Project Houseleek - A Case Study of Applied Object Recognition Models in Internet of Things

Jure Knezović*, Branimir Pervan*, Zvonimir Relja** and Josip Knezović*
*University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, Croatia
jure.knezovic@fer.hr; branimir.pervan@fer.hr; josip.knezovic@fer.hr

**Brain Information Technologies d.o.o., Zagreb, Croatia
zvonimir@brain-it.hr

***Abstract* - Nowadays, the gap between academic work and practical application of that work is rapidly diminishing. This fact can be backed by several factors: the increase in availability of the research results, as well as research artifacts; the rise in the level of education in general; the availability of broadband networks and the more affordable prices of the technology used for research. Also, due to the pervasion of the technology in all spheres of society, there is an emergence of new possibilities of applying disruptive technologies at all levels, including homes or workplaces of individual users.**

This paper presents Project Houseleek: a multilayer system that utilizes disruptive technologies to enhance and facilitate access to individual premises in smart areas. On the authentication layer, the system uses disruptive deep learning technologies to identify or learn itself a person in a real-world environment from an image grabbed in relatively rough conditions, while at the authorization layer it learns at runtime the access rights to specific parts of the smart area for that person. The testing system is implemented at the Department of Control and Computer Engineering, Faculty of Electrical Engineering and Computing where it exceeded the expectations of the users on both authentication and authorization layers.

Keywords - deep learning, smart home, internet of things, face recognition

I. Introduction

Automation is the technology by which a process or procedure is performed without human assistance [1]. Although the term “automation” has mostly been used in the context of industrial automation, recently it is used to describe home automation as well. Home automation is, therefore, an automatic centralized control also known as building automation for a home, shortly the smart home or the smart house. The potential uses of the Internet of Things (IoT) in home automation are endless. Every home device connected to the Internet can be programmed to automate something that was usually carried out manually. IoT can be implemented to control lighting, climate, appliances, and numerous other processes. Home security is one of such applications of large importance and with the ability to be improved by employing the concepts from IoT. Most relevant to home security is the process of access control and alarm systems [15].

Machine learning is sub-branch of artificial intelligence, representing a group of methods of data analysis that are used to automate analytical model building in a way that computers can learn and improve without being explicitly programmed. The basic idea of machine learning is to develop the computer programs that can access the data, identify patterns and make decisions or predictions with minimal human intervention and adjust actions accordingly [2]. Particularly, machine learning algorithms can be used to achieve higher levels of efficiency while being applied to the IoT, therefore saving time, energy, and money.

Project Houseleek is designed to create a specific home IoT environment that would carry out users’ authentication and authorization in smart environments by detecting the movement in front of the door and identify persons triggering the movement sensors. The aim is to establish a connection for each one of the IoT modules with the centralized server, on which face recognition algorithm will be run. Furthermore, the goal is to keep the hardware price of the designed IoT module relatively low (under 10$).

The rest of the paper is organized as follows: Section II. describes the implementation in detail including technical details, communication protocols, object detection, and face recognition algorithms. Section III. concludes the paper including future work.

II. Implementation

A. Overview

The first objective of the Houseleek project is to set up the home IoT environment that detects movement at the front door, captures a picture with the camera and decides if there is a person on the captured image. A centralized solution called openHAB [14] is used to establish a connection between the server and remote modules. OpenHAB has predefined rules of communication with Houseleek server that has face recognition and object detection algorithms. For example, if movement happens at the front door, openHAB will request Houseleek server for a face recognition algorithm. If the server has successfully identified the person as a familiar person, for example, a homeowner or a family member, a warm welcome message to greet the person at the front door will be played. In the opposite case, when that person is a stranger, home IoT system should send a notification message to the homeowner's mobile phone and alert homeowner of the possible threat.

Hardware parts used to meet these requirements are ESP32 microcontroller [3], a motion detector (PIR sensor), OV2640 camera, and stereo speaker. Rules correspond to different actions and represent a specific reaction. Therefore, all rules defined for several possible actions that our home system should observe were employed within the home IoT system. Any action observed by home IoT system is sent to open source centralized solution openHAB from where the rule for the upcoming reaction is determined. For example, when the homeowner appears at the front door, the system would observe his appearance as an action, and a predefined rule to that action would be playing the welcome message.

