Access Control Using Deep Learning

Geng LiangYu   
MTech (IS) Year 2 (PT)  
NUS (ISS) Singapore  
e0384909@u.nus.edu

Tan Chin Gee  
MTech (IS) Year 2 (PT)  
NUS (ISS) Singapore  
e0384927@u.nus.edu Ong Boon Ping  
MTech (IS) Year 2 (PT)  
NUS (ISS) Singaporee0384803@u.nus.edu

Hand Dongchou Francis  
MTech (IS) Year 2 (PT)  
NUS (ISS) Singapore  
0385045@u.nus.edu

Cao Liang  
MTech (IS) Year 2 (PT)  
NUS (ISS) Singapore  
e0384184@u.nus.edu

*Abstract*—An Access Control solution is proposed to protect an important asset for the main Singapore office of Oracle Corporation. The asset concerned is a Smart City Demo Asset that is displayed at the reception area of the office. The solution deploys Deep Learning technology and it differentiates authorized personnel from other people when they are in the proximity of the asset. The reason for such a security system is that the demo asset concerned is a setup with parts that are (1) delicate, (2) have a certain cost, (3) integral to the functioning ability of the whole demo.

# Introduction

A Smart City Demo Asset that is currently being built and targeting for completion by June 2020 is to be placed at the reception area of the Oracle office. This area is frequented by more than a thousand people a day and needs to be protected against damage and theft from unauthorized access. However, Oracle would like to retain the esthetic nature of the office, and so no physical barrier can be deployed in this area. As a result we are proposing the use of Deep Learning to recognize authorized personnel who can access the asset. An alert mechanism will be set up to inform stakeholders of unauthorized access when it happens, in real time. An intruder detection will also be set up to activate this system as soon as someone walks near the restricted area.



Fig 1: Reception Area of Oracle Office

# System Design

## Detecting a presence

When a person appears within the proximity of the demo asset, the security system will be turned on. This is made possible with a camera and an accompanying algorithm. The algorithm runs a background segmentation code and is constantly on a lookout for any change in the image. When a change happens, it triggers the face recognition algorithm to start operating.

## Recognizing the Face

The input is a video stream captured by one camera with a 4K resolution. The OpenCV library is employed to capture the video stream and to process the image frames. The captured images are transformed into Numpy arrays for processing. When recognition is done, the result is then shared with some peripheral systems.

In order to recognize the face from the image captured, a classifier is required. To enable fast classification, encoding of the face detection is also required. In this paper, face recognition is using pre-trained model to encode the face and the recognize the face using classifier.

Fig 2: Face detection and recognition process

The classifier requires training before it can be used to predict the face belonging to which person of interest. Hence, sampling and training must be done.



Fig 3: Face detection and recognition model training

# Face Sampling and Augmentation

In this context, face sampling consists of collection of image belonging to person of interest as well as negative targets.

Images are taken from 3 persons of interest in various angle, light exposure and expression. This will improve the accuracy and precision of the model.

Images are also taken or collected from non-targeted persons; this is aimed to reduce false positive cases.

However, only limited number of samples can be collected. 82 images are collected for targeted person and 37 non-targeted images are collected. Augmentation is used to increase the number of samples.

In this context, the images are flipped horizontally, rotated or translated. To further enhance the model accuracy, gaussian blur and Laplacian of Gaussian filter are used on existing image and generate blurred or sharpened images.

As a result, 102750 images are generated and used for training process.

# Solution

The system can monitor multiple faces on the image frame. Apart from the OpenCV2, the Haar Wavelet face cascade method (standard function used in opencv2) is used to capture the human face pixels. Also, MTCNN (courtesy of MIT) is used to refine the face area captured using Haar Wavelet. Image capturing is disallowed if MTCNN cannot capture eye/nose/mouth location.

## Constraints

Lighting can be different in between testing ground and training ground. Bright lighting (which is mainly yellow lighting that has significant R and G component) can reduce the contrast in between face and surrounding environment. Face area which is not exposed to the light may not be recognized.

Though targeted face is known, the application ground is an open space that allow free movement. Hence, accuracy varies between individual visitor.

