ISSN 2278-3091
Volume 9, No.5, September - October 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse63952020.pdf
https://doi.org/10.30534/ijatcse/2020/63952020

DeepEye: A Surveillance System Using Deep Learning for Intruder Detection in SmartHome Remote App

Low Qi Wei[1], Zeratul Izzah Mohd Yusoh[2], Halizah Basiron[3], Chin Kon Yee[4]
[1,4] Revenue Monster Sdn. Bhd.,
B-2-30, 2nd Floor, Block Bougainvillea,
10 Boulevard, PJU 6A, Petaling Jaya, Selangor.
[1] rex@revenuemonster.mycom, [4] ky@revenuemonster.my
[2,3] Centre for Advanced Computing Technology
Faculty of Information and Communication Technology
Universiti Teknikal Malaysia Melaka (UTeM)
Melaka, Malaysia
[2] zeratul@utem.edu.my, [3] halizah@utem.edu.my

ABSTRACT

Closed Circuit Television (CCTV) as a Video Surveillance System has been around for a long time but existing solutions are very costly and are not concerned with identity verification. This paper proposes a low cost surveillance system using deep learning for intruder detection in a remote app operating on moderate computational resources. Extensive research show that existing video surveillance system is only as good when it is monitored by a human being as they do not offer intruder detection capabilities. The motivation of this project is to mitigates the weaknesses of existing solutions by incorporates learning-based face recognition techniques to enables real-time intruder detection on a cost-friendly device. In addition, this solution includes a real time smart app for users to monitor their premise remotely. The trained face recognition model provides a method for users to train a face recognition model on Raspberry Pi and ultimately, the ability to discriminate identity in real-time. The developed solution is capable of recognizing identity at high accuracy and the final cost is cheaper compared with other offerings that uses similar techniques. The developed solution achieved an average accuracy of 90.11%, that is not far behind from techniques that require the use of expensive Graphical Processing Units. The smart app developed will allow users to monitor their premise remotely in real-time. The proposed solution is ensured to have the end product highly scalable. Experiment results demonstrate the feasibility of the solution.

Key words: Face Recognition; Deep Learning; Intruder Detection; Surveillance System.

1. INTRODUCTION

In recent years, we have witnessed the extraordinarily large deployment of Closed-Circuit Television (CCTV) in all places around the globe as a surveillance system to deter crime. The primary goal of CCTV as a Video Surveillance System (VSS) is to fight crime and act as an evidence collector. However, researchers found that the cameras deployed may be more effective as a detection tool than as a deterrent [1]. It has been a popular debate topic on the effectiveness of CCTV when it comes to crime fighting. This is especially true when it comes to the domain of premise security.

Based on the 2017 Mid-Year Crime Index report compiled by Numbeo [2], Malaysia currently ranks at number 2 with a vital crime index of 68.56 in Southeast Asia for highest crime rate with residential break-ins is one of the most contributing factors. This is a serious matter because crime apparently has a negative impact on the economic growth of a country [3]. Residential break-ins have occurred, and single-family homes are the most frequent targets [2]. Crime rate did not plummet with a significant rate even with electronic access control system in place [4]. Existing surveillance solutions need constant monitoring from the residential owners. It makes no difference if the surveillance system is left unmonitored because victims certainly will not be aware of the planned burglary. Thus, what we require is a surveillance system that is capable of identifying intruder intelligently and effectively.

The problem with the conventional approach is that existing video surveillance solutions are not as effective as how they portrayed to the public. Most video surveillance solutions only used as a tool to collect evidence and do not equipped with the ability to identity target of interest. Not only that, most video surveillance solutions come with a hefty price. This is largely due to the preinstalled array of sensors, commonly including night vision and infrared sensor. It is not justifiable for an ineffective solution to cost that large amount of money. To sum up, the conventional CCTVs are not capable of identifying intruder and dispatch alert, and most video surveillance solutions are very expensive.

As for the technique used, face recognition technique is now being massively employed in the domain of entertainment, information security and many other domains including CCTV [25]. To accommodate the increasing needs of face recognition application, numerous techniques have been invented, from the classical Fisherfaces method to the advanced convolutional models. The holistic matching methods and geometric features methods are efficient and robust but are not as accurate as the learning-based Convolutional Neural Network (CNN) models [5]. However, the learning-based methods generally require a substantial amount of computing power which is normally inaccessible by the mass public.

