IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 42, No. 5, October, 2025, pp. 2443-2452
Journal homepage: http://iieta.org/journals/ts

Face Detection and Recognition System Using Deep Learning Model in Biometric

Zaki Sekhri[1], Abdenour Mekhmoukh[1*], Mourad Zribi[2]

[1] Université de Bejaia, Faculté de Technologie, Laboratoire de Technologie Industrielle et de l'Information (LTII), Bejaia 06000, Algeria
[2] Unité de Recherche 4491, Laboratoire d'Informatique Signal et Image de la Côte d'Opale, Université du Littoral Côte d'Opale, Calais Cedex 62228, France

Corresponding Author Email: abdenour.mekhmoukh@univ-bejaia.dz

https://doi.org/10.18280/ts.420501

Received: 22 January 2025
Revised: 27 March 2025
Accepted: 6 August 2025
Available online: 31 October 2025

Keywords:
biometrics, face detection, Viola-Jones algorithm, images processing, deep learning, Face96

ABSTRACT

Biometrics is the technology used to verify identity and/or identify people based on their individual characteristics, whether physical or behavioral. Given its importance, this field has become an area of research in its own right. Today, authentication of individuals is widely used in all fields requiring controlled or secure access, such as banking applications, access to highly secure locations like government headquarters, connection to a computer or computer network, e-commerce, ... etc. Detecting faces is a fundamental step in any facial recognition system. The system can more successfully identify and recognize people when facial detection is done accurately. In this study, we employ the Haar Cascades algorithm to detect faces, which are then stored in a database. Our primary goal is to enhance the precision of face recognition by leveraging a deep-learning method, using neural networks with convolutions (CNNs). The proposed method is broken down into three phases. The first phase is the face detection from still images in the Face96 database using Viola-Jones algorithm. Followed by the second phase, which is the processing of the detected faces to make them suitable for the third phase, which is the recognition of faces detected by the concept checks based on classification whether two facial images are identical or not. According to experimental findings, our suggested solution performs better than facial recognition techniques.

1. INTRODUCTION

Systems for biometric detection and recognition, particularly those based on facial recognition, are extensively utilized in a variety of industries because to its capacity for authentication identities with high precision and ease of use. These systems leverage specific physical or behavioral characteristics unique to individuals for identification purposes. Various developments are devoted to facial recognition systems for various potential applications in fields requiring controlled or secure access such as banking applications, access to highly secure locations such as government headquarters, connection to a computer or computer network, e-commerce, ... etc. Facial recognition technology is generally used for the following two tasks: confirming and recognizing faces. The face identification system compares the person displayed with every individual in the database to offer a list of matched individuals, whereas the face verification system merely determines whether two photographs of faces are identical or not. The face detection system [1, 2] generally comprises three important phases:

- The function of the process is to detect faces in the image input and discover all the faces in the image.
- Image processing contains faces detected from the Face96 database.
- Face recognition is the process of identifying a person by comparing their face to a database of recognized faces.

Several methods have been developed for face detection. Among these methods are Principal Component Analysis (PCA) [3], Eigen Faces (EF) [4], and Local Binary Pattern (LBP) [5]. Most of these methods are tested on ideal environments and on databases where face images are well aligned. However, these assumptions are not valid in wild environments. Research on face recognition is quite active and presents a substantial challenge in the study of multidimensional visual models. The difficulty of identifying a human face is especially high because to its varying characteristics, such as age, expressions, hairstyle changes, etc. [6-9], so to have a good face detection and recognition system, in wild settings, we must overcome a number of obstacles, including changes in positions, lighting, and facial emotions, background, angle, and distance from the camera. Resolving all of these issues can improve facial recognition systems' precision and robustness.

Despite the advances in facial recognition technology, these systems continue to exhibit limitations that hinder their effectiveness in real-world scenarios. A significant challenge confronting these systems is their sensitivity to variations in light, which can adversely affect image quality and compromise the extraction of pertinent features. Conventional

methods, such as PCA and LBP, are predicated on fixed features, rendering them poorly adaptive to changes in facial appearance due to facial expressions, viewing angles, or partial occlusion by objects such as glasses or masks. Even approaches based on deep learning, while demonstrating effectiveness, necessitate substantial amounts of annotated data and considerable computing power to achieve optimal performance. Consequently, there is an imperative to engineer more robust and adaptable models that can effectively navigate the multifaceted constraints present in the real world while maintaining a high degree of accuracy.