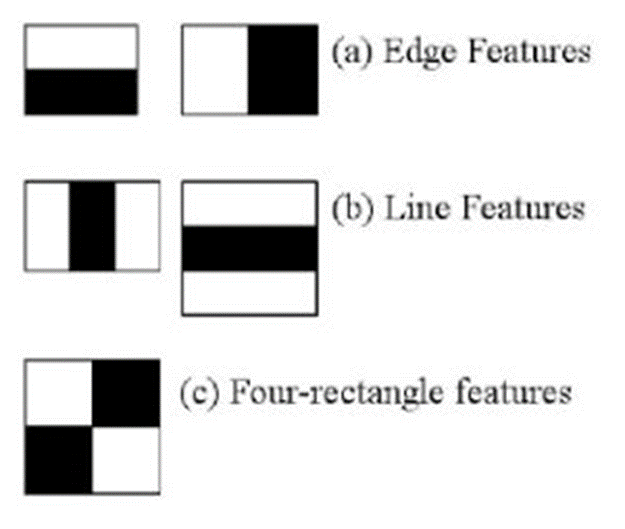
For the targeted face, not all facial expression/hair style can be captured. Spectacle effect can also be extra factor to the recognition accuracy.

## Face Detection - Haar Face Cascade

In order to allow fast face detection, Opencv Haar Face Cascade is used.

Haar Cascade [1] has 4 stages:

1. Haar Feature Selection
2. Creating Integral Images
3. Adaboost Training
4. Cascading Classifiers

Fig 3: Haar Feature Selection

For each feature, we try to find out what is the best threshold that can predict whether the trained images are human faces or not.

However, not all features obtained thru integral images are useful. The model is applied on the features are considered to be a weak classifier as each of them only hold information at a certain spatial region of the whole image.

Adaboost classifier is actually an ensemble classifier of the weak classifier that applied on each Haar feature. Boosting allows stronger feature classifier to be ensembled as a weighted averaged strong classifier.

Finally, the Adaboost classifiers are cascaded in stages. As a result, the image area that fits the face features the most are highlighted.

## Face Landmark Detection and Alignment - MTCNN

MTCNN [2] or Multi-Task Cascaded Convolutional Neural Network is used for face landmark detection and face boundary detection.

MTCNN has 3 stages of neural networks.

At 1st stage, the input image is scaled down multiple times to build an **image pyramid.** Each scaled version of image pyramid is passed through its convolutional neural network (CNN) that knowns as Proposal Network (P-Net).

P-Net is using 12x12 kernel with strides 2 to process input image. P-Net will produce the bounding box for each 12x12 kernel. The resultant bounding boxes that have higher probability (generated by P-Net as face classification result) will be parsed to next stage.

At the 2nd Stage, all the candidate bounded images from stage 1 are fed into CNN that are known as Refine Network or R-Net. 2nd stage is using 24x24 kernel on the inputs.

At each 24x24 kernel output, CNN process another set of bounding boxes and probabilities. So, we are able to reject some of the candidate images from stage 1. This is how the 2nd stage is refining 1st stage result.

*a**b* 

At 3rd stage, 48x48 kernel is applied on 2nd stage candidate bounded image.

While producing new bounding coordinates and probability of fitting the face feature, it also compute the face landmark coordinate.

5 face landmark points (left/right eye, left/right mouth, nose tip) are computed. The 5-points are obtained though minimization of Euclidean loss.

Best bounding coordinate and 5 face landmark coordinates are selected. Hence, 3rd stage is known as O-Network or output network.

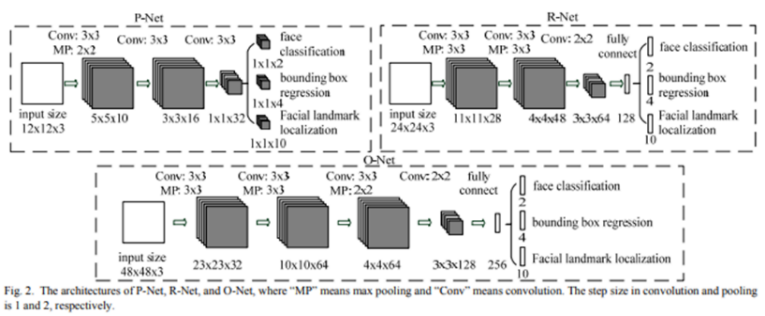


Fig 4: Architectures of P-Net, R-Net, O-Net

OpenCV Haar Face Cascade is based on Haar features. Hence, it is subjected to error where features of other objects fit the threshold given.