We present the design and implementation of A Surveillance System using Deep Learning for Intruder Detection in a Remote App (DeepEye), a proactive approach to deter intruder with the assistance of state-of-the-art face recognition algorithm.

The objective is to develop a low cost interconnected deep learning enabled surveillance suite to deter crime in residential area with the use of state of the arts algorithm. All DeepEye requires to operate is a low-cost computer, for example Arduino or Raspberry Pi and a webcam with a reasonable number of pixels connecting to the Internet. In this project, a Raspberry Pi 3 model B with an 8MP camera module will be used. Real-time video streams will be captured and be processed with Google Cloud Platform (GCP) components.

Then, the video feeds will be analyzed with a deep neural network that is being hosted on GCP. Users will be notified immediately with a remote app, if suspicious events are detected. Then, users will be able to initiate action to deter crime by utilizing connected IoT devices with the app. In short, DeepEye is trying to enhance premise security by enabling intelligence in conventional CCTV with a carefully crafted project pipeline and the state-of-the-art technology at a minimal cost.

Existing solutions [6-10] for this problem are either too expensive or ineffective. Also, it is found that some of the existing solutions did incorporate motion detector module but none of the solutions has the capability of identity verification. With the tuned deep neural network, it is possible to achieve near real-time face recognition in a cost-friendly machine with capped capability.

The rest of the paper is organized this way - the complete system architecture will be discussed in Section 2**Error! Reference source not found.**. There will be two subsections dedicated to discuss major components used in DeepEye. In Section 3, we will discuss the results of the experiment. Finally, we will conclude and briefly discuss future work in Section 4.

2. SYSTEM ARCHITECTURE

The project comprises of two main components: 1) the Deep Learning Face Recognition System, and 2) the DeepEye mobile companion app - SmartHome. Both components are fully utilizing the services offered in Google Cloud Platform (GCP) for development and deployment processes. In addition, there is also a minimal number of hardware used in this proposed solution. The detail explanation of the two main components, GCP utilization as well as the hardware used are described in Section 2. Figure 1 illustrates the overall architecture of the project.

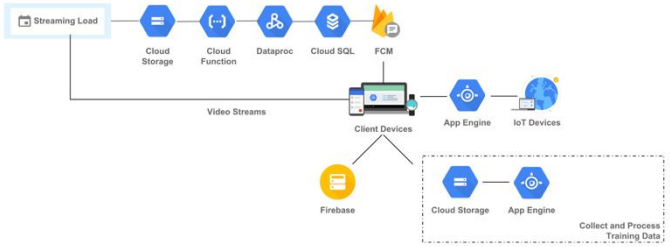

Figure 1: DeepEye System's Architecture

2.1 DeepEye Face Recognition System

A. Background Study

Several criteria were taken into consideration before building a real-time face recognition system. One of the vital objectives of DeepEye is to enable real-time face recognition with deep neural network. Since intruder detection is the main goal in this project, the ability to discriminate whether new faces are known or unknown is imperative. Previous geometric-based face recognition techniques: eigenface analysis [11], template matching [12], graph matching, fiducial point based approach [13], and others [14] are considerably efficient but their prediction accuracy is rather low compared to learning-based techniques. Learning-based approaches are proven to be much more accurate but very often require a massive amount of computing power. In the remaining section, we compare the differences between OpenFace's NN4, Facebook's DeepFace [15] and Google's FaceNet [16]. Both DeepFace and FaceNet are prominent deep learning face recognition model and have achieved a rather remarkable accuracy on the Labeled Faces in the Wild (LFW) benchmark [17]. The high accuracy of both techniques are largely due to the use of large private dataset and their proprietary face alignment and representation techniques.

To aligned to the objective dedicated in this project, we have decided to implement a deep learning approach in building DeepEye. A good deep learning model does not only rely on a fine network architecture, but also heavily depends on the training data [26]. We adopted a modified neural network model, *NN4* from OpenFace that is based on the prominent GoogLeNet architecture [18]. The differences between the

NN4 model and the two models are depicted in Table 1. It is worth noting that NN4 is trained with a publicly available dataset with much lesser images. The size of the parameters is directly proportional to computing power, which ultimately leads to lower implementation cost. This heuristic enables DeepEye to run on computer with minimal specification, for instance, Raspberry Pi.

