Journal of Theoretical and Applied Information Technology
15th March 2020. Vol.98. No 05

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

FACE RECOGNITION FOR ONLINE USERS' AUTHENTICATION

[1] FIRAS AJJOUR, [2] DR. MHD BASSAM KURDY

[1] Research Scholar, Department of Web Sciences, Syrian Virtual University, Syria

[2] Professor, Department of Artificial Intelligence, Syrian Virtual University, Syria

E-Mail: [1] Firas_20190@Svuonline.Org , [2] T_bkurdy@Svuonline.Org

ABSTRACT

In the last decade, advancement in Artificial Intelligence attracted a lot of experts that lead to massive growth and advancement in all human life aspects. Therefore, one of the key fields to point at, which attracted a lot of attention and development lately, is Face Recognition.

In recent years, Face Recognition tends to be one of the most widely used technologies in many different domains and workspaces, such as emotional recognition, security, health sector, marketing, and retail, etc.

this approach will consist of an online system with real-time functionality (close to real-time), that will be responsible for the declaration of users to be recognized later. Based on the recognition results, the system will then grant the users the needed authentication.

In this research, various different challenges related to the development and the use of Face Recognition, including the variations in light conditions, camera resolution, processing power, facial changes over time, number of users to be recognized, etc...

During this work, "Viola and Jones" and "MTCNN" were used for face detection, and "FaceNet" was applied for facial features extraction. Also, similarity neural network (Similarity Net) has been created to regress similarity percent between user's features' vectors, beside it has been trained on user's features by exploiting the Euclidian distance between embeddings.

This approach was tested on a group of datasets - personal, Kaggle and LFW dataset. The tests returned 100% successful recognitions on personal and Kaggle dataset, and 99.5% on LFW dataset.

KEYWORDS: *Face Detection, Face Recognition, Facenet, MTCNN (Multi-Task Cascaded Convolutional Neural Networks for Face Detection), Embeddings, Feature Vectors, Kaggle (Website for AI Contests), LFW (Labelled Faces in The Wild), CNN (Convolutional Neural Network), CLAHE (Contrast Limited Adaptive Histogram), Histogram Equalization, Face Authentication.*

1. INTRODUCTION

Recently, due to higher accuracy and efficiency artificial intelligence provides while dealing with data, extracting information and forecasting, it received great attention from tech communities around the world. In the era of information technology, privacy and data protection has become a crucial matter, especially during the vast shift of people reliant on technology and mobile devices. With that being said, data privacy is exposed to high potential risk, therefore, action must be taken to find and create solutions to protect user data, such as choosing complex passwords and provide the users ways to manage and remember their passwords. This is why Face Recognition is considered as one possible way that may help in identifying and fixing those risks and creating solutions for them.

Authenticate users using Face Recognition is also known as Face Authentication. It is worth

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

noting that the Face Authentication domain is viewed as a very challenging domain considering its sensitivity. According to [1], Face Recognition can be declared in two classifications:

- The process of finding and tagging a face as a "known" or "unknown".
- The process of finding a user identity.

In this research, the focus is on the confirmation process. However, the proposed approach could also apply to finding user identity. In fact, Face Recognition process is the same order of steps: Image pre-processing, face detection and then face recognition.

Digging deep into details, major problems arise in the face recognition process, which are defined in [1], [2] and [3]. These problems include complex backgrounds, change in facial expressions, poses, facial occlusion, user aging, camera resolution, number of users, and illumination variation [4], which could all prevent and affect the face detection and recognition process.

In [5], [6] and [7], researchers have used the deep convolutional neural networks to achieve robust face detection that can resist illumination, occlusion, pose variation, blur and expression variation. On the other hand, in [8], [9] and [10], due to its capabilities of collecting highly detailed features of a human face, researchers have studied the application of CNN in the Face Recognition stage especially after being trained on millions of faces. The good news is that deep learning can help us solve many of the previously mentioned problems, but the bad news is the challenge involved in the need for a powerful processor.

In this research, non-controllable limitations have been faced such as:

- Working online with a large number of users.
- Considering that every user should be processed in the least time (as close as possible to real-time).
- Internet malfunctions (slow and unreliable connection).
- Absence of control over client's computer which forces us to take into consideration a minimum assumption for the processing power.

In conclusion, a system was being designed, that can deal with the previously mentioned problems, which returns very good results. In summary, the main contributions of this paper include:

- Applying "Viola and Jones" algorithm [11] with "MTCNN" [12] to achieve the best results of face detection, while minimizing the errors of face detection.
- Applying "Viola and Jones" on the client-side, to avoid the need for a high processing power on client's computer.
- Exploiting the frontal faces classifiers of "Viola and Jones" to detect only frontal faces, to avoid compromising the accuracy of the proposed approach by using partially hidden faces, besides using the detected faces on the client-side to reduce the amount of data sent over the internet (by sending the cropped faces only).
- Implementing "FaceNet" model [13] to extract features vectors out of faces' images. Using 512D embeddings to form features vectors of users' faces. As a result, sensitive data has been collected for similarity algorithm.
- Implementing regression neural network named (Similarity Net) to evaluate the percent of faces similarities, in order to exploit the dynamic of the neural network to solve the clustering problem.
- Developing a methodology to exploit "adaptive threshold" algorithm [14] to enhance the clustering process and enhance the identity confirmation decision.
- Forming an algorithm to train and re-train similarity networks by exploiting the "adaptive threshold" weaknesses and dynamics of the neural network, to avoid doing training for all users at once.
- proposing a real-life applicable approach, that can provide a security system which uses face recognition to secure users' accounts by their faces.

