImage$$

The Euclidean Distance between the train weights and test weights then be generated to determine if a test image is a human face. A threshold need to be defined to measure the distance. During the experiment, the threshold for 64*64 scale has been set to 650 and threshold for 40*40 scale has been set to 380.

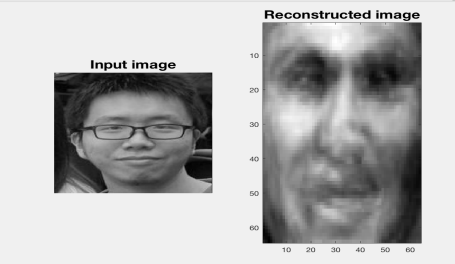


(a) Face reconstruction on train image 1

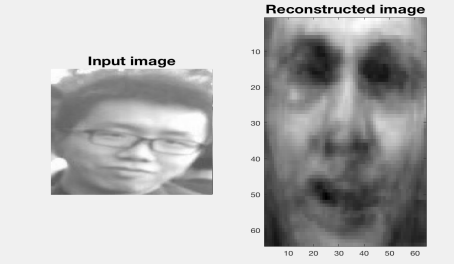


(b) Face reconstruction on train image 2

Figure 3: Face reconstruction on train image



(a) Face reconstruction on test image 1



(b) Face reconstruction on test image 2

Figure 4: Face reconstruction on test image

Figure 5: Eigen faces

2.2.5 Multi-scale Windows for Multiple Faces Recognition

For detecting multiple faces in one image. We need to use multi-scale window technique. Specifically, different scales of faces classification model will be

trained to deal with complicated size problem of faces in one image.
Figure 3.3, it shows that the the multiple faces recognition in the famous photo that also a is challenge for face detection benchmark.

Figure 6: Recognition result on the image of Solvey Conference 1927.

2.3 Summary and other Work

The main advantage of EigenFace is that it is probably the simplest method for face recognition. It utilizes the Eigen values of each training face matrix and assigns different weights when we compute the result for new faces. The complexity of this method is mainly concerned on dimensional reduction of each face matrix and scanning all eigen vector of faces to find some nearest faces. Therefore this method would support the real time face tracking as its fast computational time.

However there is a problem that as using PCA would be robust to compute the face classifier therefore this method is very sensitive to the picture that when it comes with complex image that human with other things(animal, vehicles, etc).

Regarding some other works, openCV has a good example open source to present the eigen face to detect the face in real time video. It uses JAVA JFRAME as an interface to track the face in the real time webcam or Video.

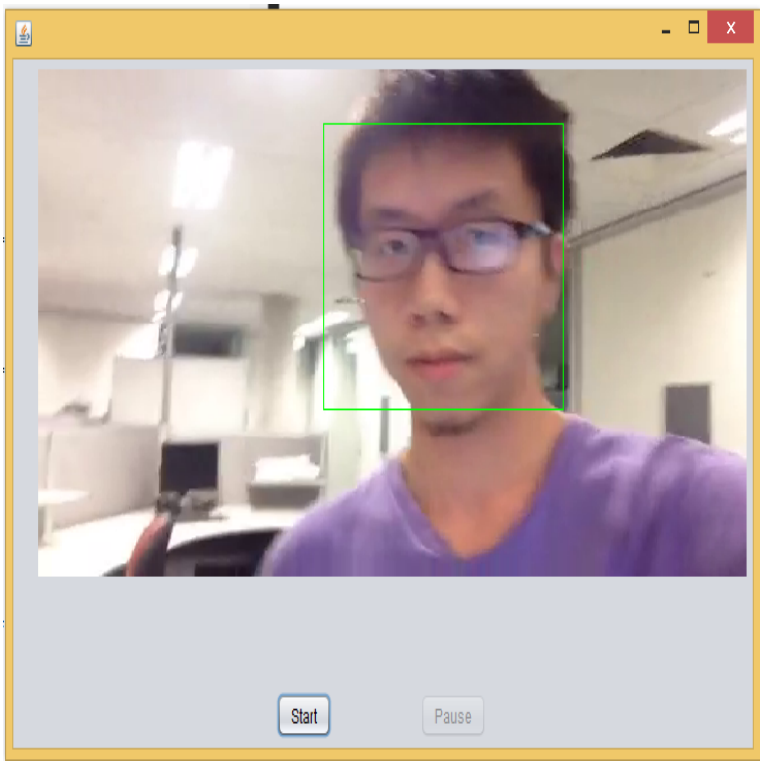


Figure 7: The real time eigen face example in openCV

3 Particle Swarm Optimization for Real Time Video Human Face Tracking

Particle Swarm Optimization(PSO) was firstly developed by Kenndy and Eberhart based on the social behavior of birds hunting[6]. Generally, this algorithm aims to search the best solution(Global Best Particle or Local Best Particle) of certain target with different candidate solutions(particles).