Table 1: Comparison Between Face Recognition Algorithms

Criteria / Algorithm	NN4	FaceNet	DeepFace
Dataset	CASIA-WebFace and FaceScrub	Private dataset	Private dataset
Parameters	500k	100M-200M	4.4M
Representation technique	Triplet-loss function	Triplet-loss function	3D Alignment
*Accuracy	0.9292 $\pm$ 0.0134	0.9963 $\pm$ 0.009	0.9735 $\pm$ 0.0025

*Accuracy units are rounded to 4 digits; Accuracy are obtained through benchmark tests on the LFW dataset.

B. Data Collection

To train a deep learning model that is able to correctly discriminate premise owners and intruders, we provide an option to collect faces through the mobile companion app, SmartHome. All collected frames will be dispatched to GCP and DeepEye face recognition model will be trained automatically. Figure 2 illustrates the flow of the overall training process of the images and the technical implementation will be discussed in sub-section B.

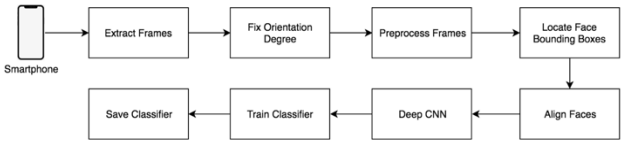

Figure 2: Model training overflow

C. Face Detection and Alignment

DeepEye first preprocess real-time images by utilizing several open source machine learning and OpenCV toolkits. To improve face detection under different sets of lighting conditions, colors in images are enhanced and equalized with a procedure known as Contrast Limited Adaptive Histogram Equalization (CLAHE). Facial features are much noticeable and faces are aligned more accurately, after applying CLAHE. After that, *dlib's* frontal face detector is used to detect faces in an image [19]. Determining the correct and precise region of a face will save up huge loads of processing resources when analyzing faces in later stages.

Faces may be appeared from all angles, direction and poses. It could be beneficial if faces are aligned to as if the person in the image is looking directly at the camera before determining the identity of the person. The face detection is the starting point for all face analysis tasks, and face alignment can be considered as crucial intermediary step for many face analyses type including biometric recognition and mental state understanding [20]. We use the facial landmarks detector provided by *dlib* to determine all 68 facial landmarks. The model provided by library was trained using the data from the iBUG 300-W dataset and it positions 68 points on frontal faces, similar to the baseline [24]. Figure 3a shows the visualization of all 68 landmarks being graphically drawn in an OpenCV frame. And Figure 3b shows the face alignment on an angle using the affine transformation method.

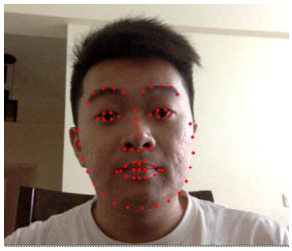

Figure 3a: Facial Landmarks

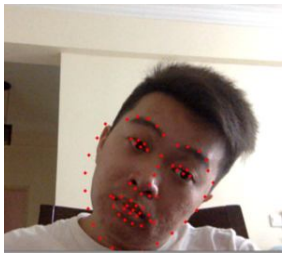

Figure 3b: Face Appearing in Various Angle

D. Deep Neural Network Face Recognition Model

When all faces in the temporary directory are aligned, the faces of all users will be passed into the deep convolutional neural network for model training. Often, face recognition applications seek for a desirable low-dimensional representation that generalizes well to new faces that the neural network wasn't trained on [21]. The preprocessed training images are too high-dimensional and thus it is very difficult to generate good prediction results during the classification phase. The NN4 neural network model is used as a feature extractor to produce low-dimensional representation that characterizes a person's face. The model learns to cluster face representation of the same person during training phase in a unit hypersphere with a triplet loss function. The embedding is represented by $f(x) \in \mathbb{R}^d$ and it embeds an image x into a d-dimensional Euclidean space. The L_2 (similarity) distance between 2 faces can be computed with Equation (1).

$$embeddings_1 = net.forward(face_1)$$
$$embeddings_2 = net.forward(face_2)$$
$$L_2 = embeddings_1 - embeddings_2 \quad (1)$$

E. Classification

A classification model is trained with Linear Support Vector Machine. All facial embedding's with their corresponding labels are saved into a file programmatically during face representation. The labels will then be encoded with values between 0 and *n_classes – 1* and transformed into respective label index before fitting into a classifier, as depicted in Equation (2). A pickle model file will be saved for face prediction.

$$labelEncoder = labelEncoder(\ ).fit(labelList)$$
$$labelIndec = labelEncoder.transform(labelList)$$
$$classiier.fit(embeddings, labelIndex) \quad (2)$$

F. Real-time Prediction

With the trained classifier, it is now possible to discriminate identify in real-time in Raspberry pi. It exhibits the same architecture as the Model Training architecture. One notable distinction is the generated embedding from Deep CNN will be passed to the trained classifier instead. Thereon, the class with the highest probability is going to be the true identity of the predicted faces.