2. RELATED WORK

2.1. Preprocessing

The most efficient preprocessing method is the illumination variation handling. According to [2], there is no decisive solution for illumination handling, but decreasing its effect by using some statistical methods. In [4], researchers proposed that using color components in an image can limit the light effect, so using "Contrast Limited Adaptive Histogram" (CLAHE) can normalize the color of the image effectively, as shown in Figure 2and Figure 4. Histogram equalization is usually used to handle illumination. It applies its transformation to every pixel in the image, while transformation has been

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

extracted from the total image histogram. This method is generally effective, unless there is an illumination change peak in image histogram, then the problem of Histogram equalization will be worse, as shown in Figure 2 and Figure 3 . Another method developed to handle this problem was called "Adaptive Histogram Equalization" (AHE), where the transformation method is applied to every pixel according to neighboring pixels. However, this doesn't solve the problem completely. This is when the application of "CLAHE" can be useful by using a mix of both (equalization histogram and AHE) by applying the histogram equalization transformation method on small pieces of the image, piece by piece, as shown in Figure 1; so the effect of illumination will be handled in its region [15][16].

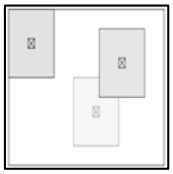

Figure 1. Histogram - regions

Figure 2. original image without any processing

Figure 3. applying global equalization

Figure 4. applying CLAHE

2.2. Face Detection

Face detection is the first step in the authentication process, and many types of research were made in this domain. One of the best researches is "Viola and Jones" algorithm [11], that can work in real-time with low processing power computers, although this algorithm still has its weaknesses. MTCNN [12] gives better detection results than "Viola and Jones", but this method requires higher processing power.

2.2.1. viola and jones

According to [11] , "Viola and Jones" algorithm is able to handle a high rate of frames (about 15 frames per second) on an old computer (700 MHz intel Pentium 3), using (AdaBoost) as a learning algorithm that extracts critical features from a large set of potential features, and generates the integral image which is a calculation of the original image that makes features extraction very fast.

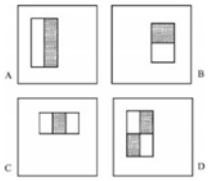

Figure 5. types of predesigned features extractors

Features extraction is done by predesigned shapes with their calculation as shown in Figure 5, A and B example of Vertical and horizontal two-rectangle feature, C example of horizontal three-rectangle feature, D example of four-rectangle feature. This algorithm has classifiers to classify the extracted features and decide whether this image is positive or negative. These classifiers are being learnt by the "AdaBoost" algorithm which is considered to be a good algorithm to define a small number of effective features from a set of possible features, as shown in *Figure 6*.

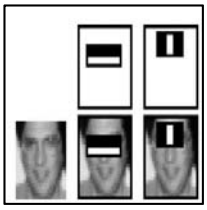

Figure 6. the first and second selected features by AdaBoost

After training the classifiers, they are combined in a cascaded form to increase the speed of the algorithm's processing, then the sub-windows of the image are processed by those classifiers, as shown in

Figure *7*. So, the algorithm can eliminate the sub-windows and regions classified as "not containing the needed features".

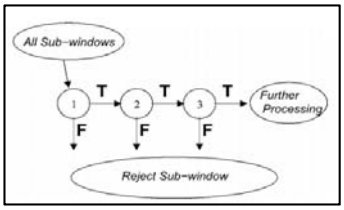

Figure 7. cascaded features classifiers

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

This algorithm according to [11] was tested on a dataset of 23 images with 149 faces and it returned an accuracy of detection up to 94.1%.

2.2.2. multi-task cascaded convolutional neural network

In [12], researchers proposed the use of CNN (Convolutional Neural Network) with cascaded tasks to enhance the performance of face detection. "MTCNN" was formed to work in three cascaded phases as in

Figure *8*, whereas each phase used a designed CNN to detect special points in the human face. The first phase uses Shallow CNN called "Proposal network P-Net" which finds the candidates' windows that may contain faces. The second phase applies "Refine Network R-net" to get confidence of each window, where windows with higher confidence will be chosen and windows with false candidates will be rejected. Then, a more complex network called "Output Network O-Net" is used to refilter the remaining windows and only keep the windows containing faces, and it also finds the facial landmark of every window. At this point, every remaining window should be defined as a face. In every phase, the algorithm performs calibration with bounding box regression and conducts non-maximum suppression to regress the features in every window and remove crossed windows.

Using three types of CNN in this algorithm will affect its performance badly, however using fewer filters in every network will recompense the performance leak. Researchers used filters 3*3 to 5*5 that limited the ability of CNN to do classification, but since a binary classification was needed, they decided to use these filters to enhance the speed of the algorithm, which resulted in an algorithm accuracy of 95.4%.

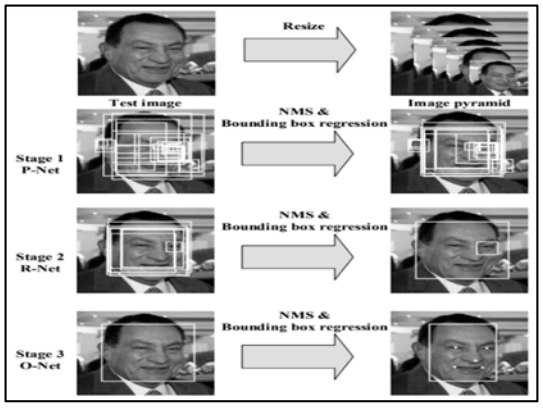

Figure 8. MTCNN processing steps of

2.3. Face Recognition

In this research, obtaining too many images of a user was not an option, because as mentioned earlier, the approach would work over the internet (which may be unreliable), and the most amount of user's images available was around 20 images, and these images may have posed and illumination variation. Therefore, using "One-Shot learning" [17] was the best option. As shown in Figure 9, in order to do the recognition using this technique, extracting features' vectors then a comparison have to be done to decide if there is a match.

