Biometrics Based Attendance Tracking System

A File Paper

Presented to

Department of Computer Science

Midwestern State University

In Partial Fulfillment

Of the Requirements of the Degree

Master of Science

By

Ali Ahmadiadl

May 2019

**Abstract**

*Attendance tracking is an important consideration in most educational institutions around the world, and despite all the technological advancements in recent years it is still being done using traditional time-consuming methods like roll call in many cases. In this paper we propose a Biometric Based Attendance Tracking System (BBATS) using facial recognition, that will be easily accessible for educators as a smartphone application. We approach this problem by using deep learning and facial recognition algorithms together. The system is designed to facilitate classroom attendance by incorporating a “multiple facial detection” capability. This capability enables the user to simply take and upload a single photo of an entire class. Each photo is subsequently processed letting our machine learning models locate and identify students giving an instructor an accurate representation of attendance. Over time this will allow an instructor more time to teach while* *maintaining consistent and complete attendance records.*

**1. Introduction**

In most educational institutions around the world, classes usually begin with the teacher, lecturer, professor, or any educator in general, ensuring that all the students are present in the classroom. Education is important and students cannot learn if they are not attending their classes to do so. In the traditional way the educator usually calls out the name of the students who prove their attendance in a call-and-response manner. This method has been used for centuries and clearly has flaws. It is very time-consuming and issues, such as present students responding for an absent one, are inevitable. These flaws can be mitigated by using technology. Some of the more commonly used technological solutions are using student ID card reader attendance tracking modules at classrooms or using biometric based tracking systems like fingerprint scanners. Despite the modern nature of these solutions, unfortunately, they are also flawed. For example, present students could swipe the ID cards for their absent friends. More importantly, these solutions require extra hardware to be installed along with the infrastructure to support it. The solutions are not easy to implement and add an additional financial strain on the educational institutions. These mentioned methods can also be time-consuming due to the long check-in lines. The solution we propose in this paper uses deep learning facial recognition techniques implementing a biometric based attendance tracking system (BBATS) that eliminates the flaws in the previously stated solutions.

**2. Method**

Our goal was to design a simple, efficient and easy to use alternative solution for student attendance tracking. The most time-consuming portion of using this solution is only done once, right at the beginning of each semester. The educator is tasked with taking a single photo of each student enrolled in his or her class using a smartphone application. From then on, the educator simply takes a photo of the entire class at the beginning of each class session for the remainder of the semester to take attendance.

BBATS consists of three components: a facial recognition engine and a backend API, which are both located on the server, along with a mobile application that will be installed on user’s devices. The engine and the app communicate with each other using the API.

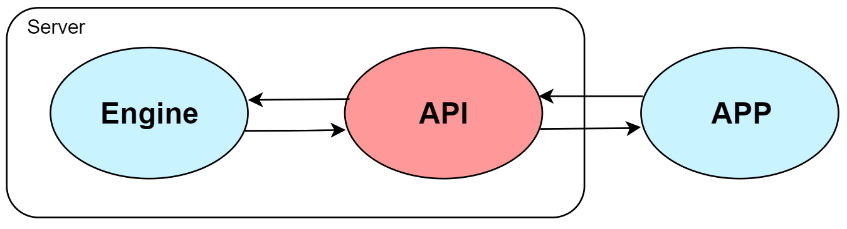


Figure 1. BBATS components

**2.1. Facial Recognition With Deep Learning**

In the past few years, facial recognition, like many other computer vision applications, has followed the same transitions. Traditional methods based on hand-engineered features have been replaced by deep learning methods based on CNNs [1]. Face recognition systems based on CNNs, like FaceNet [2] and DeepFace [3], have become the standard due to the significant accuracy improvement achieved over other types of traditional techniques like Eigenfaces [4], Fisherfaces [5], and Local Binary Patterns Histograms [6]. Besides FaceNet and DeepFace, some of the popular efforts in facial recognition with deep neural networks are the Visual Geometry Group (VGG) Face Descriptor [7] and Lightened Convolutional Neural Networks (CNNs) [8]. In this paper we use OpenFace [9] which uses a modified version of FaceNet.

There are 4 steps for facial recognition, face detection (finding all the faces in an image), face alignment/preprocessing, encoding faces (extracting facial embeddings) and face matching/classification (finding the person’s name from the encodings).