Houseleek implementation consists of the following actions:

- detect movement
- alert about the movement
- acquire a picture from the camera
- send the picture to the face recognition API.

If a person is found on the picture, the picture is then forwarded to face recognition API

- if face recognition API recognizes a familiar person, play the welcome message with their name on the home stereo speakers
- if face recognition API does not recognize a familiar person, send a warning message to homeowner's mobile phone.

Basic implementation of the described system is shown in Figure 1.

B. Communication

Communication between openHAB server and face recognition algorithm is done using Hypertext Transfer Protocol (HTTP), while IoT devices communicate with openHAB using Message Queuing Telemetry Transport protocol (MQTT protocol). Therefore, to connect software with openHAB, face recognition and object detection need to be available over HTTP request/response method. HTTP request contains an image for the analysis, and an HTTP response sent from Django server to openHAB contains data with information about recognized objects from the image. openHAB then reads the response and acts according to the predefined rule.

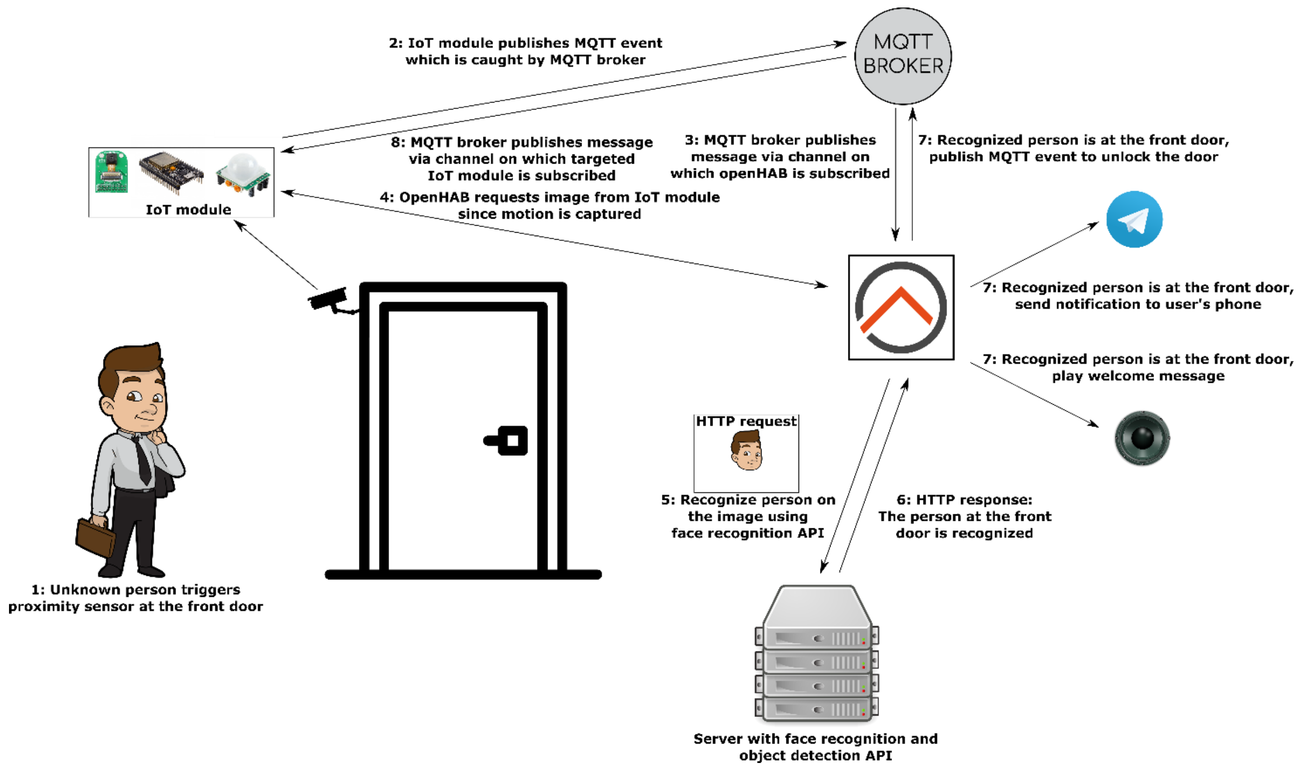
2: IoT module publishes MQTT event which is caught by MQTT broker
MQTT BROKER
8: MQTT broker publishes message via channel on which targeted IoT module is subscribed
3: MQTT broker publishes message via channel on which openHAB is subscribed
7: Recognized person is at the front door, publish MQTT event to unlock the door
IoT module
4: OpenHAB requests image from IoT module since motion is captured
7: Recognized person is at the front door, send notification to user's phone
7: Recognized person is at the front door, play welcome message
HTTP request
5: Recognize person on the image using face recognition API
6: HTTP response: The person at the front door is recognized
1: Unknown person triggers proximity sensor at the front door
Server with face recognition and object detection API