3.1 PSO Framework

In any particular image, all particles are initially created with 2D vector coordinates of this digital image. the center of a search window is geo-coordinator of the particle. All search windows are considered particle which

are the candidates solutions of final result that is the global best. Fitness values are evaluated by a fitness function. The particles will converge to the optimum and particle with the largest fitness value will be chosen as the solution (global best). Continuing updates will apply to renew their positions in the search space.

Specifically for every single particle, it would have 2 geo-relevant parametric that its position and velocity. After updating, the new position will be generated by summing the old position and velocity while velocity is going to be further updated based on new position obtained.

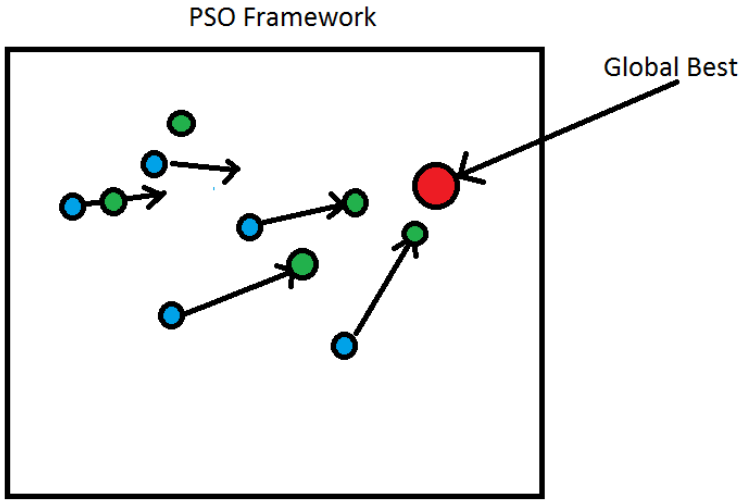


Figure 8: PSO Framework

Above figure 4.3 illustrates how PSO algorithm work, the red point is the global best that all the new candidate particle tracing for while the green point is the local best that represent the best solution in each iteration for each particle after updated. The update function can be summarized as:

$$v_n^{t+1} = w * v_n^t + C_1 * R_1(Pbest_n^t - x_n^t) + C_2 * R_2 * (Gbest^t - x_n^t)$$

$$x_n^{t+1} = x_n^t + v_n^{t+1}$$

Where w is the weight, $C_1, R1, C2, R2$ are constant and random number and $Pbest$ and $Gbest$ represent local best and global best.

3.2 Fitness Function and Weight Assignment

From previous section, we see that $Pbest_n^t - x_n^t$ and $Gbest_n^t - x_n^t$ that indicate the distance of global best and local best with particle being updated respectively, we define it as fitness function [9] which means how this particle fit the solution. For setting up this function, we typically have 2 classical methods that histogram and pure matrix distance.

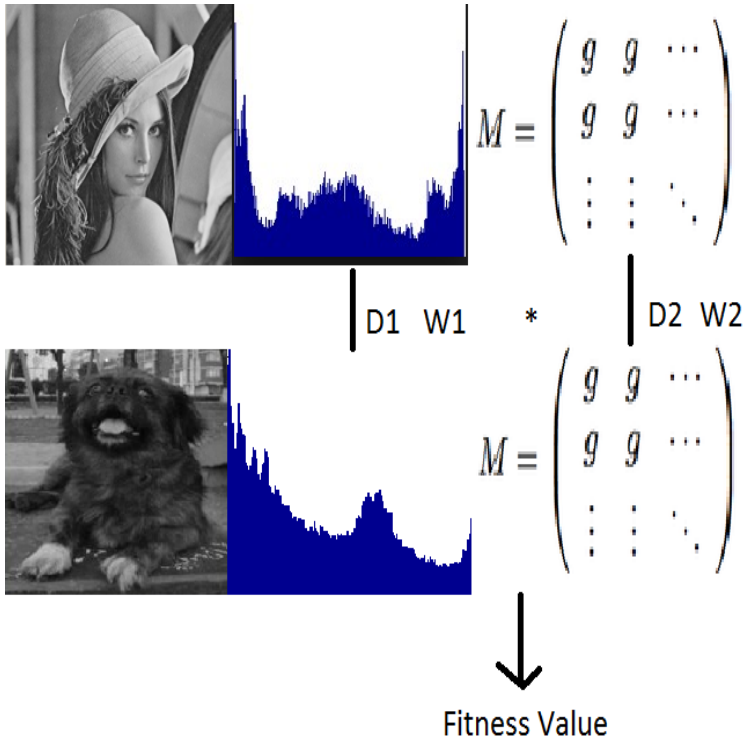


Figure 9: Fitness Function

Setting up the distance function can be another topic to discuss. Matrix distance can guarantee the fast speed as its $O(n)$ linear complex however, it would not always to be accurate for instance when face expression change(from cry to smile) the matrix distance would change correspondingly