**2.1.1. Facial Detection and Normalization**

In order to extract faces from the image we need to use a face detector. Face detectors find the position of the faces in an image and, if any, return the coordinates of a bounding box for each one of them. We use the Dlib [10] library to implement a face detector that uses a combination of the Histogram of Oriented Gradient (HOG) descriptor [11] and the Support Vector Machine (SVM) classifier [12]. The idea behind the histogram of oriented gradients descriptor is that object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions [11]. In this method we compare sub-windows of the input image to a HOG face pattern generated from lots of face images using a linear SVM for binary face/non-face classification.

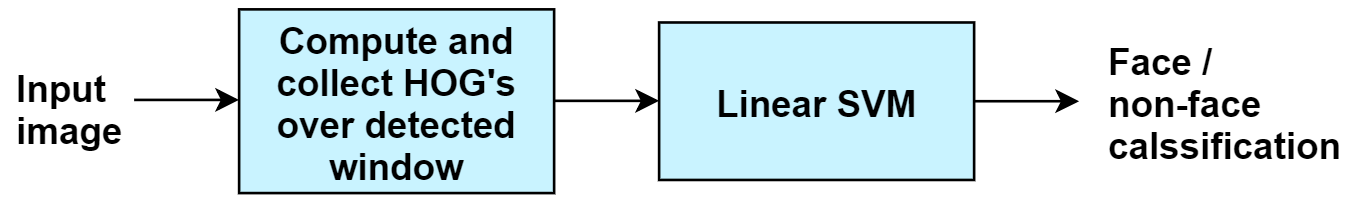


Figure 2. Facial detection using histogram of oriented gradients and support vector machine

After detecting all the faces in the image, we need to align and normalize them. This preprocessing step is crucial since a smaller public dataset is being used for pretraining the deep neural network that we use in the next step, in comparison to the main version of FaceNet which uses a much larger private dataset with about 200 times the size. We need to reduce the size of the input space by normalizing the faces so that the eyes, nose, and mouth appear at similar locations in each image [9]. For this task, an implementation of Kazemi’s face alignment algorithm is used that presents a method to precisely estimate the position of facial landmarks in a computationally efficient way [13].

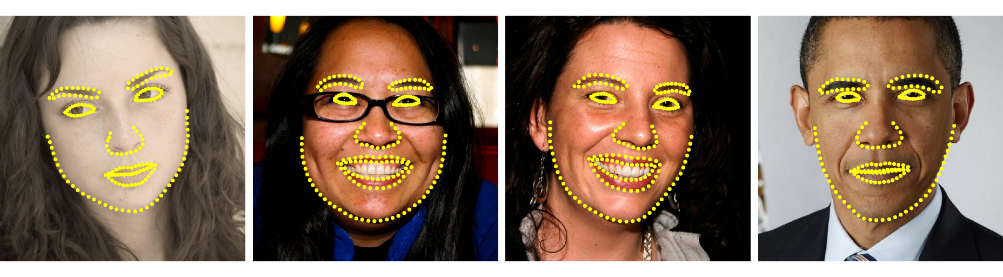


Figure . Selected results of Kazemi's facial landmark [13]

After detecting 68 facial landmarks, we use OpenCV [14] computer vision library to apply a simple 2D affine transformation, and finally resize the faces to 96x96 as suggested in FaceNet’s NN4 network which drastically reduces the CPU requirements [2].



Figure 4. Facial detection and normalization

**2.1.2. Facial Embeddings Extraction**

In this paper we use OpenFace which uses a modiﬁed version of FaceNet’s NN4 network presented in Appendix C that is based on the GoogLeNet [9, 15]. FaceNet is a unified embedding for facial recognition and clustering. It uses a deep convolutional network to directly map face images to 128 set of real numbers (128 facial embeddings) where distances directly correspond to a measure of face similarity. FaceNet is one of the well-known efforts in facial recognition with deep neural networks. It depends on a triplet loss function to compute the accuracy of the neural net classifying a face and can cluster faces because of the resulting measurements on a hypersphere [2].

Triplet loss is a learning algorithm for artificial neural networks where a baseline (anchor) input is compared to a positive (truthy) input and a negative (falsy) input. The distance from the baseline (anchor) input to the positive (truthy) input is minimized, and the distance from the baseline (anchor) input to the negative (falsy) input is maximized [16, 17].