Figure 1. System components with communication scheme

Based on those actions, openHAB has following rules set up, which define what will be executed when the corresponding action is detected:

- if a movement is detected, send a picture from the camera to object detection API
- Object detection, using methods described in paragraph C, is run on the picture from the camera.

MQTT is a protocol often used in IoT, where network bandwidth may be extremely limited, and is therefore designed to be extremely lightweight. The MQTT protocol works on top of TCP/IP, where both the client and the broker need to have a TCP/IP stack. The client can be represented with any device that aims to publish or subscribe to the predefined topics, and in houseleek project that would be the ESP32 microcontroller. The broker side of the protocol is responsible for handling all the announcements and subscriptions. The concept of MQTT protocol is different than HTTP as the client does not have to pull the information, but the broker pushes one to the client in case of a new event, as shown in Figure 2. To establish that, each client has a permanently opened connection to the broker. If such a connection is interrupted by any means, the broker can buffer all messages and send them to the client when it becomes available again. Microcontrollers in this project act as Houseleek nodes that are publishing and subscribing to topics. Of all the defined topics, the most important are those for motion detection and camera. The topic associated with the motion sensor has a route named */iot/motion/id*, where <u>*id*</u> is the identifier of the sensor. Mounted test sensor has a route named */iot/motion/1*. The ESP32 microcontroller reads the data from the motion sensor and if a case of movement, it publishes the event to the topic with the message "*TRUE*". Any device in the network that

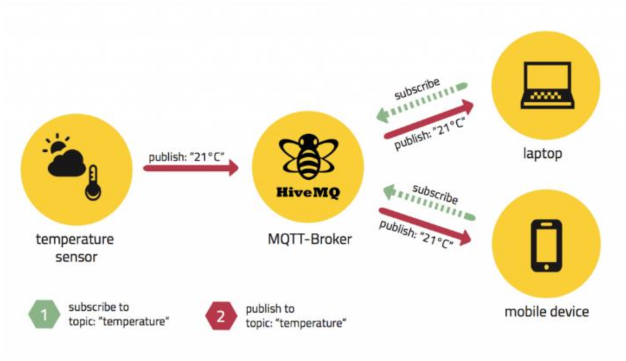

Figure 2. Illustration of MQTT protocol scheme for weather temperature [17]

is subscribed to that topic will automatically know that movement has been observed because they are waiting for the message with payload "*TRUE*". The broker part of the MQTT protocol is installed on Houseleek server node. As a main part of the publish-subscribe protocol, it enables a communication channel by which all the messages are broadcast. It attends all subscriptions, publishes events, and serves as a central hub that is highly scalable and is easy to monitor. When a new message needs to be published, the broker automatically sends it to subscribers, *i.e.* nodes subscribed to the corresponding topic.

MQTT allows authentication and authorization which is an important addition to any IoT system. Username and password are set within the MQTT broker part and are required to be sent together with the data payload each time an event for publish or subscribe action occurs. Therefore, Secure Sockets Layer certificate was installed to protect the username and password as it is not safe to send them as a plain text through the opened network. In that way, username and password are sent encrypted through the network channel.

