Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 47 (2026), No. 2, pp. 539–551
DOI : 10.47974/JIOS-1606

Hybrid CNN models for suspect facial recognition system

Ghada Abdelhady *
Department of General Systems Engineering
October University for Modern Sciences and Arts
Giza 12572
Egypt

Ahmed Mohsen †
Electrical Communication
Back Office Core Circuit Switching
IP Multimedia Subsystem and International Gateway Engineer
Huawei
Cairo 44741
Egypt

Nermeen Abdelnaser §
Electrical Communication
Value Added Service Engineer
Huawei
Cairo 44741
Egypt

Abstract

The criminal landscape is evolving rapidly. Perpetrators are constantly upping their technological game, exploiting new tools for nefarious purposes. Unfortunately, law enforcement isn't always keeping pace. While a seasoned deceiver might be adept at hiding their true colors, there are often involuntary tells – especially in facial expressions. However, pinpointing these subtle clues in a bustling environment can be a real headache for authorities. Face recognition is a vital application in computer vision with many real-world use cases, such as security systems, access control, and authentication. It is worth mentioning that the Convolution Neural Network (CNN) plays an important role in speeding up and

* *E-mail:* **gabdelmouez@msa.edu.eg (Corresponding Author)**
† *E-mail:* **ahmed.mohsen7@msa.edu.eg**
§ *E-mail:* **nermeen.abdelnasser@msa.edu.eg**

enhancing the accuracy of facial recognition algorithms to assist in surveillance systems and access control.

This project is designed to empower various authorities, including government agencies, international airports, and Egyptian border security, to enhance their investigative capabilities. by using more efficient models for face detection and recognition to build a hybrid algorithm of two CNN models to detect and predict suspicious persons and report to the authorities. Beyond facial recognition capabilities, the proposed model incorporates face detection functionalities. This will enable the identification of individuals flagged for travel restrictions or suspected of attempting unauthorized departures.

This paper presents a survey of popular face recognition models using CNN and explains their features. In addition to the survey, it integrates two algorithms for detection and recognition based on a detailed analysis applied to each presented model, explaining their accuracy and the most crucial factor, processing time. The paper also introduces a practical use case that can be established in any crowded region.

Subject Classification: *(2010) 68T45 (Image processing and computer vision), 94A08 (Security applications)*

Keywords: *CNN (Convolutional Neural Network), Criminal investigation tracker, FaceNet, Face recognition, Haar cascade classifier, MTCNN (Multi-task Cascaded Convolutional Networks), Surveillance.*

1. Introduction

In recent years, deep learning models have achieved state-of-the-art performance in face recognition tasks. Convolutional Neural Networks (CNNs) have been widely used and there are several popular models that have achieved excellent results on many face recognition benchmarks. Here are some examples such as Google FaceNet, Haar Cascade, MTCNN algorithms.

Recent advancements in deep learning have significantly propelled facial recognition technology. Dubey and Jain [1] successfully employed a VGG16 transfer learning model for automatic facial recognition, showcasing its potential across various applications such as security, medical science, and to know the behavior of a person or community. Their work highlighted the accuracy and robustness achievable through advanced neural network architectures, paving the way for more sophisticated biometric authentication and surveillance systems.[1] Our research builds upon these advancements, focusing on suspect facial recognition systems. We aim to leverage similar cutting-edge techniques to improve the accuracy and efficiency of identification processes in law enforcement and security settings.

Venkatesan et al. [2] investigated the implementation of facial recognition alongside Triple DES encryption to enhance online transaction security. Their work addressed the critical issue of proxy detection, ensuring secure and reliable transactions. This approach, combining advanced encryption with biometric authentication, represents a significant step forward in mitigating fraud and safeguarding the integrity of digital payment systems. [2] Our research aims to contribute further by exploring novel hybrid CNN models specifically designed for suspect identification in crowded regions like airports to secure them from any expected attack. Our goal is to bridge the gap between security and usability in different applications.