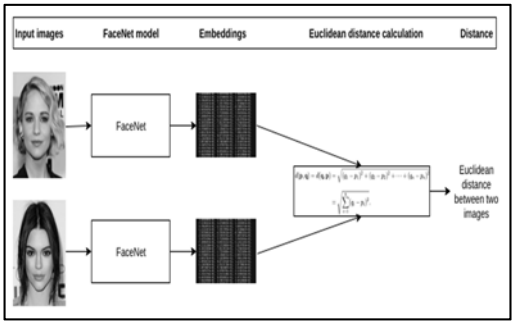

Figure 9. One-shot learning process

2.3.1. Facenet Model

In [13] the researchers proposed FaceNet Model, which is a convolutional neural network trained to extract features of faces by working directly on the face image pixels without the need to pre-designed features to be grabbed out of the faces. "FaceNet" extracts facial features automatically and converts them into the vector of 128, 256, or 512 dimensions. It was tested with a classification network and a triplet loss algorithm on LFW dataset,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

and it achieved a recognition accuracy up to 99.63% by using embeddings features vectors of 128D.

When this algorithm was tested, researchers used a Triplet loss algorithm to enhance similarity decision based on the distance of embeddings. The algorithm assumed that the shortest distance between any two persons should be bigger than the largest distance of two vectors of the same person as explained in Equation 1 and Figure 10.

$$\left\| f\left(x_i^a\right) - f(x_i^p) \right\|_2^2 + \alpha < \left\| f\left(x_i^a\right) - f(x_i^n) \right\|_2^2$$

Equation 1. Triplet loss

α: is the margin to be added to the distance between p (positive features vector) and n (negative features vector).

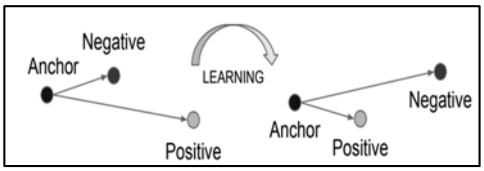

Figure 10. Triplet loss idea

In [13] CNN was used with this network as a black box to get the vectors that represent the features of the face, a triplet loss was applied on small groups of users' embeddings (Features' Vectors) to train the classification network, as shown in Figure 11.

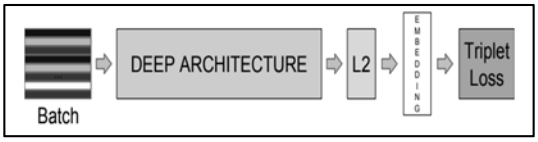

Figure 11. FaceNet Process

3. METHODOLOGY

This paper propose using "Viola and Jones" [11] for face detection on client-side, and MTCNN [12] to confirm face detection on server-side, and FaceNet [13] with the help of Inception ResNetV1 [18] trained on VGGFace2 [19] which generates 512D features vector. All that gives us solid ground to create the algorithm, to apply Euclidian distance to define user's thresholds according to [14], which used later to nominate identities of the user based on comparing the Euclidian distances of users' features vectors with adaptive threshold of each user. When using this technique some mis-identifications happen as in [14], this leaves us to figure out a technique to handle clustering weaknesses, and do identity confirmation. Therefore, as a solution, proposing regression neural network named (Similarity Net) as an identity confirmation, this network will be created for each user and trained on the user features' vectors and his most similar users' embeddings.

3.1. Similarity Neural Network (Similarity Net)

This network is a regression neural network [20] created to calculate the similarity between features vectors. it consists of an input layer, 512 neurons, output layer, 1 neuron, and one hidden layer, 64 neurons. This network is fully connected and trained using Adam optimizer [21] and also uses Mean Square Error (MSE) as a loss function. It designed to be trained by a small number of features vector.

The aim is to use this network as a second identity confirmation. This network will be trained online while a new user is registering or updating the user facial features on the system. At this time, an instance of this similarity network will be created and trained on this user's embeddings so it can be used again when the user tries to login next time. This is when the network will calculate the similarity and give the identity confirmation decision.

In this section of the approach, failures of using Euclidian distance and adaptive threshold method (proposed in [14]) were exploited to collect the user features and the user's "similar users' features" (users' who look alike the user we are identifying), in order to train the similarity network as in Figure 12 and Figure 13 , (this process will be explained in-depth in the next section).

Journal of Theoretical and Applied Information Technology
15th March 2020. Vol.98. No 05

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

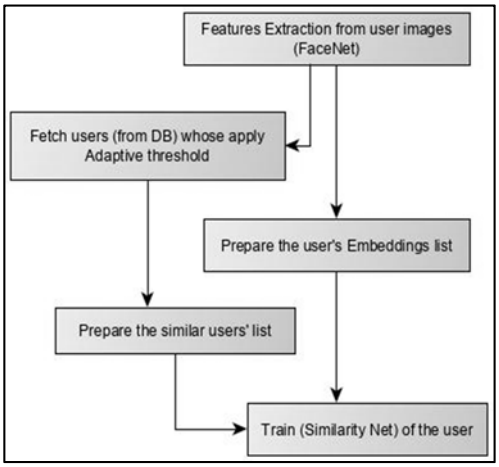

Figure 12. train similarity network of a user

```
def SetUserThreshold(crnt_user):
    global Syste_users_changed
    # looping the users in System
    for usr in System_users:
        # Making sure not to Get distance of user with himself
        if usr[ConstVars.USER_NAME] != crnt_user[ConstVars.USER_NAME]:
            #Finding the shortest distance between two users
            dist = getNearestFeatureDistance(usr, crnt_user)
            #Compare the shortest distance with adaptive threshold
            if dist < crnt_user[ConstVars.DYNAMIC_THRESHOLD]:
                crnt_user[ConstVars.DYNAMIC_THRESHOLD] = dist
                # create new dictionary for close_user to get just what we need
                closeUser = {ConstVars.USER_NAME: usr[ConstVars.USER_NAME],
                             ConstVars.FEATURES: usr[ConstVars.FEATURES]}
                crnt_user[ConstVars.CLOSER_USERs] .append(closeUser)
                for f in usr[ConstVars.FEATURES]:
                    crnt_user[ConstVars.FALSE_FEATURES].append(f)
                    crnt_user[ConstVars.TRAINED] = False
            if dist < usr[ConstVars.DYNAMIC_THRESHOLD]:
                usr[ConstVars.DYNAMIC_THRESHOLD] = dist
                # create new dictionary for close_user to get just what we need
                closeUser = {ConstVars.USER_NAME: crnt_user[ConstVars.USER_NAME],
                             ConstVars.FEATURES: crnt_user[ConstVars.FEATURES]}
                usr[ConstVars.CLOSER_USERs] .append(closeUser)
                for f in crnt_user[ConstVars.FEATURES]:
                    usr[ConstVars.FALSE_FEATURES].append(f)
                    usr[ConstVars.TRAINED] = False
                Syste_users_changed = True
```