2.2 SmartHome remote App

Home automation has been around for a quite some time now and with the emergence of smartphones, Wi-Fi and IoT (Internet of Things) have made the technology more convenient and affordable for all Mobile app frameworks [22]. The SmartHome app allows users to monitor live events of home and control connected IoT devices from anywhere. When an intruder or suspicious event is detected, a remote notification will be dispatched to the app and the users can take immediate action with connected IoT devices. In this section, we will briefly discuss components consists in SmartHome.

A. React Native

The app is built for both iOS and Android users. The cross-platform mobile framework other that native development that is chosen is React Native. React Native is an excellence in terms of performance because it provides a way to bridge native modules to JavaScript without sacrificing performance. Unlike other cross-platform mobile frameworks, React Native abstracts away the excessive WebView and directly interact to the native mobile engine [20]. In some cases, React Native outperforms native framework and provides better user experience.

B. IoT-enabled Devices

In this project, we use 2 units of TP-Link LB100 Smart Wi-Fi LED Bulb as a Proof-of-Concept (POC). Studies have shown that improved visibility in a scene will likely reduce the possibility of crime and criminals are mentally scared of lights [23]. Premise owners will feel more secure when they have control over their houses even if they are not physically in the premise. The use of IoT-enabled devices in DeepEye is to scare off intruders if suspicious events are detected by DeepEye Face Recognition System. The users will be able to remotely control the devices to prevent further losses by posting a POST request to an Express server hosted in Google App Engine, which will be discussed briefly in Section 2.3. The Express server is designed to work with all compatible IoT-enabled devices.

2.3 Google Cloud Platform

In this section, we will discuss the use of each GCP components in our project. Google Cloud Platform (GCP) provides a series of easy-to-use components that allow not only developers, but entrepreneurs to unlock full potential of their businesses by leveraging the power of cloud. Security has always and will always be a concerning topic and under no circumstances a menial task. Also, it is a top priority to ensure high service uptime and a well-thought load balancing strategy. After reviewing all major cloud providers, we decide to use GCP based on several observations:

i. Private global fiber network
ii. Live migration of Virtual Machines (VMs)
iii. State of the art security
iv. Able to handle massive traffics
v. Dedicated components for machine learning jobs

Currently, DeepEye is configured to live in *asia-south1* as we are targeting regional users at the moment. First and foremost, real-time video streams from the Raspberry Pi camera will uploaded to Google Cloud Storage (GCS) as blobs when motion or suspicious events is detected. Video blobs stored in GCS can be retrieved later as evidence. The upload process will return a *callback* and Google Cloud Function will be used to pipe the request to Dataproc where DeepEye face recognition model lives. This is where analysis will be done. Live events will be parsed and stored in Cloud SQL. If an event is found suspicious, a remote notification will be dispatch to the SmartHome app and notify premise owner.

Concurrently, SmartHome app users will be able to subscribe and view video streams from the port forwarded endpoint if they wish to see live events happening at their home. An express server is install on a Google App Engine (GAE) instance to process IoT device requests from the SmartHome app. This server is designed to process user's request to control integrated IoT devices. The flexible app engine environment is capable of scaling automatically up to a maximum number of 20 instances if traffics overloaded. Throughout the time of development, we experienced zero downtime from Google. Next, Firebase is used as a storage for the mobile app. Firebase's NoSQL database provides a fast query request which is really suitable for the nature of mobile app. An additional Cloud Storage Bucket and an GAE instance is designed to collect and process the training images from users, which will then be used to train DeepEye's deep learning face recognition model.

In short, Google Cloud Platform has proven to be valuable set of tools to DeepEye as GCP automatically handles loads and takes care of security. With GCP, DeepEye is able to run and expand its infrastructure on cloud with ease.

2.4 Hardware

DeepEye is deeply integrated with GCP and all it needs to operate is a *http* endpoint that streams video. In our project, we use a unit of Raspberry Pi 3 Model B, an 8MP camera module with a Linux streaming server, UV4L. Local video streams are recorded at 30 frames per second with a bit rate of

2000k to make sure frames collected are good enough to process. Our tests showed that the streaming server is able to operate with less than 3% CPU resources. Video streams are encoded and port forwarded to both GCP and SmartHome app.