In general, facial recognition for criminal identification is a security system that takes images of suspicious persons and pre-processes them to remove unwanted elements such as noise. The system then classifies the images based on landmarks such as the distance between the eyes and the length of the jawline. It then searches the database for an exact match and displays the output. This system focuses on implementing criminal identification systems, which can be difficult due to the use of latent fingerprints that may not be obtained from the crime scene. Criminals are getting smarter and are usually very careful when leaving fingerprints at crime scenes. The system includes a face database and image processing algorithms to match the face feed to the faces stored in the database. [3] [4]

To achieve the desired results of such systems, it should be designed based on two processes: detection and recognition. Face detection is one of the most important steps in a facial recognition system and has mainly four phases: knowledge-based, feature-invariant, template-matching, and appearance-based [4]. Regarding face recognition, it requires three processes: training, testing, and evaluation. In the training and testing processes, the algorithm takes samples of the images to be learned and tested and then determines a distinct model for each image. In the evaluation process, a model of a newly acquired test image is compared against all existing models in the database, which is supposedly obtained from the authorities, and the nearest corresponding model is acquired to determine whether the recognition is triggered [5]. This initial stage employs Principal Component Analysis (PCA) on a comprehensive dataset of facial images. PCA extracts a set of key features, known as eigenfaces, that act as building blocks for human facial representations. Essentially, any human face can be reconstructed as a unique combination of these standard eigenfaces.

In the following sections, we explore several popular algorithms and integrate them in pairs to achieve maximum speed. All the presented hybrid models use FaceNet for detection, which has proven to have a high performance in face detection using capturing images in real time. The next section focuses on FaceNet and then the sections go in-depth through the other models and the models integration, shedding light on the evaluation of each model, their strengths, and weaknesses.

2. **Google Facenet Model**

2.1 *Overview*

As mentioned earlier, face recognition is a challenging problem in computer vision, as it involves detecting and identifying human faces in images or videos. Deep learning models, such as FaceNet, have achieved significant progress in face recognition for real-time applications. FaceNet is a deep neural network that can extract a high-dimensional embedding of a face, which can be used for face verification and identification. [6]

The technique teaches mapping from input images of faces to a high-dimensional feature space, where each face is represented as a compact embedding using a deep convolutional neural network (CNN) architecture. FaceNet is widely regarded as one of the most accurate face recognition algorithms available, having achieved cutting-edge performance on a variety of facial recognition benchmarks. On numerous benchmark datasets, the FaceNet method has been demonstrated to outperform other popular facial recognition algorithms, such as Eigenfaces, Fisherfaces, and local binary patterns (LBP).[7]

One of the FaceNet algorithm's primary features is its capacity to build highly discriminative embeddings capable of successfully distinguishing between distinct people, even under difficult settings such as changes in position, lighting, and facial expression. This is accomplished by employing a deep convolutional neural network architecture with a triplet loss function. Another benefit of the FaceNet algorithm is its ability to do real-time facial recognition, making it suitable for applications such as security systems and mobile devices. Several open-source FaceNet implementations are available in popular deep-learning libraries like Keras and Tensorflow, making it simple for developers to utilize and incorporate into their projects.[8]