MTCNN can identify the face landmark (eye/mouth/nose) of an image. When a Haar Cascade image is processed by MTCNN, we can reject the image captured from Haar Cascade from further processing if face landmark coordinates are not returned from MTCNN.

In this project, the face image pixels within the MTCNN boundary are captured (MTCNN outputs boundary coordinate instead of image).

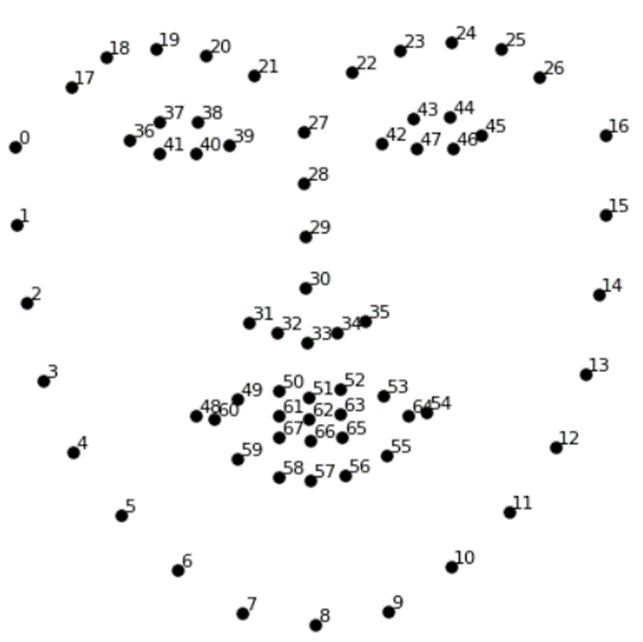
The pixels which are too far away from left eye/right eye will be cropped. Forehead and pixels under mouth will also be cropped. This will reduce possible noise due to varying hairstyles and background.

Based on left and right eye position, alignment is made so that left and right eye is within 10 degree to the horizontal axis.

## Face Encoding – DLIB based Face\_recognition

Face encoding is done through face\_recognition python library developed by Ageitgey. This library is mainly based on open source Dlib. It returns result in 128-bit code which is suitable for face encoding.

The face\_recognition process has several steps to refine and encode the face image:

* HoG transformation (further refine bounding area from MTCNN in this project)
* dlib face landmark detection and face projection
* neural network trained using triplet loss for error computation

Histogram of Gradient is an 2D array consists of gradient at each pixel. Gradient is to be computed from intensity difference across its neighbouring pixels.

Face Cascade and MTCNN should have filtered out images which are not showing sufficient face features at this point. HoG will refine the bound of image by taking out more background pixels.

HoG is working since nose, mouth, eyes should have unique gradient features and relative distance across nose, mouth, eyes, hair and background, will define the face area.

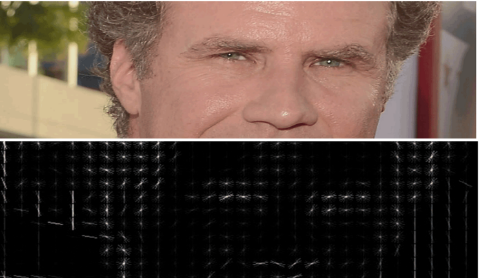
HoG is less sensitive to light intensity (Gradient usually considering surround pixels rather than light intensity at one pixel) and can define the features without influence of local light intensity. (MTCNN, face\_cascade should have taken out image with extreme light intensities, so less intensity will not have side effect). DLIB provides a fast computation of HoG.

Fig 4: HOG Representation

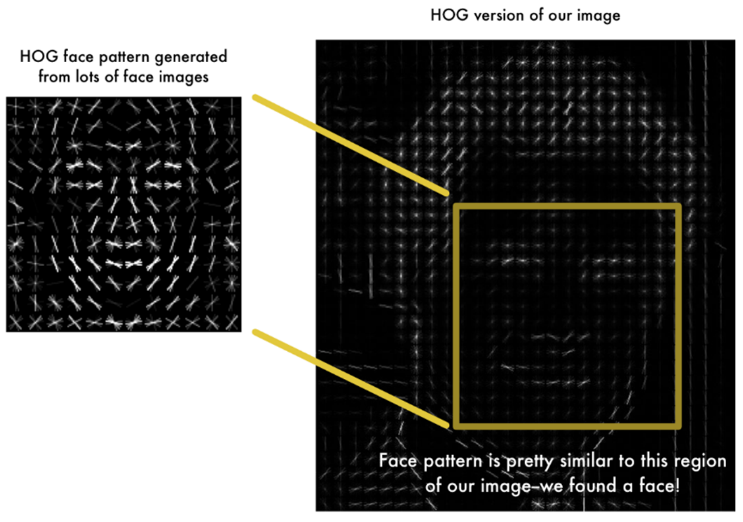


Fig 5: HOG Based Face Recognition

With HoG refined image, Dlib is able to output 68 landmark points.