Figure 13. Python Code for adaptive threshold

3.2. New User Registration

The main purpose of this approach is securing the web application or the web service using face recognition. That means the system has to use face recognition to authenticate a big number of users. These users are not listed in a predefined list; however, they are making registrations regularly. Therefore, using the neural network as a classifier to classify users as in [13] is not an option, since retraining a classifier neural network from scratch when a new user registers is not the best option, it will simply be a waste of time and processing power. Instead, the shifting toward the one-shot learning technique as in [17] and developing a process shown in Figure 14, was very useful to achieve the goal of this research.

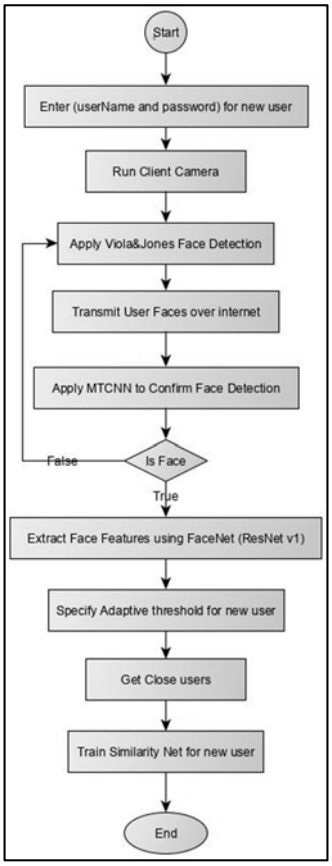

Figure 14. register (declare) new user process

The process starts by a client-side face detection using "Viola and Jones" [11] implemented using JavaScript library PICO.JS [22], then confirms face detection using MTCNN [12] on the server-side, where more processing power is available to apply CNN for more accurate results. Our next step includes features extraction for every image of the user, using FaceNet [13], to apply clustering, one of the most effective ways, endorsed by [13], is the use of Euclidian distance, but this raises a question: what is the best threshold to use?

According to [14], using adaptive threshold is more effective than using fixed threshold, so the next step in the process is to define the threshold of this user by comparing the user embeddings with the saved users (saved in database) and get the minimum distance as a threshold, as in Equation 2.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$X_u = \min(\int_i^0 d(e_i, \mathrm{u}))$$

Equation 2. adaptive threshold

X_u: is the adaptive threshold of the user (u), $d(e_i, \mathrm{u})$: the distance function between the saved user with index (i) from the user (u) whom calculating the threshold is for. During the process of finding the "Adaptive threshold", three points should be considered (Figure 13):

- When threshold of any saved user (users in the database) is larger than the distance between the saved user and the new user, the saved user's threshold has to be changed to be equal to that distance value.
- When the saved user's threshold changes (from the previous step), the new user's embeddings have to be added to the saved user's similar list (the list holds the users look alike the saved user), beside flagging the saved user's similarity network to be retrained on the next login.
- When the new user's threshold changes (from the first step), saved user's embeddings should be added to the new user's similar list (the list holds the users look alike the new user), beside flagging the new user's similarity network to be retrained on the next login.

Finally, after collecting the new user lists (the user embeddings list, similar users' list of the new user), these lists should be passed to the user's similarity neural network to be trained (Figure 15).

The power of the similarity network is in its ability to be trained only on users with similarities (Distance smaller or equal to the adaptive threshold) and learn how to find differences in features. For the other saved users (who's not similar to the new user), confirming the dissimilarity can be done by using Euclidian distance and adaptive threshold.

As shown, every user in the proposed approach has his own trained similarity neural network which will be used as a confirmation step to the clustering algorithm on top of the similarity network that already does the job of triplet loss in [13]. There will be no need for a global network to be trained and re-trained to achieve this task. There are some possible cases where retraining the similarity network of a user is necessary whenever the user updates his features, or when a similar user is registering or updating his embeddings, then, a the user similar users' list will be updated with the new embeddings, and therefore retraining similarity network for updated users is needed (Figure 13 and Figure 15).

```
def trainUserNetwork(user):
    while(not user[ConstVars.TRAINED]):
        # create Similarity Net model
        confirm = Confirmer()
        # False_Features: the features (Features Vectors)
        #       of the close users to this user
        # Features: the features (Features Vectors) of this users
        confirm.train(user[ConstVars.FEATURES], user[ConstVars.FALSE_FEATURES])
        # Saving the Network weights
        user[ConstVars.NN_MODEL_WEIGHTS] = confirm.getModelWeights()
        user[ConstVars.TRAINED] = True
        # testing the Similarity Net (if it's trained well)
        for u in user[ConstVars.CLOSER_USERs]:
            sim = getSimilarity(user, u)
            if sim >= 0.75:
                # if the similarity with any of the close users > 75%
                # we have to adjust the data and retrain the network
                for f in u[ConstVars.FEATURES]:
                    # we adjust the data by duplicate some features
                    # of the close user who cuase the problem
                    user[ConstVars.FALSE_FEATURES].append(f)
                user[ConstVars.TRAINED] = False
                break
```

Figure 15. Python Code for training Similarity Net of a user

3.3. User Identification

User Identification can be described as the core of the proposed approach because it returns the authentication decision, so making sure that no misidentification will be done. The process starts on the client-side using "Viola and Jones" face detection then transfers user face images to the server-side. Next, the face detection is confirmed using MTCNN. Features will be extracted using FaceNet as embeddings. Later, a search for a candidate identity of the saved users is done by checking the distance of the user being identified with every saved user, and accept only the users whose their distance is smaller or equal to their thresholds, according to Equation 3.

$$f(C) = \int_i^0 t(U_i) \geq d(U_i, n)$$

Equation 3. find identity candidates

$f(C)$ function to find candidate identities, $t(U_i)$ Euclidian threshold of user U with index i, $d(U_i, n)$: the Euclidian distance function between user U_i and n the user we are identifying.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

After fetching candidates, testing the user's features with every candidate's similarity neural network is needed. Now, eliminating every candidate with a similarity percent lower than (97%) and choosing the best similarity ratio.