even the face is belong to the identical person. On the other hand, Histogram would less likely to have this issue but might cause more running time. Therefore these two methods both have their strength and weakness. To choose which to be utilized really depends the real-world case.

3.3 Pipe Line

The pile line for transferring from image tracking to video tracking can be summarized as follow. This work is done by me and Dr Guang Liu in 2016[9].

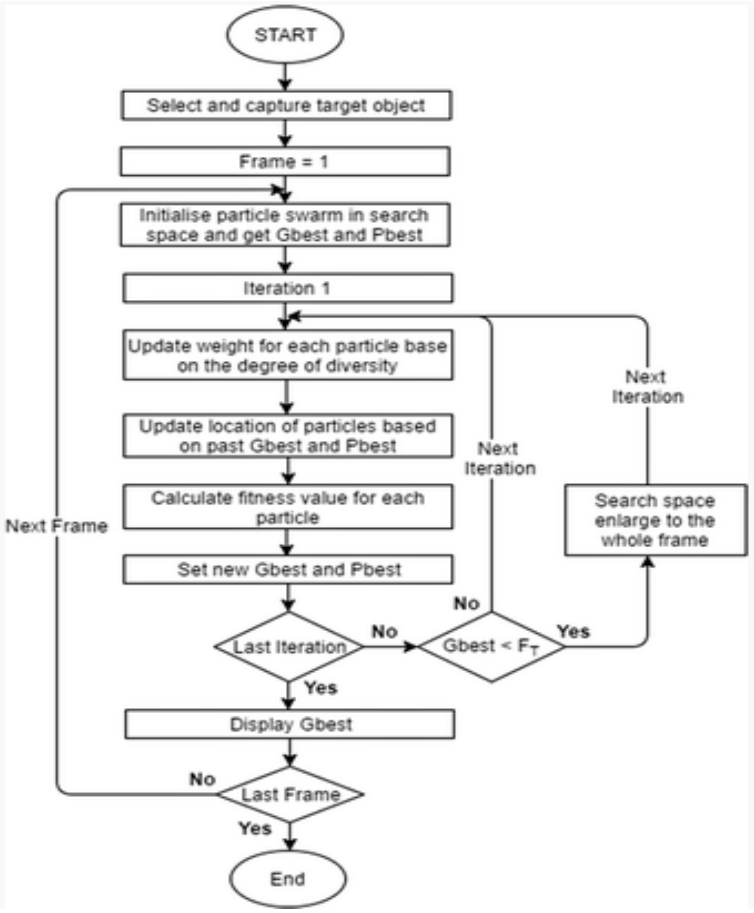


Figure 10: PSO pipeline

3.4 Experiment Result

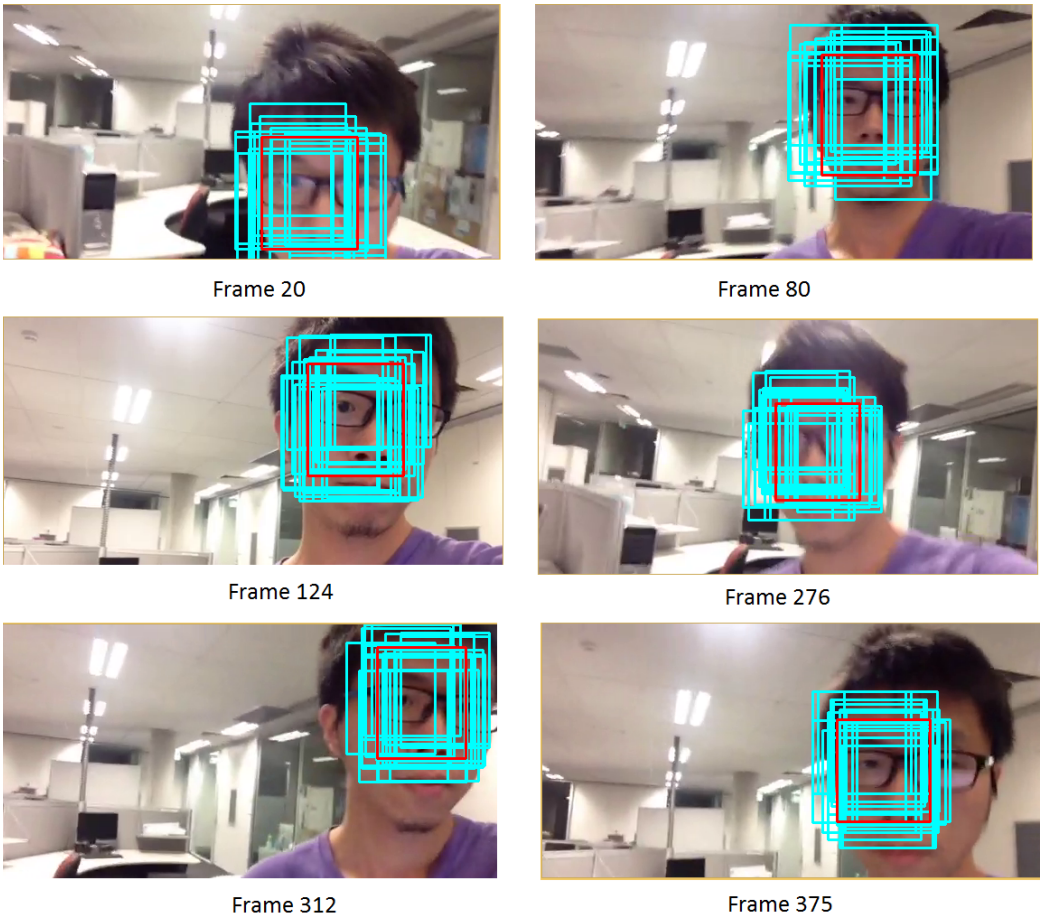


Figure 11: PSO Experiment

As shown, the result of tracking human face that using PSO is quite accurate. To save the space in this report, I only plot the 6 results(frames). In those frame, we can see that the face is mostly targeted by the red rectangle representing the global best solution for tracing while blue rectangles are candidate particles, which help to locate and target on solution.

3.5 Discussion of PSO

Comparing to other methods, PSO is a relatively easy methods, it does not require a trained model to perform the job as it is an inference approach. There is important to thing to note here which is that PSO cannot identify the human face by its own as it is not used for recognition but tracing. The main job that PSO can perform is to track face in real time. So furthermore, except the human face, this algorithm can also be used in the other moving object tracking in other video sets. I have also attempted to trace other objects using PSO, which I would like to discuss in following section.

4 Convolution Neural Net for Image Classification