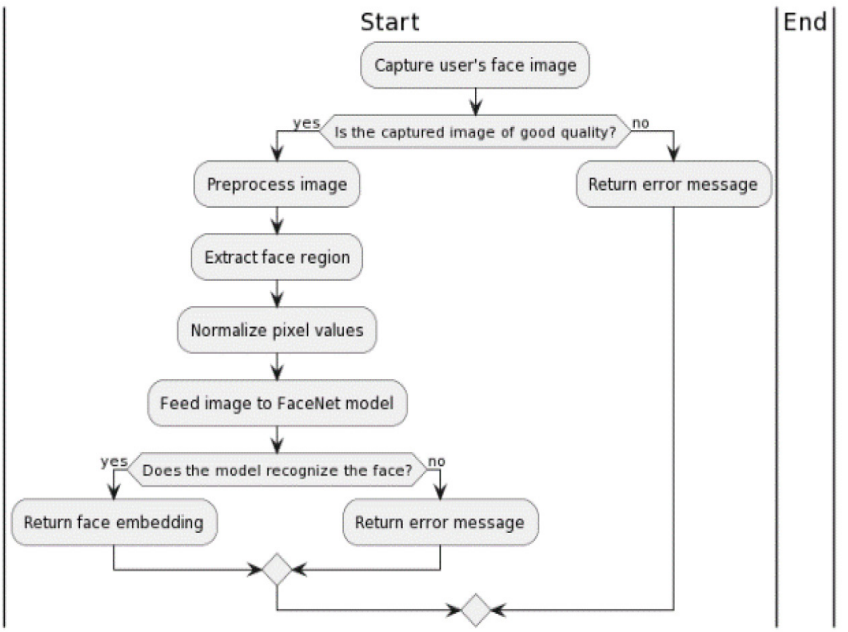

Figure 1

FaceNet Activity Diagram

2.2 *FaceNet Activity Diagram*

Overall, the activity diagram for the Facenet model shows that the face recognition process involves several steps, from input image preprocessing to feature extraction, embedding generation, and face matching. Figure 1 shows the activity diagram for the Facenet model [9]. It begins with the input of an image containing a face. The image is preprocessed to normalize the size and color of the face, and to align it in a standardized way. Then the Feature extraction stage comes to extract a set of high-dimensional features from the preprocessed face image. This is done using a deep convolutional neural network. The feature vectors obtained from the convolutional layers are used to generate an embedding vector that represents the face in a high-dimensional feature space. The embedding vectors of the input face image and the faces in the database are compared using a similarity metric such as cosine similarity. If the similarity score exceeds a certain threshold, the input face is recognized as a match. The recognition process ends with the output of the recognized identity or an indication that the input face is not recognized.

3. **CNN Models for Face Recognition Algorithm**

There are several models of face recognition algorithms using CNN, each with its own approach and technique for identifying and verifying faces. Here are some popular models:

3.1 *Face_recognition Library*

The face_recognition library is a Python library that provides a simple interface for conducting face detection and recognition operations. It does these tasks using pre-trained models from the dlib and OpenCV libraries, and it supports both standard object detection algorithms like HOG and deep learning models. Face embeddings are generated by the library using a deep learning technique and can be used to identify persons. [3], [10]

One of its primary advantages is its simplicity and ease of use, which allows it to be used by developers who are unfamiliar with the specifics of facial recognition algorithms. It is also functional with a broad variety of Python-based applications, making it simple to include into current projects.

3.2 *MTCNN*

MTCNN (Multi-Task Cascaded Convolutional Networks) is a prominent face detection approach based on deep learning that is widely utilized in computer vision applications. MTCNN is intended to detect faces in images with high accuracy and efficiency, especially under difficult settings such as changes in lighting, position, and facial expression.

One of MTCNN's primary features is its ability to detect faces at various scales and properly localize face areas. This is accomplished by employing a multi-scale pyramid picture representation and a cascade architecture that gradually refines the bounding boxes. [11]

MTCNN has demonstrated cutting-edge performance on various face detection benchmarks, including the WIDER FACE and FDDB datasets. It's widely utilized in computer vision applications like facial recognition, emotion recognition, and video surveillance. MTCNN is included in a number of deep learning libraries, including TensorFlow and PyTorch, making it simple to integrate into current applications" [5], [12], [13].

3.3 *Haar Cascade*

Haar Cascade is a well-known machine learning-based face detection technique used in computer vision applications. It is built on the idea of

Haar-like features, which are simple rectangular features that capture local image attributes like edges and contrasts. The Haar Cascade approach uses a huge dataset of positive and negative pictures to train a classifier. The positive images contain examples of the thing to be detected (for example, faces), but the negative images do not. By collecting Haar-like features from the photos and using them to train a cascade of classifiers, the classifier learns to distinguish between positive and negative images. The speed and efficiency of the Haar Cascade algorithm are two of its main advantages. It can detect faces in real-time. [4]

4. **Hybrid Integrated CNN Models**

In this paper, we are going to introduce three hybrid deep learning models integrating between FaceNet algorithm and other algorithms. These models will be performing face recognition and detection. The majority of them will make use of Python libraries like OpenCV, PIL (Pillow), NumPy, Matplotlib, and Keras.

4.1 *Facenet and Haar Cascade hybrid*

This section details the system's approach to face recognition and detection. It leverages a hybrid architecture that combines the strengths of the FaceNet model and the Haar cascade classifier. The Haar cascade classifier, a machine learning-based object detection algorithm used to detect faces in images, is the first section of the system.

Upon receiving the image folder path, the code iterates through each image within the folder. It leverages OpenCV to read the image data. Subsequently, the Haar cascade classifier is employed to detect any present faces. For each identified face, the corresponding region within the image is extracted. These extracted face regions are then uniformly resized to 160x160 pixels and converted into an array format suitable for processing by the FaceNet model. Finally, the FaceNet model processes each array representation, generating a unique "face embedding" for each detected face. (also known as a face signature or face descriptor). The key is the image filename (without the extension), and the face embedding is stored in a dictionary called a database.