To reduce complication of processing, especially for real life application, the identification process should start by getting user name and capture a face image in order to apply face recognition and authenticate the user. The process will not search for candidates using Euclidian distance and adaptive threshold, but fetching the claimed identity using username and then doing the first check by comparing Euclidian distance with adaptive threshold of the claimed identity. then, a second check will be done using similarity Network to get the similarity ratio, if the ratio is equal or greater than 97%, the identity will be confirmed and the user will be authenticated, otherwise the user will not be authenticated Figure 17 and Figure.16.

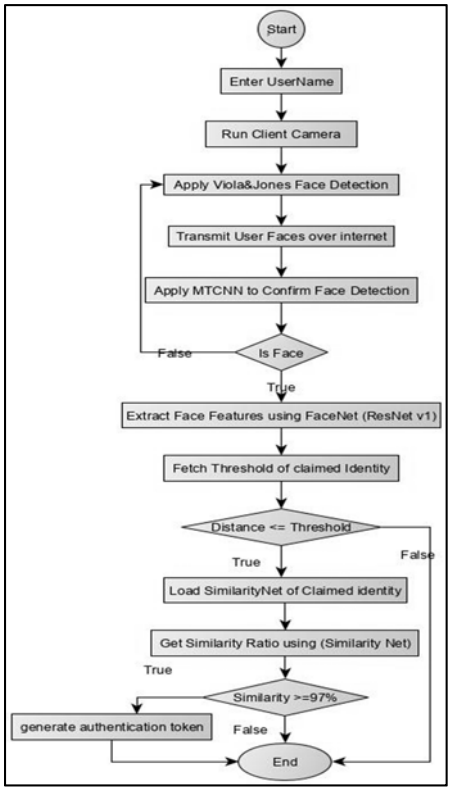

Figure.16: Identify user process

```python
def CheckIdentity(crnt_user):
    for u in System_users:
        # find the user by username
        if u[ConstVars.USER_NAME] == crnt_user[ConstVars.USER_NAME]:
            # calculate Euclidian Distance between
            # features we got from client and found user
            dist = getNearestFeatureDistance(u, crnt_user)
            # Check if shortest distance smaller than found user's
            #       adaptive threshold
            if dist <= u[ConstVars.DYNAMIC_THRESHOLD]:
                # calculate the similarity between found user
                # and features vectors from the client
                # using found user's similarity Net
                prediction = getSimilarity(u, crnt_user)
                print("similarity= ", prediction)
                # checking similarity threshold
                if prediction >= 0.97:
                    # logedIn succeeded and sure
                    return 1, prediction
                else:
                    # face recognition problem
                    return 0, prediction
            else:
                # face recognition Problem
                return 0, 0
    # userName does not exist
    return -1, 0
```

Figure 17. Python Code for Check identity of a user

3.4. Updating User Face

If face recognition is done and identified the face correctly, many factors may lead to not confirm user identity in the future, because of facial accessories, aging, facial hair, and so on.

this research suggests a process that may be applied by using user credentials (username and password) when recognition or identity match fails. The process starts by checking user credentials, extracting facial features, comparing them with existing users (excluding the user being updated) and then updating the threshold where new user's features and similar users' embeddings lists is provided. Last step, updating the previously mentioned lists with the new user's features and redo the training for the user's "Similarity Net". Steps are shown in Figure 18.

Journal of Theoretical and Applied Information Technology
15th March 2020. Vol.98. No 05

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

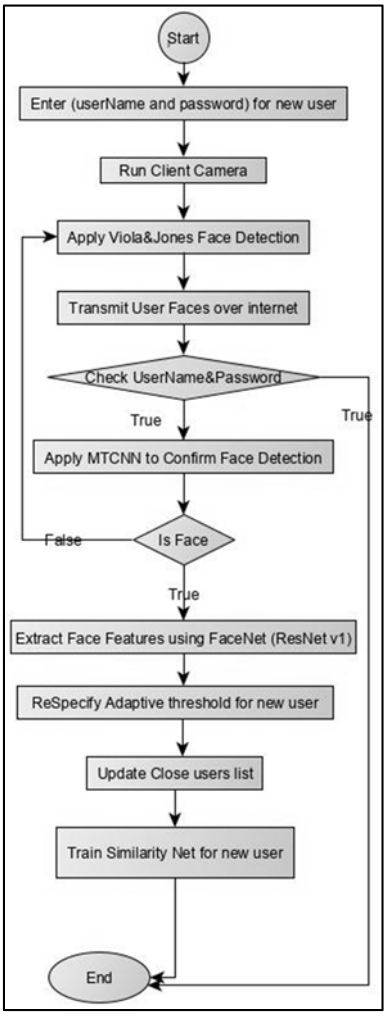

Figure 18. Update user's features process

4. EXPERIMENTS

To test the proposed approach, multiple datasets containing photos/videos of users were used. Next, splitting these users' photos/videos into two parts, the first one is used to test the registration process, while the second is used to test the authentication process. The goal in this research paper is to do an identity confirmation (checking claimed identity), So, the user's name and image was used to fulfil the goal of this test. If both of them passed, the user would be defined as "Authenticated", otherwise the user would be defined as "Not Authenticated", as shown in Figure.16 and Figure 17.