3. RESULT AND DISCUSSION

This section will discuss an experiment that has been conducted to evaluate the performance of DeepEye. The experiment was held in a house in Malaysia, and all 6 residents agreed to take part in this experiment. In this experiment, 4 residents are to be the *"the owner"* while the rest are to be the *"the intruder"*. As for the instruments, a unit of Raspberry Pi 3 Model B with an 8MP camera module is deployed at the living room for 7 consecutive days. To train a DeepEye face recognition model, 3 seconds video for every owner are obtained through the SmartHome app and processed with the project pipeline defined in Section 2**Error! Reference source not found.**. A sample of this is as Figure 4. To validate the result, we manually cross referenced the frames of video streaming from the events detected. The result is tabulated in Table 2.

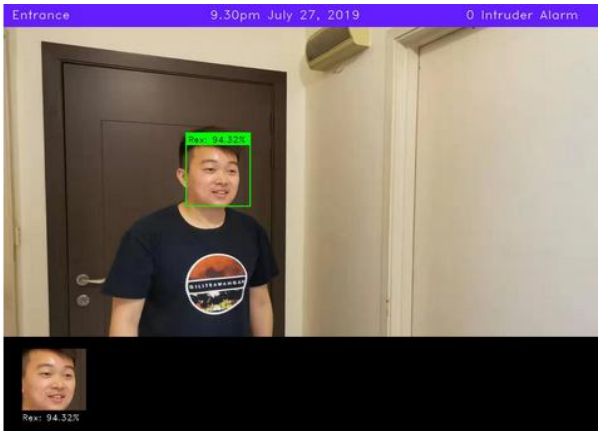

Figure 4: A sample of face recognition video footage

An average accuracy of 0.9011 is achieved throughout the experiment. Although the accuracy is high, there are several occurrences where intruders are incorrectly identified as known *"the owner"*. This is largely due to the fact that DeepEye is deployed in a constrained environment and images captured are not still. Even with heavily image processing algorithm in place, there are possibilities where the deep learning model misclassifies an unknown face as a known face. This is due to the nature of real-time video streaming where motion frames' clarity is not as good as still images, and face features might get distorted when light flashes. Hence, the deep neural network might easily get tricked.

Table 2: Experiment Results

Day	Face Detected	TP	FP	TN	FN	*Accuracy
1	54	32	2	13	7	0.8333
2	80	44	1	30	5	0.9250
3	78	48	3	21	6	0.8846
4	120	82	1	33	5	0.9504
5	64	46	2	12	4	0.9066
6	36	28	0	6	2	0.9444
7	44	20	2	18	4	0.8636

*Accuracy units are rounded to 4 digits; TP = Resident correctly identified as intruder; FP = Intruder incorrectly identified as resident; TN = Intruder correctly identified as intruder; FN = Resident incorrectly identified as intruder;

We also studied the performance of Google App Engine instances in our project and we found that generally, it takes around 3 to 4 seconds to complete a job on GCP and dispatches a notification to SmartHome app. Overall, the GCP pipeline does a great job processing user requests and we experienced no downtime during throughout the experiment. There is a period where latency spikes to 7 seconds and we suspect it is due to internal network congestion with GCP.

4. CONCLUSION AND FUTURE WORKS

To conclude, DeepEye with SmartHome mobile companion app has proven to be effective in detecting intruders. State of the art deep learning face algorithm is employed to ensure high accuracy. However, prediction accuracy might vary in different places due to the fuzzy set of lighting and constrained environment. Deep integration with GCP allows load balancing with ease and system development gets tremendously faster. As for future work, we plan to reshape the algorithm to an incremental learning approach where the deep neural networks learn to discriminate intruders from house residents even when faces appear from all possible angles and directions. In addition, we reckon that it is possible to evolve DeepEye completely into Platform as a Service (PaaS) business model. Existing CCTVs and any video capturing devices can be an input for DeepEye and users can enjoy the ability to detect intruders with the SmartHome app. For example, DeepEye is able to convert a conventional premise CCTV into a full-fledged intruder detection system without incurring excessive hardware costs.

ACKNOWLEDGEMENT

We acknowledge financial assistance and the support from Universiti Teknikal Malaysia Melaka.