The js_to_image() function is then defined, which uses OpenCV to convert the image data received as a JavaScript reply (in base64 format) to a NumPy array. To facilitate face identification within the system, a dedicated function named findFaces is implemented. This function accepts image data as input and utilizes the Haar cascade classifier for face

detection. Subsequently, it compares the extracted face embeddings of the detected faces against known face embeddings stored in the database for identification purposes. Leveraging OpenCV's functionalities, the identified faces within the image are visually marked with their corresponding identities. This annotated image is then saved as "photo.jpg" for further analysis or record-keeping. Additionally, the code incorporates functionalities written in JavaScript for capturing images directly from a webcam. These captured images are displayed within the notebook environment and can also be saved to the local storage device. These are the functions take_photo() and js_reply_to_image().

The time required to recognize a human face has decreased significantly as facial recognition technology has advanced. Modern facial recognition systems, when combined with techniques such as FaceNet and Haar cascade classifiers, can identify human faces with remarkable accuracy in as little as 80 seconds on average. The high accuracy of these systems, which can reach 95% or higher, ensures that the possibility of false positives or false negatives is minimized, making them reliable and trustworthy for critical tasks.

4.2 *MTCNN and FaceNet hybrid*

Face detection and identification tasks are performed using the FaceNet model and the MTCNN face detector. It loads a folder of images, detects faces in each image, then extracts facial embeddings using the FaceNet model. The preprocessing and database creation: the system begins by defining a folder to store images for facial recognition. It then leverages MTCNN, a leading face detection library, to identify faces within each image in the designated folder. Once faces are detected, they are meticulously cropped and resized to a standardized format of 160x160 pixels to ensure consistency. Finally, the FaceNet model is employed to convert these cropped faces into unique mathematical representations known as "embeddings." These embeddings are compiled and stored within a centralized database dictionary for future reference.

Face Recognition Process can be summarized in few steps. A crucial function named "findFaces" takes an image as input and utilizes MTCNN for face detection. The detected faces are then compared against the pre-existing face embeddings stored in the database for identification purposes. This comparison allows the system to recognize the individuals present in the image. Following recognition, the system employs labeling techniques within OpenCV to visually mark the identified faces within

the original image, assigning each face its corresponding identity. This annotated image is subsequently saved as "photo.jpg" for further analysis or record-keeping. We reach out to the Webcam Integration. In this stage, the model offers an additional function programmed in JavaScript. This functionality allows users to capture images directly from their webcam. Captured images are displayed within the user's working environment (likely a Jupyter Notebook). Users also have the option to save these captured images to their local storage device.

Lastly, regarding the performance of the system, notably, this model operates at a speed of 32 milliseconds per step, demonstrating its efficiency in processing images. **FaceNet and face_recognition library hybrid**

The model defines a function that uses OpenCV to convert a base64-encoded image to a NumPy array, as well as another function that detects and recognizes faces in images. The face_recognition library is used for face detection, while the FaceNet model is used for face recognition.

The model also includes a function for capturing an image with the user's webcam and displaying it with recognized faces and names. The taken image is processed using the findFaces() function, which discovers and recognizes faces in the image and provides the processed image's filename.

It makes use of a database of known faces saved in a pickle file that is loaded at the start of the model. Then the code runs through all of the photographs in the supplied folder, detecting and extracting faces from each image and calculating their embeddings with the FaceNet model. The embeddings are then saved to a pickle file in a dictionary with the filename as the key and the embedding as the value.

Overall, this model can be used for facial recognition tasks like authenticating a person's identity in a picture or video. The speed of this model is 30ms/step

5. Practical Experiment for Suspect Facial Recognition System

In this section, we focus on the software analysis of the selected hybrid system in the light of the results of the previous section. Based on the previous results, hybrid of the FaceNet and Face_Recognition Library can be used for facial recognition tasks like authenticating a person's identity in a picture or video. The speed of this model is 30ms/step. The UML Activity and Use Case diagrams will be analyzed in the following sections.