DLIB has a neural network pre-trained with 68-point [**iBUG 300-W dataset**](https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/)

With DLIB’s landmark output, face\_recognition library can make face projection.

Slanting face can be re-projected with all the 68 points at largest alignment possible. Nose is at centered while the image is re-projected based on direction on other landmark point as well. (Compared to MTCNN with 5 points, total re-projection is possible)

Fig 6: 68 Landmark Points

Fig 7: Face Re-projection

With DLIB face re-projection, Ageitgey is able to train a neural network with much less sample size. It defines the input to the network to have the right face alignment. Ageitgey’s network is computing triplet loss for error correction.

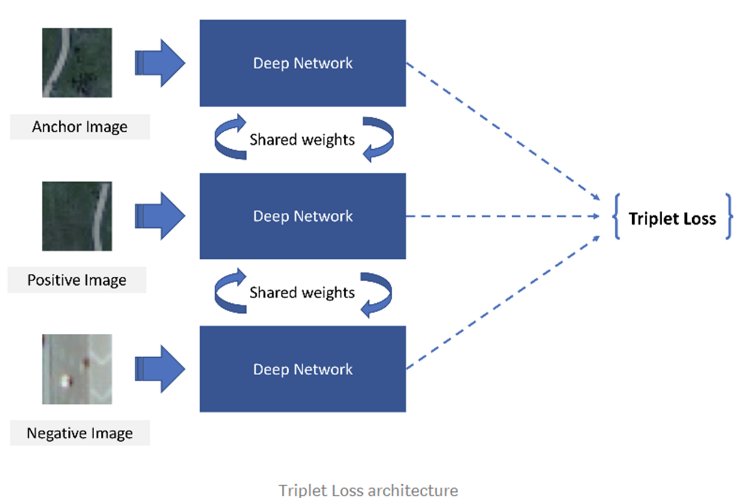
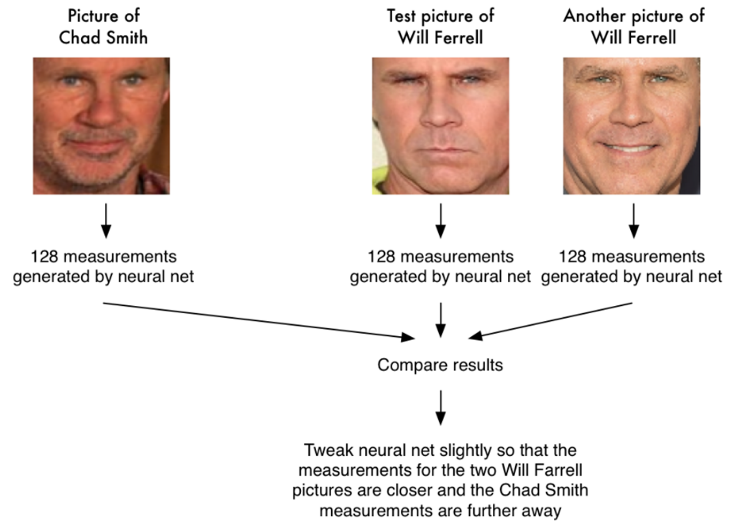
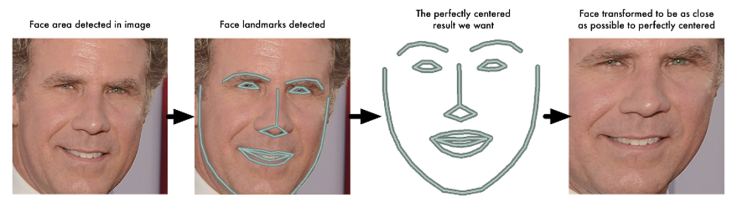
Triplet loss computation means three images are input to the network at the same time. Though there are three images, all the images are going through shared layers (this means all three sub-networks are exactly the same layers with same weight).