In our case ‘anchor’ is the face of person A, ‘positive’ is another face of person A, and ‘negative’ is the face of a person who is not A. Selection of triplets is done such that samples are hard-positive (distance between the anchor and the positive exemplar is high) or hard-negative (distance between the anchor and the negative exemplar is low). Selecting the hardest negatives can, in practice, lead to bad local minima early on in training. In order to mitigate this, it helps to select such that

where is the anchor, is the positive face exemplar, and is the negative exemplar. These negative exemplars are called “semi-hard,” as they are further away from the anchor than the positive exemplar, but still hard because the squared distance is close to the anchor-positive distance [2].

OpenFace uses FaceNet model to pretrain a neural network which generates 128 facial embeddings that represent a generic face. By retraining the neural network using normalized student faces we generate 128 facial embeddings for each student, these numeric presentations of student faces will be used later in face matching and classification step.

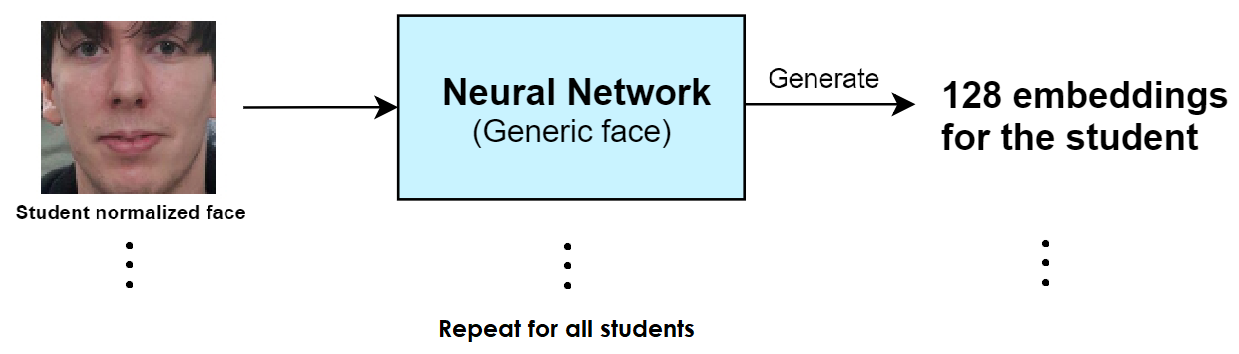


Figure 5. Generating 128 facial embeddings

**2.1.3. Face Matching and Classification**

In this step 128 embeddings of an unknown face will be generated and compared to all the previously generated embeddings of the students to produce a similarity score that indicates the likelihood that the face belongs to a known student. For this classification we use OpenCV’s linear support vector machine that can be used to match image features. The classifier will be trained using all the student’s face embeddings when the educator, using the app, sets-up the class at the beginning of the semester. Later to infer student attendance at the beginning of each class, facial embeddings of extracted faces from the group photo will be classified one by one using this classifier in a very short amount of time.