5.1 *UML Use Case Diagram*

A use case diagram in Figure 2 is describe how the system users will perform the required tasks. It summarizes the behaviour of the system. The accompanying use case diagram depicts the system's interaction with three primary actors: users, developers, and administrators. Seven distinct use cases are illustrated, encompassing the initial user request for access, followed by the decision-making process regarding access approval or denial. This decision can be made by either a developer or by the implemented algorithm itself, which is the hybrid model presented in this research. This hybrid model leverages the Face_Recognition Library for face detection and FaceNet for facial recognition functionalities.

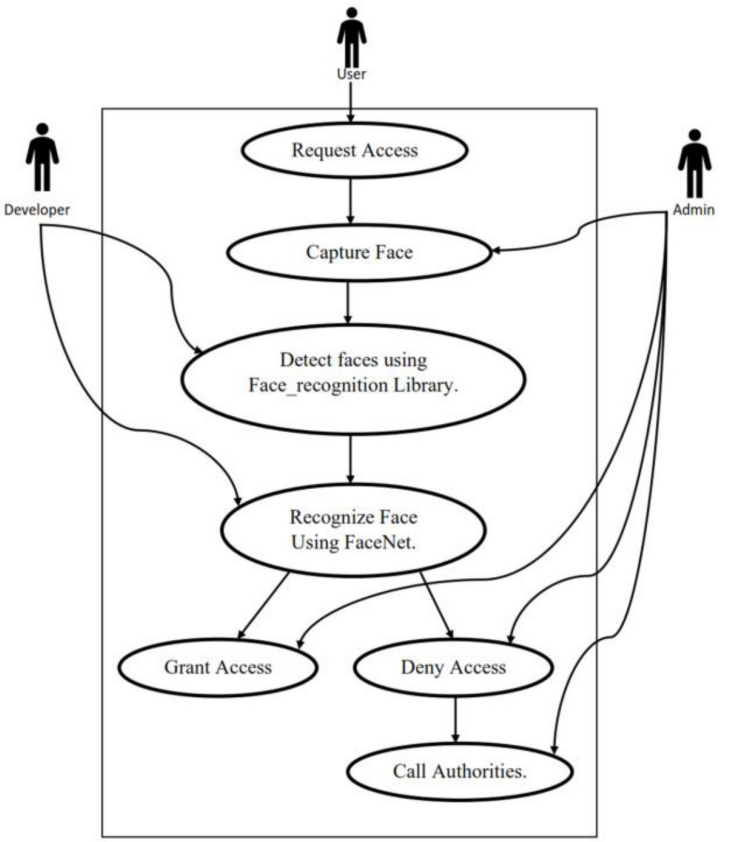

Figure 2

UML Use Case Diagram for the proposed system

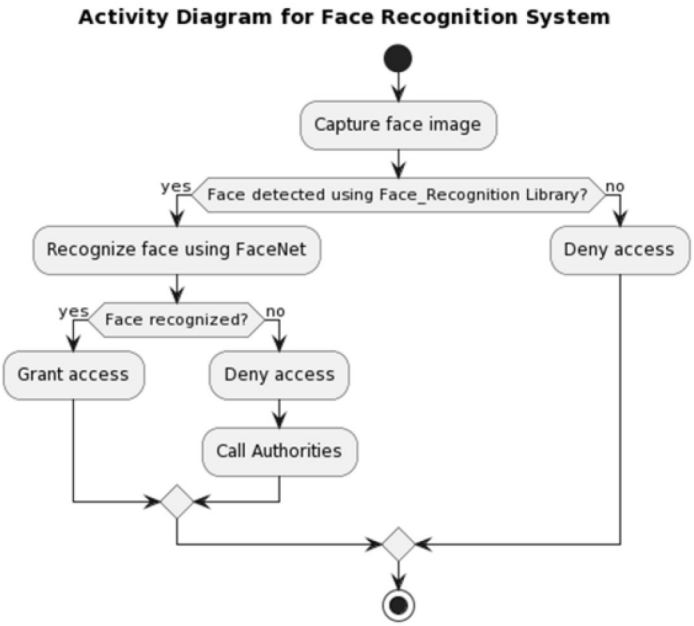

Figure 3

UML Activity Diagram

5.2 *UML Activity Diagram*

Figure 3 visually explains the flow of the system steps. The shown activity diagram describes the steps in the use case diagram in Figure 2.