C. *Object detection*

The TensorFlow [4] framework is used to fulfill project objectives established for object detection. It is an open source machine learning framework, developed for internal use by Google, but released under the Apache 2.0 open source license on November 9, 2015. TensorFlow object detection API includes construction, training, and deployment of object detection models. It accounts for the structuring of various necessary elements for object detection while giving the user an ability to quickly explore different configurations using TensorFlow backend. TensorFlow can be paired with collections of pre-trained detection models; COCO dataset [5], Kitti dataset [6], and Open Images dataset [7]. Those datasets are large-scale object detection, segmentation, and captioning datasets. All of them include models such as people, cars, animals, and various other groups of objects. As this project requires to detect people, those models will be extremely useful to fulfill all the project objectives. We have chosen to use COCO dataset since it contains the vast variety of pre-trained models with easy to compare specifications. Of many models for object detection available within TensorFlow API, most common ones are Single Shot Multibox Detector (SSD) developed and published by Google in 2016 [8], and Faster R-CNN developed by Microsoft [9].

Single Shot Multibox Detector (SSD) is an object detection model which uses a single deep neural network combining regional proposals and feature extraction. Various sets of boxes, defined with different aspect ratios and scales, are employed to the feature maps. The feature extraction for the bounding boxes can be extracted within the single pass of the image through an image classification network when all maps are computed. Scores are generated for each object category in every of the default bounding boxes, and for better fitting of the ground truth boxes, adjustment offsets are calculated for each box [8]. For image classification within SSD object detection model, one can choose various classification networks. Depending on project needs, each classification network brings certain benefits. Inceptionv3 network [10] is trained to detect objects at different scales, ResNet [11] achieves very high accuracy overall, where Mobilenet network [12] is trained to minimize the required computational resources. The performance data of the COCO-trained models for those classification networks of SSD architecture was evaluated using Nvidia GeForce GTX TITAN X card. In this project we focused on achieving the best precision in reasonable time, meaning that the use of SSD model architecture would bring faster detection, but it would also reflect on lower detection precision. On the other hand, Faster R-CNN typically outperforms SSD, but it requires significantly more computational power. Faster R-CNN models should be used if the project demands better precision, and the total execution time is not relevant. For this project, speed was of more essence compared to precision, which resulted with the decision to employ SSD model architecture for our system, since we only need to distinguish a person from other objects on the picture.

Object detection API is needed to connect the chosen pre-trained model with the utilized server that will communicate via HTTP protocol. HTTP request should contain an image that is needed to be analyzed, and HTTP response should inform if something is recognized on the image. For HTTP server, Django is chosen, a high-level Python Web framework. OpenHAB communicates with Django web server via HTTP requests. When Houseleek node (ESP32 microcontroller, camera and motion sensor) detects a movement, openHAB makes a request which contains JPEG image that needs to be processed with Django view function. View function takes a JPEG image and runs it through TensorFlow object detection API, which uses SSD model architecture with Mobilenet network. When TensorFlow processes the image, view function takes as the input all detected classes with corresponding scores, which are probabilities that the object is on the image (percentage 0-100%) and returns them as an HTTP response to openHAB.

D. Face recognition

In paragraph C, TensorFlow object detection API which is implemented in our system was described, which can easily detect humans at the front door (images), but it is not able to recognize them. Without face recognition API, TensorFlow object detection would act as a smart proximity sensor which can then trigger a relay and allow entrance only if persons are found in the picture. To recognize persons on the picture, trained model over the images of people who the homeowner wants to be recognized needs to be established. This process would require months of image gathering of people at the front door. Such images would need to be labeled so that the model knows who is on which image. As that job is tedious and does not guarantee success, other options were explored using deep learning algorithms. Recognizing a person on an image is not a simple assignment. To overcome such task, several sub-assignments need to be addressed accordingly:

- Look at the image and detect all faces.
- Focus on each face and recognize the person even if the face is u turned in a different direction or has bad lighting.
- Pick out unique features of the face to set it apart from other people - e.g. distance between the eyes, size of a nose, eye color, face length, etc.
- Determine the person’s name by comparing such unique features to all already known people from the database