This section will introduce the deep learning way that Convolution Neural Net, and how I implement CNN for human face recognition. Convolution Neural Net is able to classify the human face in the picture and have excellent performance. But in the case of image with multiple faces or some other objects. Single Convolution Neural Net won't be able to properly detect and segment the faces. Therefore a segmentation before classification would be necessary.To do this, I will use Region-Based Convolution Neural Net(RCNN). I will give detail explain in following section.

4.1 Convolution Neural Net

4.1.1 Convolution Neural Network

By mathematical definition, convolution is a function derived from two given functions by integration that expresses how the shape of one is modified by the other[3]. In computer graphics theory, convolution processing normally serves as a linear filer to sharp the image that outline the features.

Convolution Neural Network is essentially nothing different from the deep neural network but take each neuron as feature map that convolution layer and use convolution filter as its weight.

Forward Activation of CNN is essentially to take convolution processing of each feature map to produce the activation map. In this stage the activation function like Sigmoid and Tanh (shown as follow) would be applied to translate the each pixel of activation map. After that, the new feature maps would be constructed by summing up all activation maps of previous layer. In this stage, a sub-sampling technique(mostly max pooling [8]) would be applied that for one to shrink the map and for another sharpen the features of image.

$$f(x) = f(\sum w_i x_i)$$

$$f(x) = sigmoid(x) = \frac{1}{1 + e^{-x}}$$

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

Backward Propagation of CNN is basically the same as DNN but instead of simply update scalar value weight, CNN will update the convolution kernal of each neuron by back-propagation. That the filters would learn and artificially know what features of image should be highlighted by training.