In our work, the first phase is the face detection from still images in the Face96 database using Viola-Jones algorithm. Before that, we use the CNN who presented a model of deep-learning for features extraction in detected faces, after that, fully connected layers are used for classification in order to identify the identified face. This architecture is similar to the well-known Face Net model, developed by Google, which has set the benchmark in facial recognition tasks. This model has enabled us to expect high accuracy and robustness against the challenges of capturing images from the Face96 database that represent variations in facial expressions and head orientations and changes in illumination and pose, with the intention of enhancing the detection and facial recognition systems' precision and resilience. The structure of the paper is as follows: The related works are introduced in the following section. The face detection and localization system are explained in Section 3. Section 4 present our CNN architecture used to extract features and classify faces. The results are presented in the last Section.

2. RELATED WORK

Zamir et al. [10] developed a real-time surveillance system that couples a convolutional neural network (CNN) with a Raspberry Pi to perform face recognition efficiently. To extract facial features and landmarks for accurate identification, the network was trained on a labeled dataset; query images were then matched against this dataset and a voting scheme further boosted recognition accuracy. Evaluated on three benchmarks, the system achieved 98.24 % on a generic face-recognition set, 89.39 % on a 14-celebrity set, and 95.71 % on an additional recognition benchmark. The authors also assessed its performance under challenging conditions such as subjects wearing masks or sunglasses and in live-video streams.

Zhang et al. [11] introduced a Multi-task Cascaded Convolutional Network (MTCNN) that performs face detection and alignment jointly. By exploiting the intrinsic correlation between the two tasks, the framework improved both precision and efficiency. The network consisted of three CNN stages that progressively detected faces and facial landmarks, yielding markedly better results than prior methods, especially for large pose variations and partial occlusions.

Pai et al. [12] surveyed CNN-based face-recognition algorithms, detailing their architectures, loss functions, and accuracy gains over traditional approaches. Key milestones such as DeepFace, DeepID, and FaceNet were discussed, together with lightweight CNNs tailored for mobile and embedded devices that maintained high accuracy while reducing computational cost.

Liu et al. [13] proposed lightweight CNN enhancements for face recognition, incorporating structures such as the Squeeze-and-Excitation (SE) block and novel training strategies. The resulting models achieved high precision and low resource consumption, making them suitable for deployment on mobile and embedded platforms without sacrificing recognition performance.

3. HAAR-CASCADE CLASSIFIERS VIOLA JONES

Image processing and computer vision employ object detection, a technology focused on identifying instances of objects within images for example, cars, trees, buildings, or human faces. A core application is face detection, which seeks to determine whether faces are present in a particular image or not. Using image processing techniques, object detection is a way to determine whether an object of a particular type exists [10]. Things are classifiable according to their shape, color, and texture. Although color-coding is a useful technique for item identification, it has limitations because illumination is a key factor in object detection. To get around the previous method, object identification using attributes, shape, etc., has been used. Viola Jones algorithm was chosen for first facial recognition because of its high rate of detection as the first reason; secondly, this algorithm enables facial detection and recognition systems to operate in real time. This detector can handle a 45° rotation of the face around the vertical and horizontal axis and is often most effective on frontal faces [11]. Its four primary foundations, which enable real-time operation, are the learning classifier with Ada-Boost and cascade structure, the Integral Image, and Haar Feature Selection.

Figure 1. Detection of one or more faces via Viola-Jones algorithm

Figure 2. Problem with face pose variation (tilted and rotated)

Figure 3. Result of the detection of partially hidden faces

The Viola-Jones algorithm [14] is known for its efficient performance in face detection (Figure 1), but it has limitations when detecting non-frontal faces (Figure 2 and 3). This limitation arises from its Haar-like feature-based approach, which is optimized for frontal face detection. In our study, we have considered these constraints and compared the results with deep learning models capable of handling different facial orientations. Several works, such as that of Lienhart and Maydt, have attempted to enhance the algorithm by introducing rotated features, but they do not achieve the robustness of modern deep learning models.