REFERENCES

1. Jennider, L. (2009, March 3). **Study Questions Whether Cameras Cut Crime. Retrieved** from https://cityroom.blogs.nytimes.com/2009/03/03/study-questions-whether-cameras-cut-crime/
2. **Malaysia 2017 Crime & Safety Report.** Retrieved October 13, 2017, from

3. Gaibulloev, K., & Sandler, T. **Growth consequences of terrorism in Western Europe**. Kyklos, 2008, 61(3), 411-424.
4. Isnard, A., & Council, T. C. **Can surveillance cameras be successful in preventing crime and controlling anti-social behaviors.** In The character, impact and prevention of crime in Regional Australia Conference, Australian Institute of Criminology, Townsville, 2001, pp. 2-3.
5. **Labeled Faces in the Wild Results**. Retrieved from http://vis-www.cs.umass.edu/lfw/results.html
6. **Yoshop(TM) Home Security Dvr Dome Cctv Security Camera with Tf Card Night Vision:** Amazon.ca: Electronics. Retrieved from https://www.amazon.ca/Yoshop-Security-Camera-Night-Vision/dp/B01I8Y2XOA
7. **Security Video Surveillance Solutions** Retrieved from http://www.sensorlink.com.my/video-surveillance/
8. **Belco Security Sdn Bhd.** Retrieved from https://belcosecurity.com/
9. **Ei System Solution Sdn Bhd.** Retrieved from http://www.ei-sys.com/cctv.php
10. **NuTACT SmartCCTV.** Retrieved from http://www.nutact.com/index.html
11. Chellapa P., Wilson C., and Sirohey S., **Human and Machine Recognition of Faces: A Survey**, Proceedings of IEEE, vol. 83, no. 5, pages 705-740, 1995.
12. Brunelli R., and Poggio T. **Face Recognition: Features versus Templates,** IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 15, no. 10, pages 1042-1052, 1993.
13. McKenna S.J., Gong S., et al. **Tracking Facial Feature Points with Gabor Wavelets and Shape Models**, Lecture Notes in Computer Science, Springer Verlag, 1997.
14. Samal D. and Starovoitov V. **Approaches and methods to face recognition. A survey**, Institute of Engineering Cybernetics, Preprint #8, Minsk, 54 pages, 1998
15. Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, Lior Wolf. **DeepFace: Closing the Gap to Human-Level Performance in Face Verification. Computer Vision and Pattern Recognition** (CVPR), 2014.
16. Florian Schroff, Dmitry Kalenichenko, and James Philbin. **FaceNet: A Unified Embedding for Face Recognition and Clustering. Computer Vision and Pattern Recognition** (CVPR), 2015.
17. G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. **Labeled faces in the wild: A database for studying face recognition in unconstrained environments**. Technical Report 07-49, University of Massachusetts, Amherst, October 2007
18. C. Szegedy, W. Liu, Y. Jia et al., **Going deeper with convolutions**, arXiv preprint arXiv: 1409.4842, 2014.
19. D.E. King. Dlib-ml: **A Machine Learning Toolkit. Journal of Machine Learning Research**, vol. 10, pp. 1755-1758, 2009.
20. Jin, X., & Tan, X. **Face alignment in-the-wild: A survey**. Computer Vision and Image Understanding, 2017, 162, 1-22.
21. Amos, B., Ludwiczuk, B., & Satyanarayanan, M. Openface: **A general-purpose face recognition library with mobile applications**. 2016, CMU School of Computer Science.
22. **Wireless Home Automation using IoT**. Retrieved January 10, 2017, from https://www.elprocus.com/wireless-home-automation-using-internet-of-things/
23. Atkins, S., Husain, S., & Storey, A. **The influence of street lighting on crime and fear of crime**. London, UK: Home Office, 1991
24. Day, M.. **Exploiting facial landmarks for emotion recognition in the wild**. arXiv preprint arXiv:1603.09129
25. M. Africa AD, Borja G., Chua A., Ong D., Roque M., **Application of Computer Systems in Facial Recognition Efficiency**, International Journal of Advanced Trends in Computer Science and Engineering, Vol. 8 (4), pp 1168-1173, Aug 2019
26. Sukhada Chokkadi, Sannidhan MS, Sudeepa KB, Abhir Bhandary, **A Study on various state of the art of the Art Face Recognition System using Deep Learning Techniques**, International Journal of Advanced Trends in Computer Science and Engineering, Vol. 8 (4), pp 1590-1600, Aug 2019