Fig 8: Triplet Loss Network Architecture

The 3 images are training face images of a known person, another picture of the same known person and a picture of a totally different person.

With this combination, the weight between similar faces are closer while the weight between different faces are further apart. This applies to all 128-bits that represent 128 measurements made on the face features.

With pre-trained weights from the face\_recognition python library, up to 98% accuracy is possible.

Fig 9: Triplet Loss

# Classifier

At the end of the face recognition process, an ensemble of machine learning algorithms is used to further augment the accuracy and consistency. This ensemble includes the KNN, SVM, Logistic Regression and MLP. A voting mechanism is then employed for the ensemble.

## KNN

#### KNN is suitable for face recognition. In this context, Minkowski distance when defining the nearest neighbor. The accuracy is enhanced with weight reduced with the distance increased.

#### As a result, using 9:1 train-and-test split on 102750 samples, the sklearn KNN model accuracy can reach 99.3-99.8% in the dataset with number of neighbor set to 5.

## SVM

SVM is also commonly used for face recognition.

Transformation of the 128-byte code enables support vector plane to be found.

With good image preprocessing, Dlib based face recognition library usually encodes the image accurately. This allows SVM training by using 128-byte code.

Using 9:1 train-and-test split on 102750 samples, SVM usually gives accuracy around 98.9-99.3%.

## Logistic regression

##### Logistic regression is suitable for binary classification using probability function.

##### After image is encoded into 128-byte code thru Dlib based face recognition library, logistic regression model is built for each targeted face.

##### Logistic regression is not a linear regression model and so it allow a better binary class prediction using 128-byte code.

Using 9:1 train-and-test split on 102750 samples, Logistic Regression usually gives accuracy around 96.9-99.3%.

## Multi-layer Perceptron

##### Multi-layer Perceptron is a multi layer neural network forms by perceptron. Each perceptron will have a bias, and multiple inputs with trainable weight.

##### 120 hidden layers are used in this context using lbfgs solver. Lbfgs solver is optimizing a log loss function.

##### Hence, it is able to provide different classification result compared to other networks.

Using 9:1 train-and-test split on 102750 samples, Logistic Regression usually gives accuracy around 97.4-99.5%.

## Voting

##### Voting ensemble model is used on SVM/KNN/MLP/Logistic regression. This is because all the 4 models are having binary outputs.

##### Since there are 4 targeted face, we will perform voting for each targeted face independently. If an image is recognized to be 2 or more targeted faces, it will not consider to be recognized by the model. This is to prevent cloze ambiguity.

By assigning each model to have equal vote, hard voting is made. The training result on 102750 images shows 99.3%-99.9% accuracy.

# Result

Training result shows that voting ensemble is giving higher accuracy than individual model on person of interest 2 and person of interest 3. For person of interest 1, KNN is giving higher accuracy.

In terms of false positive rate, voting network is giving less false positive on person of interest 3. Voting ensemble model is also giving close result to the best model.

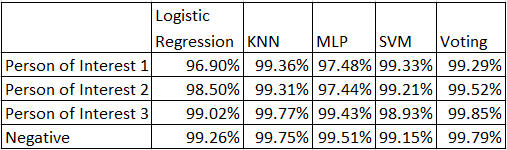


Table 1: Accuracy of classifier models

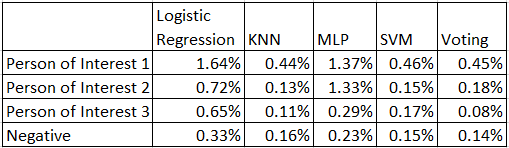


Table 2: False positive rate of classifier models

During application, voting model and weight are loaded. The face image is captured and going through all the preprocessing through face cascade, MTCNN and DLIB based face\_recognition encoding, before making prediction using voting model.

The model is giving good prediction when eyes, mouth, nose captured in the image. The model prediction is using 0.1s per frame.

##### Acknowledgment

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

##### References

1. P.Viola, “Rapid Object Detection using a Boosted Cascade of Simple Features” , 2001.
2. Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. ,“Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters”, 23(10):1499–1503 , 2016.
3. Florian Schroff, Dmitry Kalenichenko, James Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering”, 2015.