$$w = w + \eta o_j(1 - o_j)(y_j - o_j)x_j$$

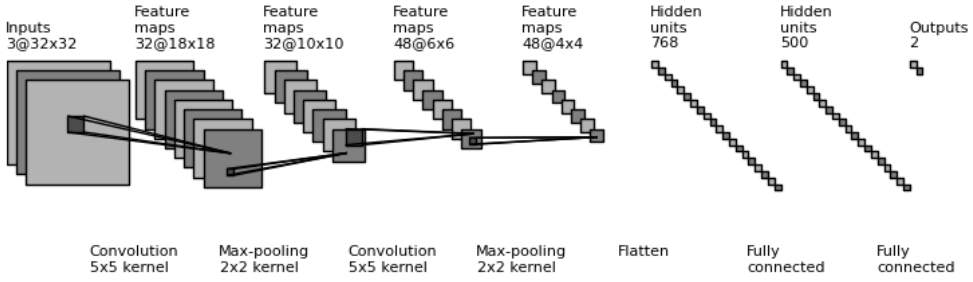


Figure 12: CNN Architecture

Architecture of CNN is hwon as above, the feature maps in each convolution layer there is a convolution kernal to take convolution processing to

generate activation map and by sub-sampling, the new feature map would be completed. As taking advantages of the 2D-matrix structure of an image or other 2D-formatted input such as a speech signal and so forth, the architecture of a CNN is designed achieved with local connections and tied weights followed by some form of down-sampling which enables translation invariant features.

4.2 Region-based Convolution Neural Net

The Region-based Convolution Neural Net[5] composes with two jobs that for one thing, the detection and segmentation of objects in an image. This step is called region proposed, it uses selective search that exact approach could be chosen from SIFT(Bags of Words) or just simple Color Histogram segmentation. For another, for each object, a Convolution Neural will be utilized to classify each object in image.

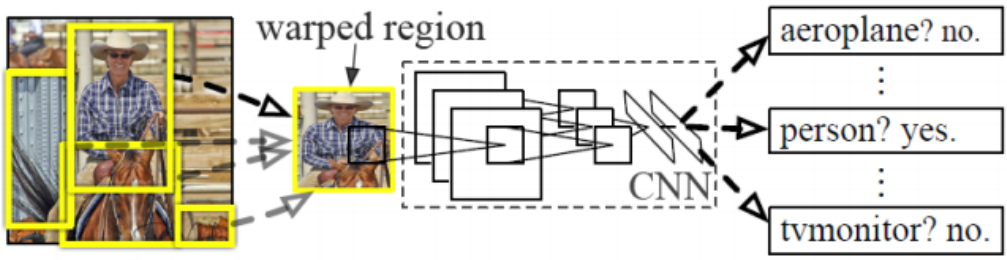


Figure 13: RCNN Propose[5]

The above figure shows the the work flow of RCNN. Clearly RCNN firstly make object search and then perform Convolution Neural Net for each object.

4.3 Data

The data set for training is from ImageNet[2], I downloaded 300 images from imageNet, labeled them as human face and others. To ensure the accuracy of experiment, I did not use too may labels there are only 9 labels I used, human, TV, cat, dog, horse, car, bus, fish and pig. The proportion distribution of each class is shown as below

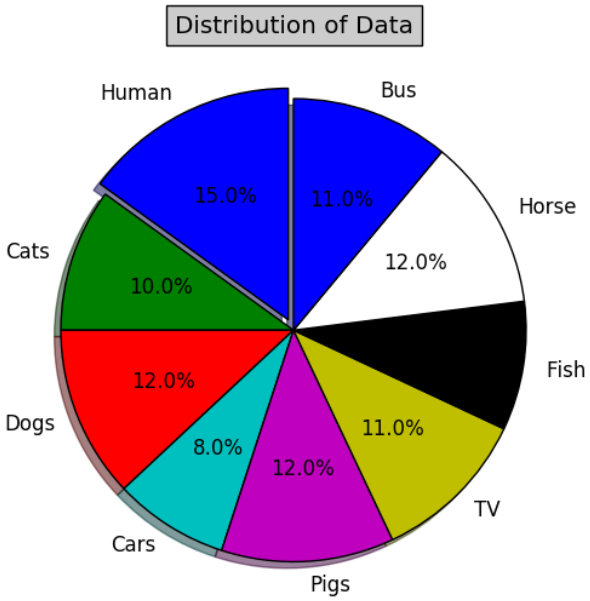


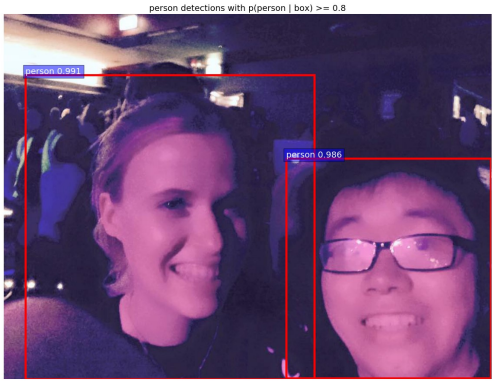
Figure 14: Distribution of Data Set

There are around 15% of images are label as human for training. This task is mainly focus on distinguish the human from other class.