3.1 The integral image

Integral images are created by economically generating the sum of the pixel intensities within a specific image area. It is employed to compute Haar-type characteristics quickly. Calculating the total area of a rectangle within the original picture is quite effective; only four additions are required for every any size rectangle. An approach for economically producing the total of pixel intensities inside a given image rectangle is called an integral image. This is employed to compute Haar-type characteristics quickly. With just four additions needed for every arbitrary rectangle size, calculating the total area region within the original image is incredibly efficient. Figure 4 illustrates the calculation procedure.

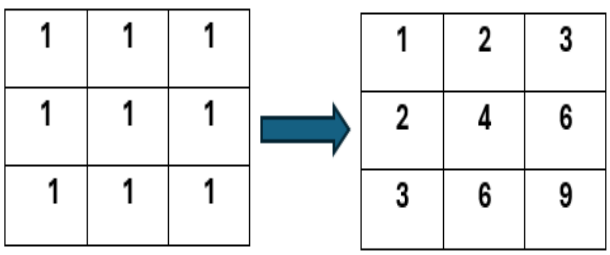

Figure 4. Integral image

3.2 Haar-like features

Object recognition uses Haar-like features, digital image features that leverage common facial characteristics. For instance, the area around the eyes is usually dark compared to the surrounding pixels, while the nose region is often brighter. Comparing the sums of pixel values within these regions, a darker region will have a lower sum than a lighter region helps identify these contrasts. This difference in pixel intensity can indicate features like eyebrows or the nose's reflective area. Viola and Jones' research identified four types of Haar-like features for facial feature detection: edge, line, center-surround, and diagonal features. These features effectively detect edges, lines, and diagonal patterns, respectively, facilitating facial component identification, according to the Figure 5.

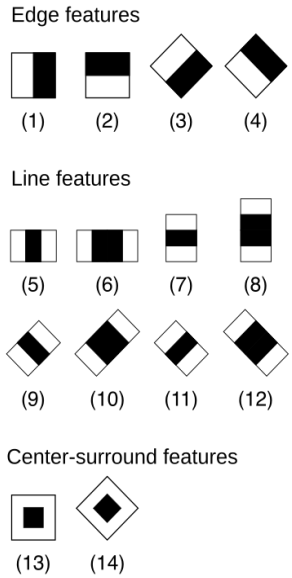

Figure 5. Haar features in Viola-Jones

The height and width of the aforementioned Haar characteristics might vary. The sum of the black and white pixels in the Haar feature applied to the face are computed, and a single value is obtained by subtracting them. The cheek, nose, eyes, and other facial features are indicated if the value is higher there.

The calculation of Haar features yields an image with around 160000+ features overall. t is inefficient to add up every pixel in an image and then deduct them in applications to get a single value that happen in real time. To solve this problem, Ada boost classifier is used.

3.3 AdaBoost classifier

The AdaBoost classifier plays a crucial role in distinguishing between relevant and irrelevant features by giving each attribute a weight according to its significance, it makes sure that only the most important traits are given priority. Haar-like features are used in the Viola-Jones object detection framework., each representing a weak classifier within an AdaBoost framework. AdaBoost assesses the performance of numerous classifiers across various sub-regions of training images. Sub-regions triggering strong classifier responses are labeled as positive (likely containing a face), while those yielding weak responses are labeled negative. High-performing classifiers receive greater weight, resulting in a strong, or boosted, classifier composed of the most effective weak classifiers. This AdaBoost training

process, using training data to learn feature importance, effectively establishes a threshold for determining which features are significant for face detection [15].