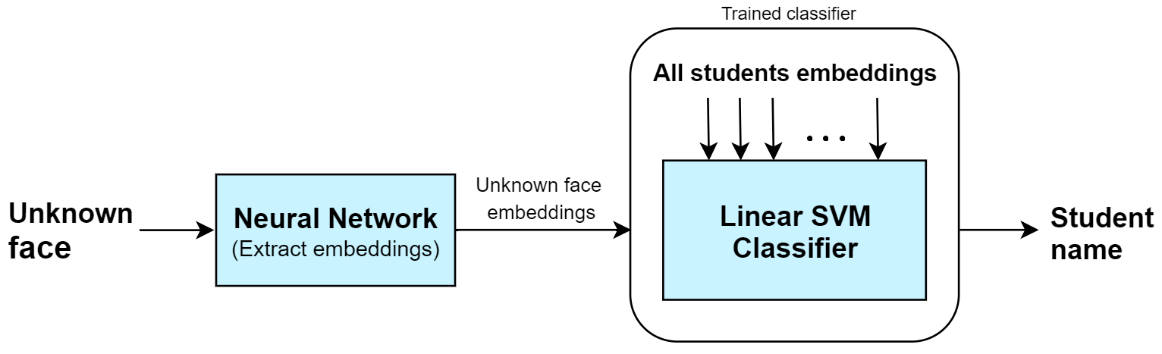


Figure 6. Classification and inference

**2.1.4. Image Data Augmentation**

One of the main problems that we faced in our work was the fact that we didn’t have enough photos of each student for feature extraction. We wanted the app to be easy to setup at the beginning of the semester. The educator would take only one photo of each student and initiate the setup. Retraining the neural network and generating facial embeddings for each student using only one photo resulted in immature facial embeddings that are not distant enough which would make our classification very inaccurate. To overcome this problem, we used multiple data augmentation techniques to generate a larger training dataset, 20 to 40 images depending on class size, for each student. Data augmentation is a technique to artificially create new training data from existing training data. Image data augmentation involves creating transformed versions of images in the training dataset that belong to the same class as the original image. Data augmentation has been shown to produce promising ways to increase the accuracy of classification tasks [18].

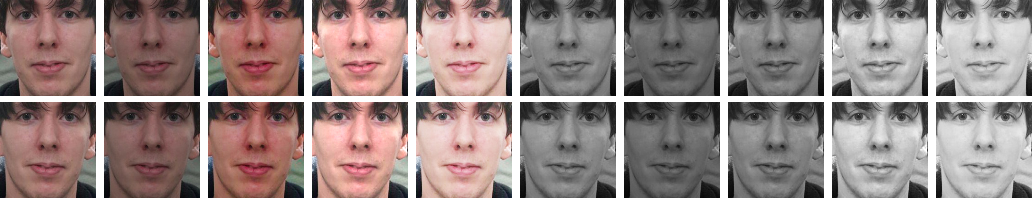


Figure 7. Image data augmentation examples

By increasing our student’s dataset size from 1 image to more than 20 images we managed to increase the accuracy by 561%. The significant increase in accuracy is due to the extremely poor performance of classification using only one student photo. This improvement in accuracy is a crucial step for us whose impact in the attendance tracking is evident in figures 8 and 9, for example. Using a training dataset of only one image per student in Figure 8 resulted a complete failure in attendance tracking. Whereas by using data augmentation and increasing the dataset size to 20 images per student, all the faces were correctly identified, and all the students were tracked as present, as shown in Figure 9.

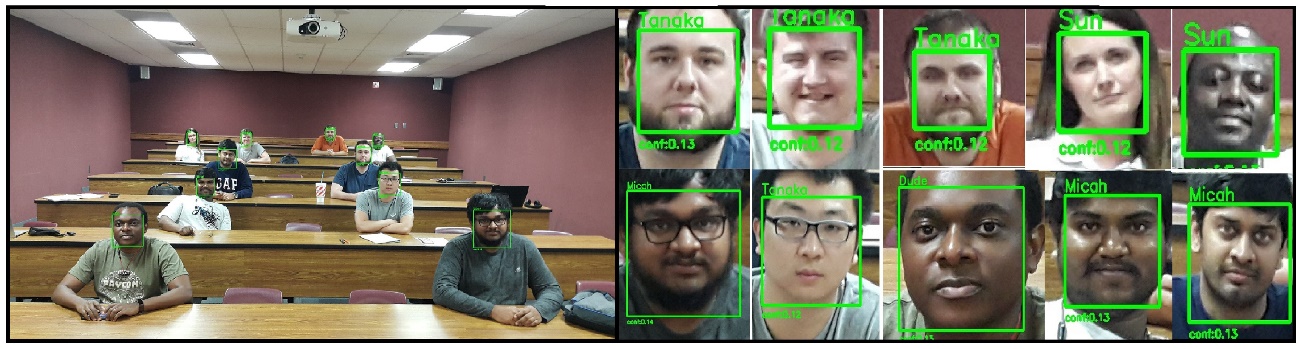


Figure 8. Attendance tracking results using single photo without data augmentation. All faces recognized incorrectly.