These problems have already been solved and are incorporated into a single Python library for face recognition [16]. The algorithm used in this project uses the Histograms of Oriented Gradients (HOG) method to detect faces [19]. It converts the image to black and white and compares each pixel with surrounding ones. The main point of such comparison is to create a vector, pointing in the direction where the image is getting darker. The process is repeated for every pixel in the image, where each pixel is replaced with such an arrow. Those arrows are gradients and indicate the light flow from bright to the dark part of the image. Since the image has many pixels, the algorithm breaks the image into small squares of 16x16 pixels and saves only one gradient per square. The gradient that would be saved is one repeating the most within the corresponding 16x16 box. The result is shown in Figure 3.

Second face recognition problem is to recognize the same face turned into different directions. The problem is solved by warping each picture so that a person’s eyes and lips are always at the same place. To find eyes and lips, the library uses the algorithm by Kazemi and Sullivan [13]. Fundamental idea is that the face has 68 specific points called landmarks. For example, the top of the chin, the inner edge of each eyebrow, top of the nose, etc. The machine learning model is trained to find these 68 specific points on any face. This way the image is prepared for the next step of face encoding where specific features of the face will be extracted.

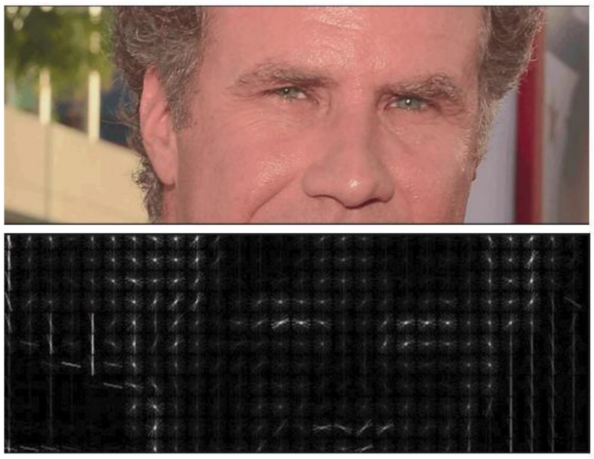

Figure 3. Image before and after applying a face detection algorithm [18]

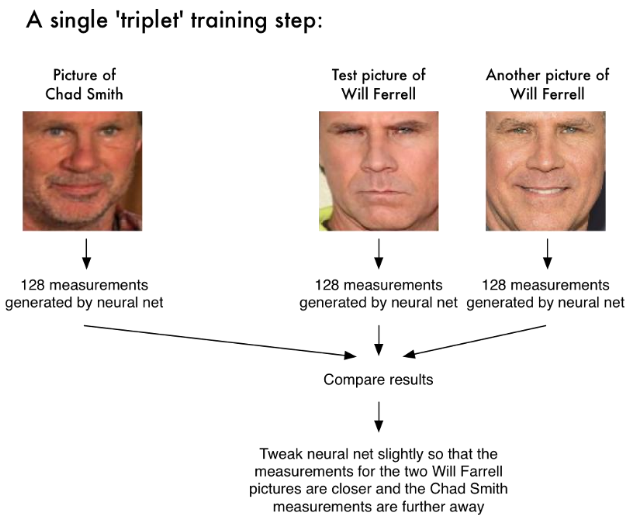
A single 'triplet' training step:
Picture of Chad Smith
Test picture of Will Ferrell
Another picture of Will Ferrell
128 measurements generated by neural net
128 measurements generated by neural net
128 measurements generated by neural net
Compare results
Tweak neural net slightly so that the measurements for the two Will Farrell pictures are closer and the Chad Smith measurements are further away

Figure 4. Deep learning one training step [18]