To test the "Update User Face" process, identity confirmation was applied, if it failed, then in this case starting the "Update User Face" process will take place. If more than three attempts return unsuccessful authentication, it will be considered as an un-authenticated user, otherwise the user will be an authenticated user.

The testing processes used a personal laptop with a CPU (2.6 Core i7) and (8G RAM). Also, no GPU used in these tests. The approach was applied on windows10 64bit and python3.5.6 based on Anaconda framework with Tensorflow1.10.

to prove the concept, a demo was created (web pages and web service) using python Flask with SocketIO to open real-time connection between client-side and server-side, to send user images instantly, and start processing during the capturing of the user's images (Figure 19, Figure 20 and Figure 21).

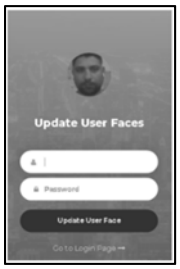

Figure 19. update user's face's features

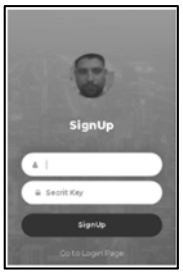

Figure 20. sign-up for new user

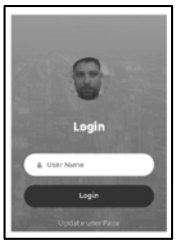

Figure 21. Authenticate a user

5. DATASETS

three types of datasets were used to test the proposed approach. The first one simulates the real processes of user login and sign-up, the second one challenges the abilities of the approach to handle bigger number of users comparing to the processing power, especially while using low-resolution images, and the third dataset tests the accuracy when dealing with dataset not suitable to this research's

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

case study in many ways like (huge number of users, lack of user's photos and significant changes in the shooting conditions of the user). Therefore, the following datasets were chosen:

5.1. Personal Dataset

A dataset for 25 users (male and female), each one had two 30 seconds videos, shot by a 1.3 Megapixel laptop webcam. The first video was used to register the user, while the second one to identify him. Figure 22 shows samples of the collected videos, with some changes in shooting conditions, considering the skin color variations (Dark and white skin).

Figure 22. samples of personal Dataset

5.2. Kaggle Dataset

This dataset [23] contains 391 users (male and female) with each user having 20 images downloaded from the Kaggle website. Each image is (180 * 200) pixels with a resolution of (96 dpi) Figure 23. user images were divided into two groups of 10 images. The first one for the registration process and the second one for the identity confirmation process. This dataset is very challenging, because it contains faces with different shooting conditions (light and poses conditions) and many facial accessories. Also, the images were shot with a resolution close to a webcam but with smaller dimensions.

Figure 23. Samples of Kaggle dataset

5.3. LFW Dataset

Labelled Faces in the wild [24]: is a data set for 5749 users (male and female), each user has an undetermined number of low-resolution images, taken in an unconstrained condition. Total images in the dataset are (13,233), only 1,680 users have two or more images. The images were divided into two sets for each user. The first set for the registration process and the second set for the identification process; however, some users had only one image used for registration and identification processes.

This dataset is a good test for feature extraction algorithm, and the algorithm's performance to handle the big number of users (according to the processing power used).

6. RESULT AND DISCUSSION

6.1. FaceNet vs Adaptive threshold vs proposed approach

According to [13] FaceNet model able to generate features' vectors (embeddings), which can be used in both classifications and clustering problems, the algorithm proposed in [13] used FaceNet model as black box to extract and train classification neural network to classify embeddings and do identification, the most important part of their algorithm is triplet loss, because it enhanced the accuracy of the algorithm, so the classification neural network trained using triplet loss to classify users' images, which means all users' images and information already available, but in this research it's not, so we have to focus on clustering to solve this problem, and we cannot apply triplet loss

because not all users information available at once at the beginning.

In [14] researchers proposed using FaceNet model to extract embeddings beside using clustering algorithm to do identification, and adaptive threshold to enhance the accuracy of identification process. But in [14] the researchers dealt with the same case study of this research, so they moved to clustering algorithm without using triplet loss. Not using triplet loss reduce the accuracy of identification more than 23% as shown in

Table 4.

This research, propose a specific solution for the defined scenario and its problems, which are protecting users over internet by using face recognition, so using "Viola and Jones" [11] for client side face detection without consume client processing power, confirm face detection using "MTCNN" [12], for face recognition stage, this approach used clustering algorithm to do identity confirmation beside apply the Adaptive threshold idea as proposed in [14], but to avoid the lack of accuracy faced in [14], regression neural network named (Similarity Net) proposed as an alternative to triplet loss, by create (Similarity Net) for every user in the system, and train this network using the user's embeddings and the embeddings of the most users look alike the user in the system, in other words the Similarity Net learn how to distinguish the user from his similar users, so this paper propose an alternative to triplet loss, which could be apply for clustering problems where not all data exist at the starting point.

6.2. Proposed Approach Evaluation

Firstly, let's consider the following factors:

- Average Declaration Time (ADT): The average time in seconds used to create a user in the system.
- Average Authentication Time (AAT): The average time in seconds used to authenticate a user in the system.
- Average of Similarity Percent (ASP): The average of identification percent of all accepted users. According to the applied constraints, the ASP should be between 97% and 100%, any user below 97% similarity was not authenticated.
- Authenticated Percent (AP): The percent of authenticated users of the total dataset.

The evaluation process created to simulate real life apply of the proposed approach, by iterate the dataset users one by one loading the footage of the user from the dataset, then apply face detection using "MTCNN", extract embeddings from detected faces using "FaceNet" model, then "Adaptive threshold" will be calculated for the user, finally similar users collected from the system to train "Similarity Net" of the user.