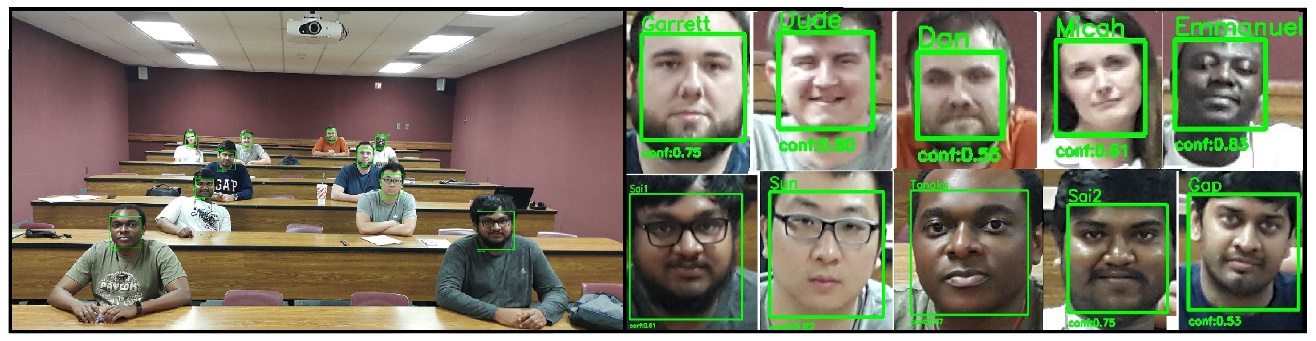


Figure 9. Attendance tracking using single photo with data augmentation. All faces recognized correctly.

**2.2. Backend API**

The mobile application communicates with the facial recognition engine using a backend API. The API is implemented using PHP and MySQL database. It provides the necessary automation and set of API routes for the app to trigger training and classification on the facial recognition engine. The API creates classes, receives images from the application and organizes them on the server.

Student attendance inferences are triggered when a group photo of the students in the class is received by the API. The API first infers the student attendance using the engine, then receives the results from the engine and stores the results on the server.

**2.3. Mobile Application**

Our mobile application is implemented using Ionic framework which provides tools and services for developing hybrid mobile applications that are executable on multiple platforms such as Android and iOS.

The app allows the educator to register, sign in, add a class (course) and set it up. Setting up a class involves entering the class information and taking a single photo of each student at the beginning of the semester. During the course of the semester, the educator can infer and record student attendance by selecting the course and taking a group photo at the beginning of each class.

To increase the accuracy of student attendance tracking we use a reconfirmation mechanism in the app for low confidence recognitions. This feature allows educators to modify and reconfirm low confidence facial recognitions in the attendance tracking record. By doing so our model will be improved overtime and educators will have a more accurate attendance record.

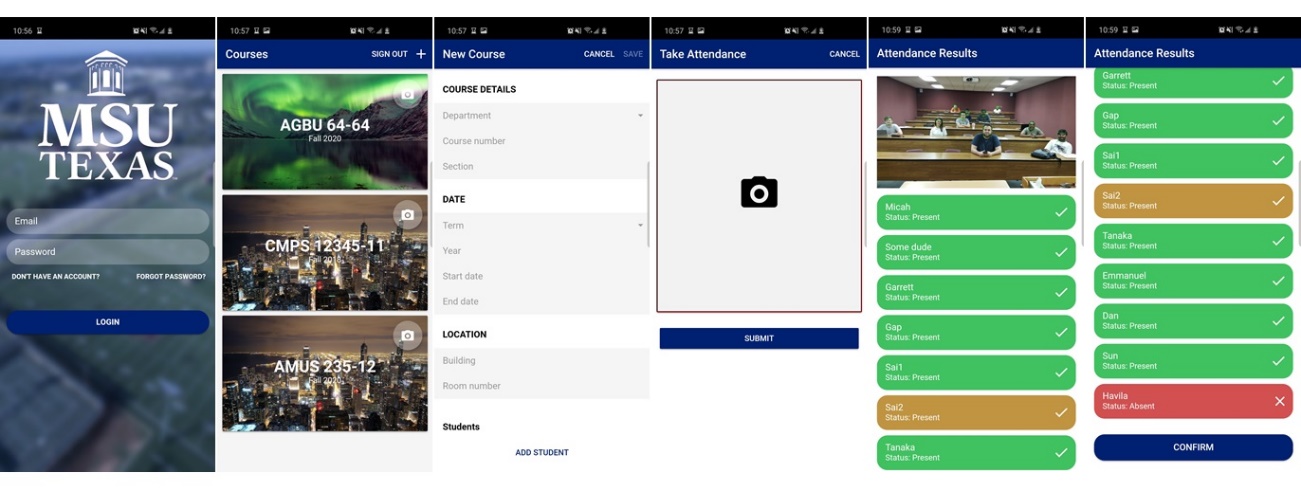


Figure . The mobile application

**3. Experiments**

We’ve recognized 207 faces correctly out of 214 faces so far which gives us an accuracy of 96.7%. By using our reconfirmation mechanism in the app for low confidence recognitions, educators can achieve up to 99% accuracy in student attendance tracking.