Face encoding is a process which converts a person’s face to quantifiable metrics. The typical approach would be to apply some mathematical and computer vision algorithms that calculate the distance between specific face landmarks, but lately, deep learning techniques are being exploited. Deep learning has a greater possibility of learning which parts of the face are more important to measure. This Python library for face recognition applied a deep learning technique by training the network to generate measurements for each face. Training process requires three faces at a time. The base of this algorithm is the triplet training method, shown in Figure 4 [20]. First two images are the face of a known person, while the third picture is the face of an unknown, random person. The algorithm looks at the measurements it is currently generating for each of those three images. It then tweaks the neural network slightly so that it makes sure the measurements it generates for pictures of known person are slightly closer while making sure the measurements generated for the unknown person are slightly further apart. After repeating this step millions of times for millions of images of thousands of different people, the neural network learns to reliably generate measurements for each person. Once the network has been trained it can generate measurements for any face, even ones that were not in the training dataset. Getting the face encoding is a useful feature for this project because project stores face encoding for familiar persons in one file that contains the name of the familiar person and face encoding. The face recognition API opens that file and compares the face encoding of unfamiliar persons to familiar face encoding from the stored file.

III. CONCLUSION

Home automation is coming into average homes faster than expected. To demonstrate the potential capabilities of next-generation IoT systems, a test system with a motion detector and an inexpensive camera was developed. On the other side, Deep learning methods were exploited to demonstrate the accessibility of complex and powerful techniques for security improvement and automation in smart homes. To achieve that, a subsystem for object detection and face recognition was implemented, utilizing advanced face encoding techniques with feature extraction and a set of transformations being applied to human face objects.

Given the configuration the system is built upon, namely a relatively low resolution and a non-high-quality input image, cheap and slow devices to build up IoT modules, and average hardware used to run recognition services, the quantitative evaluation of the system is yet to be performed. The recognition accuracy measurement together with the comparison with other similar systems remains as a part of future work. Also, future work will include the upgrades of the system with more controllers and higher accuracy sensors. On the software side of the project, there is always room for more efficient and more reliable algorithms and services.

BIBLIOGRAPHY

[1] Groover, M. P. (2007). Fundamentals of modern manufacturing: materials processes, and systems. John Wiley & Sons.

[2] Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

[3] Espressif Systems. ESP32 Technical Reference Manual. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf Last Accessed: 25 February 2019.

[4] TensorFlow. About TensorFlow. [Online]. Available: https://www.tensorflow.org/ Last Accessed: 25 February 2019.

[5] COCO dataset. [Online]. Available: http://cocodataset.org/ Last Accessed: 25 February 2019.

[6] The KITTI Vision Benchmark Suite. [Online]. Available: http://www.cvlibs.net/datasets/kitti/ Last Accessed: 25 February 2019.

[7] Open Images Dataset V4. [Online]. Available: https://storage.googleapis.com/openimages/web/index.html Last Accessed: 25 February 2019.

[8] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.

[9] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

[10] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

[11] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[12] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[13] Kazemi, V., & Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1867-1874).

[14] openHAB. openHAB - empowering the smart home. [Online]. Available: https://www.openhab.org/ Last accessed: 25 February 2019.

[15] Consumer Reports. Best home automation system. [Online]. Available: https://www.consumerreports.org/cro/magazine/2014/06/run-your-home-from-your-phone/index.htm Last accessed: 25 February 2019.

[16] Face Recognition Library. [Online]. Available: https://github.com/ageitgey/face_recognition Last Accessed: 25 February 2019.

[17] Kang, D. H., Park, M. S., Kim, H. S., Kim, D. Y., Kim, S. H., Son, H. J., & Lee, S. G. (2017, February). Room temperature control and fire alarm/suppression IoT service using MQTT on AWS. In *2017 International Conference on Platform Technology and Service (PlatCon)* (pp. 1-5). IEEE.

[18] Data Science, Machine Learning, & Visualization. Facial Recognition Challenge: Chad Smith & Will Ferrell. [Online]. Available: https://paulvanderlaken.com/2017/11/21/facial-recognition-challenge/ Last accessed: 4 April 2019.

[19] Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In international Conference on computer vision & Pattern Recognition (CVPR'05) (Vol. 1, pp. 886-893). IEEE Computer Society.

[20] Hoffer, E., & Ailon, N. (2015, October). Deep metric learning using triplet network. In International Workshop on Similarity-Based Pattern Recognition (pp. 84-92). Springer, Cham.