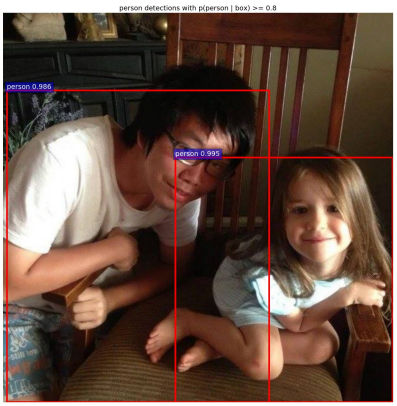
4.4 Experiment Result

The test data set is my personal photos with multiple faces. To see whether this approach is able to segment and recognize the human face.

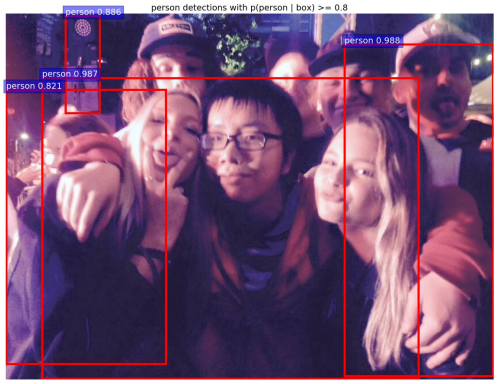
[2] `http://image-net.org/`



(a) Test Result 1



(b) Test Result 2



(c) Test Result 3

Figure 15: RCNN Experiment

Those results clearly show that the method RCNN can perform segmentation and classification with an excellent job. It gives the region in a reasonable way and which would not cause too much computational time for running CNN followed by.

4.5 Discussion of CNN

Convolution Neural Net serves as a state-of-art way of image learning and classification technique with very obvious advantage. It is able to learn and generate a powerful mode dealing with large data set. The only problem is that it requires a large training computational time which means we need to have a outstanding machine to perform CNN.

As an Extension of CNN, RCNN combines image segmentation technique and CNN, which results in that smaller sub-image will be input into CNN reducing the work of CNN and enhance the classification accuracy. The interesting discussion is on the selective search part that the scale of detected object , as mention before SIFT and Color Histogram can be good choice to perform selective search. In addition, researchers are still working on finding a better way to reduce the running time and have more reasonable segmented region.

5 An application, Mobile App for Face Classification

By summarizing all the useful methods I end up with implementing CNN in a mobile face recognition and information retrieval app, this app is able to recognize the face from a photo and classify the face to the corresponding person in data base based on feature match distance. Afterwards, it will retrieve the information of this person and display it.

5.1 Back-End - CNN with Triplet Loss function

The back-end of this app is using CNN with an improved method on loss function called Triplet Loss Function[4]. The Triplet loss is to constraint the convergence of the CNN algorithm so that an input image can be represented by a d-dimensional Euclidean space vector. The aim of the triplet loss is to ensure that the images of the same person get the similar representations while the images of different people get the distant representations. As a

result, the constraint is given by:

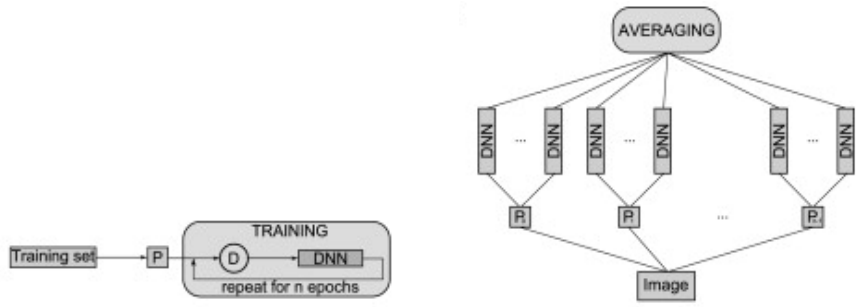
$$||x_i^a - x_i^p||_2^2 + \alpha < ||x_i^a - x_i^n||_2^2, \forall (x_i^a, x_i^p, x_i^n) \in T$$

where α is the margin between positive and negative sides and T represents the set of all possible triplet.