Pictures were taken in class sizes varying from 11 to 16 students. In our experiment we used 2 smartphone devices with 12- and 16-megapixels cameras. To expand our experiment, we generated 2 other datasets by decreasing the resolution of the taken pictures down to 8- and 4-megapixels. For each attendance tracking inference, students were asked to look at the camera. Classrooms had average lighting of 500 lux, and the cameras’ flashlights were off. We observed consistent results with both devices and the low-resolution generated datasets. However, we believe the higher resolution cameras may be able to improve the facial recognition confidence when dealing with larger classroom sizes; particularly referring to the students at the back of the room. These consistent results, using different camera resolutions, confirms that the app will perform reliably on most smartphones.

**4. conclusion**

Despite the challenges associated with attendance tracking, facial recognition provides a very good solution. By simplifying the process of ensuring that students are present in class, BBATS increases the speed and accuracy of a task that educators world over must deal with on a daily basis. Our model proves that the process can be completed much faster without any additional hardware. The global prevalence of smartphone use ensures a smooth transition for educational institutions to utilize the efficiency that BBATS provides.

**References**

*[1] Daniel Sáez Trigueros, Li Meng, Margaret Hartnett. Face Recognition: From Traditional to Deep Learning Methods. arXiv:1811.00116, 2018.*

*[2] FaceNet: A Unified Embedding for Face Recognition and Clustering, Florian Schroff, Dmitry Kalenichenko, James Philbin, CVPR 2015*

*[3] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “n,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1701–1708, 2014*

*[4] Matthew Turk and Alex Pentland. Eigenfaces for recognition. Journal of cognitive neuroscience, 3(1):71–86, 1991.*

*[5] Peter N Belhumeur, Jo˜ao P Hespanha, and David J Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 19(7):711–720, 1997.*

*[6] Timo Ahonen, Abdenour Hadid, and Matti Pietik¨ainen. Face recognition with local binary patterns. In Computer vision-eccv 2004, pages 469–481. Springer, 2004.*

*[7]* *Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. Proceedings of the British Machine Vision, 1(3):6, 2015.*

*[8] Xiang Wu, Ran He, and Zhenan Sun. A lightened cnn for deep face representation. arXiv preprint arXiv:1511.02683, 2015.*

*[9] B. Amos, B. Ludwiczuk, M. Satyanarayanan, "Openface: A general-purpose face recognition library with mobile applications," CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016.*

*[10]* *Davis E. King. Dlib-ml: A Machine Learning Toolkit. Journal of Machine Learning Research 10, pp. 1755-1758, 2009*

*[11] Dalal, N. and Triggs, B (2005) Histograms of Oriented Gradients for Human Detection. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), 1, 886-893.*

*[12] Cortes, Corinna; Vapnik, Vladimir N. (1995). "Support-vector networks". Machine Learning. 20 (3): 273–297. CiteSeerX 10.1.1.15.9362. doi:10.1007/BF00994018*

*[13]* *Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1867–1874, 2014.*

*[14]* *Pulli, Kari; Baksheev, Anatoly; Kornyakov, Kirill; Eruhimov, Victor (1 April 2012). "Realtime Computer Vision with OpenCV". Queue. pp. 40:40–40:56. doi:10.1145/2181796.2206309*

*[15]* *Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–9, 2015.*

*[16] Chechik, G.; Sharma, V.; Shalit, U.; Bengio, S. (2010). "Large Scale Online Learning of Image Similarity Through Ranking" (PDF). Journal of Machine Learning Research. 11: 1109–1135.*

*[17] Schroff, F.; Kalenichenko, D.; Philbin, J. (June 2015). FaceNet: A unified embedding for face recognition and clustering. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 815–823. doi:10.1109/CVPR.2015.7298682. ISBN 978-1-4673-6964-0.*

*[18] Perez, L., & Wang, J. (2017). The Effectiveness of Data Augmentation in Image Classification using Deep Learning. Arxiv.org. Retrieved 21 May 2018, from https://arxiv.org/abs/1712.04621*