6. **Conclusion**

The field of Artificial Intelligence (AI) is poised to revolutionize numerous sectors in the near future, with its impact expected to be far-reaching. Recognizing this potential, our research focuses on the deployment of AI and computer vision technologies to assist Egyptian authorities in safeguarding the security and safety of the Arab Republic of Egypt. In this paper, a proposed system for a criminal investigation tracker was presented, beginning with a discussion of the problem definition and popular models used for face recognition. We described each model individually and then tried different combinations to integrate the models in pairs. Following a review of these models, the proposed system combined the face_recognition Library, which detects the face, with the faceNet model to recognize the suspect person. the results obtained by

this combination is efficient and achieved a speed of 30ms/step. As detailed throughout this paper, the successful implementation of this project hinges on the utilization of meticulously designed software algorithms..

References

[1] A. K. Dubey and V. Jain, "Automatic facial recognition using VGG16 based transfer learning model," *Journal of Information and Optimization Sciences*, vol. 41, no. 7, pp. 1589–1596, Oct. (2020), doi: 10.1080/02522667.2020.1809126.

[2] R. Venkatesan, S. R. Raghunathan, K. S. Rajasekaran, and M. A. R. K. Marimuthu, "Secure online payment through facial recognition and proxy detection with the help of TripleDES encryption," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 24, no. 8, pp. 2195–2205, Nov. (2021). doi: 10.1080/09720529.2021.2011096.

[3] S. A. Hussain and A. Salim Abdallah Al Balushi, "A real time face emotion classification and recognition using deep learning model," *J Phys Conf Ser*, vol. 1432, no. 1, p. 012087, Jan. (2020), doi: 10.1088/1742-6596/1432/1/012087.

[4] D. Zh Satybaldina, N. S. Glazyrina, K. A. Kalymova, and V. S. Stepanov, "Development of an algorithm for abnormal human behavior detection in intelligent video surveillance system," *IOP Conf Ser Mater Sci Eng*, vol. 1069, no. 1, p. 012046, Mar. (2021), doi: 10.1088/1757-899X/1069/1/012046.

[5] S. Shinde, "Face Detection And Recognition Using Python And OpenCV," *Xi'an Dianzi Keji Daxue Xuebao/Journal of Xidian University*, vol. 15, pp. 64–72, Jul. (2021), doi: 10.37896/jxu15.7/008.

[6] Md. F. Ali, M. Khatun, and N. Turzo, "Facial Expression Recognition Using Neural Network," (2021).

[7] H. Wang, S. Wei, and B. Fang, "Facial expression recognition using iterative fusion of MO-HOG and deep features," *J Supercomput*, vol. 76, no. 5, pp. 3211–3221, May (2020), doi: 10.1007/s11227-018-2554-8.

[8] D. Meena and R. Sharan, "An approach to face detection and recognition," in *2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, IEEE, Dec. (2016), pp. 1–6. doi: 10.1109/ICRAIE.2016.7939462.

[9] A. M. Jagtap, V. Kangale, K. Unune, and P. Gosavi, "A Study of LBPH, Eigenface, Fisherface and Haar-like features for Face recognition using OpenCV," in *2019 International Conference on Intelligent Sustainable Systems (ICISS)*, IEEE, pp. 219–224, Feb. (2019). doi: 10.1109/ISS1.2019.8907965.

[10] K. M. Sajjad, "Pongsapas Watana Automatic License Plate Recognition using Python and OpenCV 03-Nov-20 1 Automatic License Plate Recognition using Python and OpenCV," (2020).

[11] Guan-Feng He, Sun-Kyung Kang, Won-Chang Song, and Sung-Tae Jung, "Real-time gesture recognition using 3D depth camera," in *2011 IEEE 2nd International Conference on Software Engineering and Service Science*, IEEE, pp. 187–190, Jul. (2011). doi: 10.1109/ICSESS.2011.5982286.

[12] S. P. Nair, K. Abhinav Reddy, P. K. Alluri, and S. Lalitha, "Face recognition and tracking for security surveillance," *Journal of Intelligent & Fuzzy Systems*, vol. 41, no. 5, pp. 5337–5345, Nov. (2021), doi: 10.3233/JIFS-189856.

[13] Ya Wang, Tianlong Bao, Chunhui Ding, and Ming Zhu, "Face recognition in real-world surveillance videos with deep learning method," in *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, IEEE, pp. 239–243, Jun. (2017). doi: 10.1109/ICIVC.2017.7984553.

Received October, 2023