What we want to maximize is given by:

$$\sum_i^N [||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha]$$

The architecture of CNN with Triplet Loss is shown as below



(a) Training a CNN: during training process, all original images are randomly distorted before each epoch D

(b) MCDNN architecture: an arbitrary number of columns of the input image is trained during each epoch, the final prediction is based on the average of the prediction of each CNN

Figure 16: CNN with TripleLet Loss Function

After training, the model will be saved on the server and all the input image will be worked through this model to test whether it contains faces.

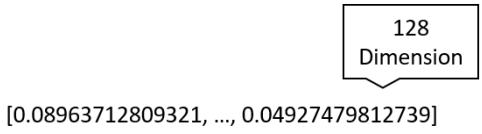
5.2 Work Flow - Android Development Process

The mobile interface of this work is using Android, the main work flow this program is that, first of all, user take photos or upload a photo. Then photo

uploaded will be directed to server. The picture will be transformed into 128-dimensional vectors as below figure shows, the representation vectors are then put in the training model. When the server receives an input picture, the same algorithm will be applied to the input picture to get the representation vector, this vector is then compared to the representation vectors of all the training pictures. Eventually, the paths to the pictures who have the most similar vectors will be return by the server.

(a) Original Picture

(b) Face



[0.08963712809321, ..., 0.04927479812739]

(c) Representation Vector

Figure 17: Implementation

The database of this app also has essential job that storing all the information of each person with person name is the label of face images. In this app, the distance method will use matrix distance. As faces photos are all represent by a 128 - dimensions Vector, therefore the comparing process would not take much time and space. Due to privacy issue, our training set only get the pictures from 6 people including me, Dr Xiuying, Dr Jianlong, Hui Cui, and two celebrities Barack Obama and Donald Trump. The pictures of Obama and Trump are from Google, while other images are collected manually. The information only include job, name facebook and linkedin.