After iterating all users in the dataset, authentication process start for every user, by apply face detection, then embedding extraction, after that find out the claimed identity, and check distance of the user's embeddings and claimed identity in the system, if the distance bigger than adaptive threshold of the claimed identity the user tagged as not authenticated, otherwise the user embeddings will be passed to the user's "Similarity Net" to check the similarity percent, the user will authenticate if the similarity percent equal or more than 97%. To investigate the efficiency of the approach, testing modules developed in Python to register and calculate the pre-defined factors (ADT, AAT, ASP and AP), which create a good image of the approach performance, accuracy and behavior related to the state of testing dataset.

6.2.1. Testing on Local Dataset

Applying the proposed approach to the personal dataset gave the results in Table 1.

Table 1: local Dataset testing results

Factor	Value
ADT (Average Declaration Time)	11 seconds
AAT (Average Authentication Time)	14 seconds
ASP (Average Similarity Percent)	99.5%
AP (Authenticated Percent)	100%

The results of this test show the status of the proposed approach in real world. By using laptop with normal processing power as mentioned previously (using CPU Core i7, with 8G RAM) and got (11 to 14) seconds in average to declare and authenticate the users, which can be considered as a good result, the Average Similarity Percent (ASP) of this test (99.5%) indicate to the efficient of "Similarity Net", so the same user with some

changes in image capturing conditions (as in this dataset) still recognized well, the (100%) as Authentication Percent (AP) shows the effectiveness of "Similarity Net" as replacement of triplet loss.

6.2.2. Testing on Kaggle Dataset

The previous sub section (6.2.1) shows the viability of the proposed approach in real life, but in the real world the approach has to deal with big number of users with limited processing power. So, to know the behave of the proposed approach in this situation the approach was tested on Kaggle dataset, where image capturing conditions have normal changes with suitable number of images for each user, and number of users is bigger than expected comparing with processing power of testing computer.

Table 2: Kaggle dataset testing results

Factor	Value
ADT (Average Declaration Time)	12 seconds
AAT (Average Authentication Time)	9 seconds
ASP (Average Similarity Percent)	99.8%
AP (Authenticated Percent)	100%

The results in Table 2, indicate that the approach worked well, even with (391 users) the number of users didn't badly affect (ADT and AAT) factors, the increase in (ADT) factor is because of the number of users and that's normal because defining (Adaptive threshold) and training (Similarity Net) are depend on the number of users in the system, whereas the decrease of (AAT) factor happened because of the number of users and good number of user's images, where the number of user's images help to increase effectivity of Euclidian distance and Adaptive threshold, the number of users here play a major rule in these two factors, too many users with low processing power on server side will produce long time in (ADT and AAT) factors.

By checking (AP and ASP) factors, a little increase in (ASP) factor happened because of the number of users, in other words, the number of users and user's images lead to more numbers of similar users, so more data to train (Similarity Net) and by the result increasing in (ASP) factor happened. The (100%) of (AP) factor is expected because of having a good number of user's images each time we apply authentication process (about 10 images), the stability in images number lead to best results in declare and authenticate users.

6.2.3. Testing on LFW Dataset

When it comes to LFW, the testing process can be described as the hardest and toughest, because the number of user's images and capturing conditions do not simulate the case being studied in this research, the test focused on ASP and AP only, without considering other factors (AAT and ADT) because these factors increased too much because of the positive relation between number of users and processing power as mentioned previously. The test was applied on a part of LFW (because of the processing power limitation) which contained 1610 individuals;

Table 3: LFW dataset testing results

Factor	Value
ASP (Average Similarity Percent)	99.2%
AP (Authenticated Percent)	95.4%

The results in Table 3 show, decreasing in (ASP) factor, because of the huge changes in conditions of captured images and non-controllable number of images of each user. When checking the decrease of (AP) factor, the most cases of not authenticated users happened because of face detection failure and lacking of users' images, beside some users' Similarity Percent smaller than 97%, because of the big changes in facial and capturing conditions.

Our approach was compared with [14] and [13], in [14] FaceNet model used with clustering algorithm, the identity threshold defined using (Adaptive threshold), in [13] proposed FaceNet model with classification neural network, the neural network trained using triplet loss algorithm. The comparison is described in

Table 4. Notice the big improvement over [14] in the authentication percent, approximately 23%, by using similarity neural network as replacement of triplet loss in this approach. However, there is a small difference with FaceNet results (about 4%), because of the approach has prevented any mis-authentication by set high similarity percent, to this case study un-authenticate a user better than mis-authenticate any user. Therefore, increasing the

threshold of similarity percentage to 97% was done, and the decreasing in authentication percent accepted.

Table 4. comparing proposed approach to FaceNet and Adaptive threshold algorithm

Dataset	Adaptive threshold	FaceNet	Our approach
LFW Dataset	76.46%	99.63%	95.4%

to get more accurate information about the effective of "Update User Face" process, more detailed data needed. So, another test was done and got the results in Table 5.

Table 5. Details of successfully checked users

LFW total tested users	2345
Successfully authenticated users	2235
Authenticated users without update process	637
Authenticated users after first update process	1201
Authenticated users after second update process	285
Authenticated users after third update process	112

As shown in Figure 24, the "Update User Face" process is very important and increase the accuracy of this approach, especially when a users' face changes, where 54% of successfully checked users done after the first update.