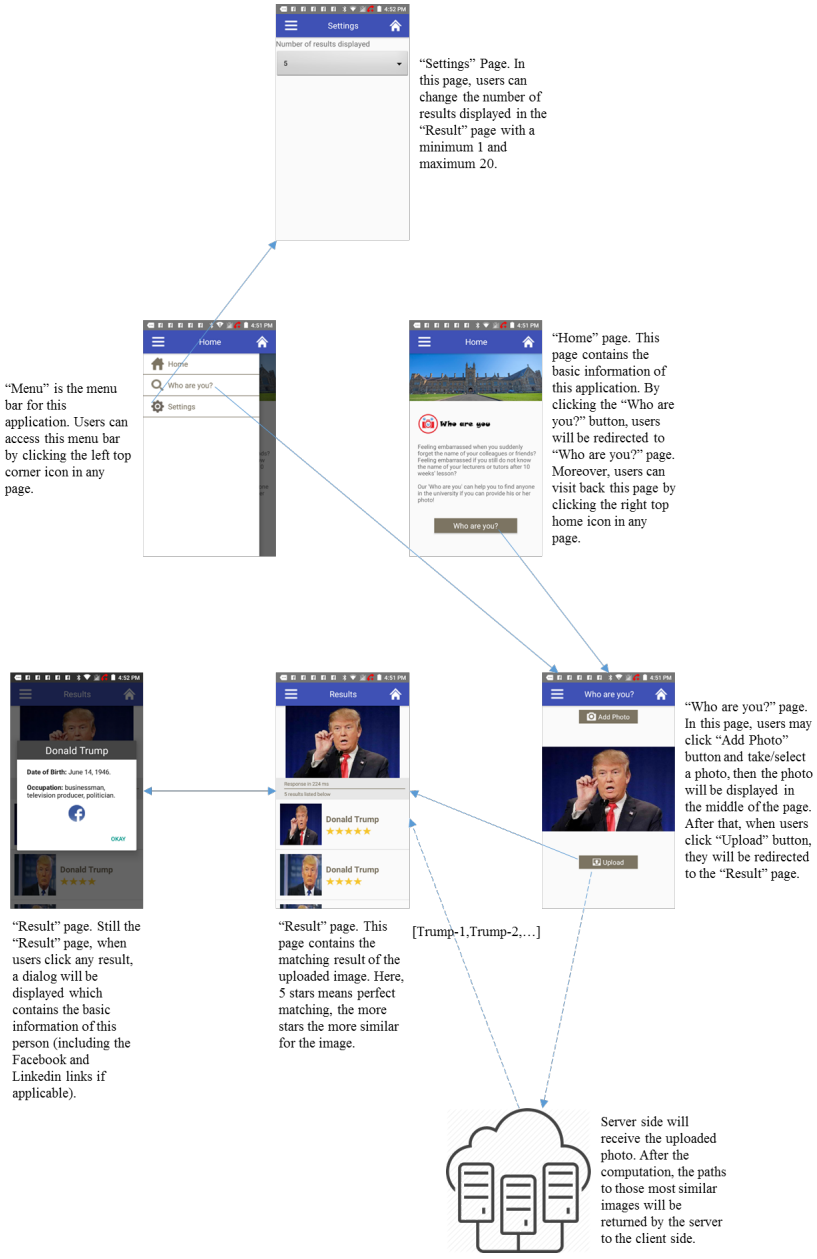


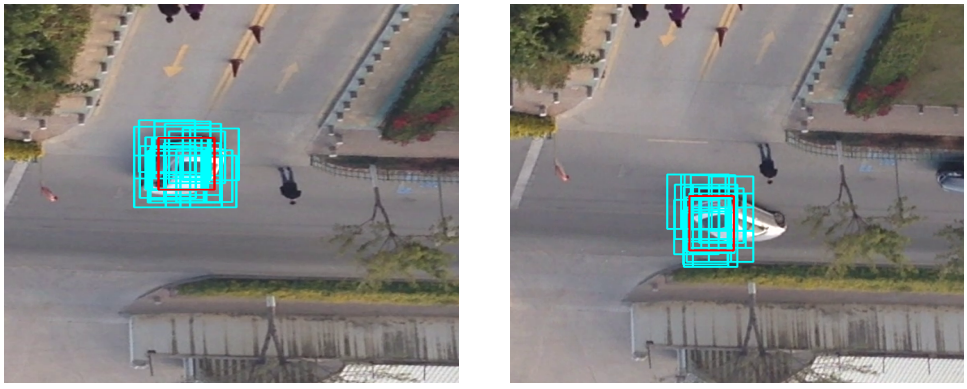
Figure 18: Workflow of the App

6 Extension

The extension work of this research is to figure out the way to scale this project from faces problem to other computer vision problem. Specifically, I would give two examples that PSO for moving object tracking and RCNN for multiple object detection.

6.1 PSO for object tracking

Except faces, I have also attempted that using PSO to track other moving objects in the real time video like car which are shown below.



(a) Frame 362 (b) Frame 541

Figure 19: PSO Tracks Car

6.2 RCNN for multiple objects Recognition

As we discuss before, by trying different images, setting up different labels and obtain the different models, RCNN is able to recognize different objects in one image, Some of attempts are as follow, the image is also input from ImageNet:

(a) Test Result 1 (b) Test Result 1

Figure 20: RCNN Detection and Recognition Experiment

7 Conclusion and Future Work

This paper summarize three approaches that EigenFace, PSO and RCNN to solve the real-human face recognition and tracking problem. These three methods have their own properties. For Eigenface it is simple and does not require a comprehensive training model, but it does not work quite well with comprehensive images. For PSO, its simplicity makes it enable to tracki face in real time in video but it cannot preform the recognition but only tracking. The RCNN, deep learning seems the best approach among all and the real-time mobile application built based on CNN also significantly prove that CNN can be an excellent way to perform the human face identification job. However it does require a very complicated trained model and a GPU machine to support the training and testing. Furthermore the extension of work shows that those methods are not just good for face problem but also have decent performance in other job therefore obviously, this job can be extended and scaled to other pattern recognition research like medical image and so forth.

Acknowledge

I would like thank my supervisor Dr Xiuying Wang and Dr Jianlong Zhou for their support as well as I would like to give my appreciation to Dr Vera Chung for borrowing me GPU machine for RCNN training and server.

References

[1] Peter N. Belhumeur, João P Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):711–720, 1997.

[2] Yoshua Bengio, Aaron C Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538*, 1, 2012.

[3] Ying-Nong Chen, Chin-Chuan Han, Cheng-Tzu Wang, Bor-Shenn Jeng, and Kuo-Chin Fan. The application of a convolution neural network on face and license plate detection. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 552–555. IEEE, 2006.

[4] De Cheng, Yihong Gong, Sanping Zhou, Jinjun Wang, and Nanning Zheng. Person re-identification by multi-channel parts-based cnn with improved triplet loss function. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1335–1344, 2016.

[5] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[6] James Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.

[7] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[8] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[9] Guang Liu, Zhenghao Chen, Henry Wing Fung Yeung, Yuk Ying Chung, and Wei-Chang Yeh. A new weight adjusted particle swarm optimization for real-time multiple object tracking. In *International Conference on Neural Information Processing*, pages 643–651. Springer, 2016.