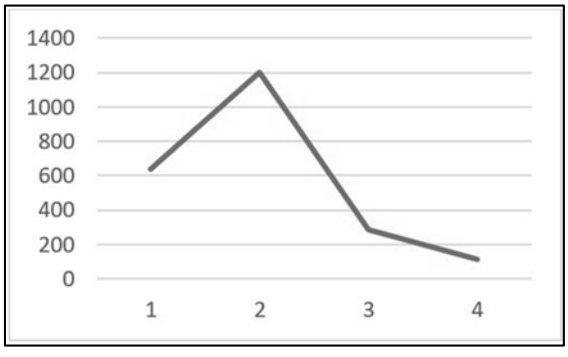

Figure 24. the effectiveness of User face update process

7. RESEARCH CONTRIBUTION AND CONCLUSION

This article presents an algorithm to secure online services and applications using face recognition. This research proposed using "Viola and Jones" for face detection on client side, and confirm the detected faces using MTCNN on the server side, beside applying FaceNet model to extract features' vectors of the user's images, and use Euclidian distance to define adaptive threshold for each user and collect embeddings to train users' Similarity Nets. Beside use mentioned models and algorithms to create new user registration process, update user process (to solve problems of facial changing and user aging), identity confirmation process with high similarity threshold to prevent mis identity confirmation. The proposed approach was tested on multiple datasets to evaluate the combined methods and processes, the experimental results proved its effectiveness. However better results possible, if a higher processing power was applied (e.g. GPUs), which could lead to a better performance (up to hundreds of times faster). the approach achieved 100% of successfully identity confirmation on viable datasets that simulate the real life applying of this approach, and 95.4% achieved when dataset has users with big changes in capturing conditions and not having enough images. So, to achieve the best results in response time and authentication process, this research recommends to use suitable processing power on the server side according to the number of users, and get a good number of user's images with best possible resolution considering internet status and server-side processing power.

REFERENCES:

[1] 1. Singh S, Prasad SVAV. Techniques and challenges of face recognition: A critical review. Procedia Comput Sci. 2018;143(2018 Jan 1):536–43.
[2] 2. Singh KR, Zaveri AM, Raghuwanshi MM. Illumination and Pose Invariant Face Recognition : A Technical Review. Int J Comput Inf Syst Ind Manag Appl. 2010;2(November 2014):29–38.
[3] 3. Zhao W, Chellappa R. Image-based face

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

recognition: Issues and methods. Opt Eng York-marcel dekker Inc. 2002;78(Jun 14):375–402.

[4] 4. Zeng B, Liu R, Liu N, Yin N. Effect of pretreatment methods on face detection in video images. J Phys Conf Ser. 2019;1176, No.(Mar):032027. IOP Publishing.

[5] 5. Zhang Y, Xu X, Liu X. Robust and High Performance Face Detector. arXiv Prepr arXiv [Internet]. 2019;1901.02350(2019 Jan 6):1–9. Available from: http://arxiv.org/abs/1901.02350

[6] 6. Li J, Wang Y, Wang C, Tai Y, Qian J, Yang J, et al. DSFD: Dual Shot Face Detector. In: InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition [Internet]. 2019. p. 5060–9. Available from: http://arxiv.org/abs/1810.10220

[7] 7. Zhang F, Fan X, Ai G, Song J, Qin Y, Wu J. Accurate Face Detection for High Performance. arXiv Prepr [Internet]. 2019;1905.01585(2019 May 5):1–9. Available from: http://arxiv.org/abs/1905.01585

[8] 8. Balaban S. Deep learning and face recognition: the state of the art. Biometric Surveill Technol Hum Act Identif XII. 2015;9457(2015 May 15):94570B.

[9] 9. Elmahmudi A, Ugail H. Deep face recognition using imperfect facial data. Futur Gener Comput Syst. 2019;99(2019 Oct):213–25.

[10] 10. Masi I, Wu Y, Hassner T, Natarajan P. Deep Face Recognition: A Survey. Proc - 31st Conf Graph Patterns Images, SIBGRAPI 2018. 2018;(2018 Oct 29):471–8.

[11] 11. Viola P, Jones M. Robust Real-Time Face Detection. Int J Comput Vis. 2004;57(2004 May):1;57(2):137-54.

[12] 12. Zhang K, Zhang Z, Li Z, Qiao Y. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. IEEE Signal Process Lett. 2016;23(10):1499–503.

[13] 13. Schroff F, Kalenichenko D, Philbin J. FaceNet: A unified embedding for face recognition and clustering. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2015;07-12-June:815–23.

[14] 14. Chou HR, Lee JH, Chan YM, Chen CS. Data-Specific Adaptive Threshold for Face Recognition and Authentication. Proc - 2nd Int Conf Multimed Inf Process Retrieval, MIPR 2019. 2019;(2019 Mar 28):153–6.

[15] 15. Sudhakar S. Histogram Equalization - Towards Data Science [Internet]. Medium. 2017 [cited 2019 Nov 25]. Available from: https://towardsdatascience.com/histogram-equalization-5d1013626e64#targetText=Histogram Equalization is a computer,intensity range of the image.&targetText=This allows for areas of,to gain a higher contrast

[16] 16. Yadav G, Maheshwari S, Agarwal A. Contrast limited adaptive histogram equalization based enhancement for real time video system. Proc 2014 Int Conf Adv Comput Commun Informatics, ICACCI 2014. 2014 Nov 26;2392–7.

[17] 17. Koch G, Zemel R, Salakhutdinov R. Siamese neural networks for one-shot image recognition. nICML Deep Learn Work. 2015;2(Jul).

[18] 18. Sandberg D. facenet/inception_resnet_v1.py at master · davidsandberg/facenet · GitHub.

[19] 19. David Sandberg. GitHub - davidsandberg/facenet: Face recognition using Tensorflow [Internet]. GitHub. 2017 [cited 2019 Nov 25]. Available from: https://github.com/davidsandberg/facenet

[20] 20. Specht DF. A General Regression Neural Network. IEEE Trans Neural Networks. 1991;2(6):568–76.

[21] 21. Kingma DP, Ba JL. Adam: A method for stochastic optimization. 3rd Int Conf Learn Represent ICLR 2015 - Conf Track Proc. 2015;1–15.

[22] 22. tehnokv. pico.js: a face-detection library in 200 lines of JavaScript [Internet]. 2018 [cited 2019 Nov 25]. Available from: https://tehnokv.com/posts/picojs-intro/

[23] 23. Gallina R. faces_data_new | Kaggle [Internet]. 2018 [cited 2019 Nov 25]. Available from: https://www.kaggle.com/gasgallo/faces-data-new/metadata

[24] 24. LFW Face Database : Main [Internet]. University of Massachusetts. 2014 [cited 2019 Nov 25]. Available from: http://vis-www.cs.umass.edu/lfw/