3.4 Cascade classifiers

After processing, the Ada Boost algorithm identifies approximately 2500 optimal features. A 24×24-pixel window is then applied to the input image to assess whether any region contains a face, calculating these features for each region remains a tedious process. The role of the cascade is to quickly rule out regions that do not contain a face and thus avoiding time-consuming calculations. Thus, it reaches the speed needed for face detection in real time. The face identification procedure is broken down into multiple steps by the cascade system we have put up. In the first step, we have a classifier built using our finest qualities. Stated differently, in the initial phase, The best features are those that the sub-region passes through, such the nasal bridge or eye identification features. All of the remaining features are accessible in later stages. The first phase evaluates an image sub-region as it enters the cascade. The result of this phase is "maybe" if it considers the sub-region to be positive, that is, a face. A sub-region is moved to the waterfall's next stage and the procedure resumes when it receives a "maybe" rating. The image is finally identified as a human face and shown to the user as a detection if all of the classifiers agree with it. Since the image lacks a human face, it is actually rejected right away if the first step provides a negative rating. If the image passes the first stage but fails the second, it is also rejected. In fact, the image can be rejected at any stage of the classifier. Here is what is going to help us increase our image processing speed [15].

4. DESCRIPTION OF PROPOSED METHODOLOGY

The process of the proposed methodology is based on the use of deep learning approach where the Viola-Jones algorithm is employed to identify faces in input images and the CNN model is used for extract features of the faces detected (see Figure 6). In our work, we use the standard Face96 database.

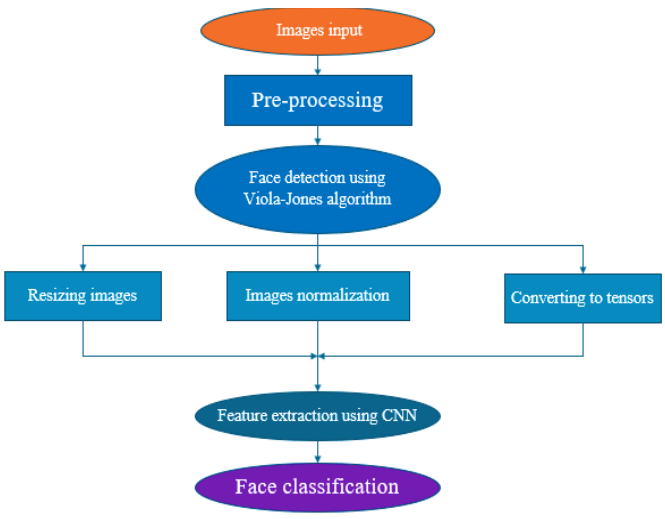

Figure 6. Flow-Chart of the proposed methodology

The collection of data includes about 3040 color images (196 × 196) pixel and a face's frontal view of 152 different persons with 20 images per person. As seen in Figure 7, the test set includes a wide range of lighting, backgrounds, and face sizes that mimic actual world circumstances.

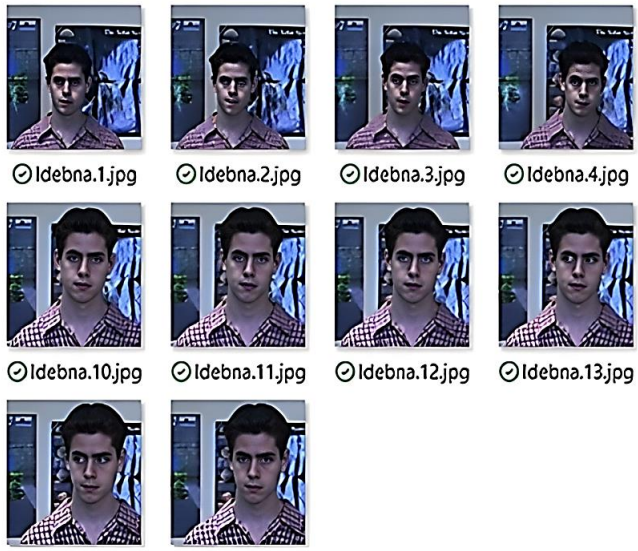

Figure 7. Original images from the Face96 database

4.1 Face detection

After the set of pre-processing steps, to find the face in the picture, Viola-Jones is used. The Viola-Jones detector was chosen as the detection algorithm because of its high detection rate and ability to operate in real time. The faces detected from the input images by the Haar Cascades algorithm are saved into a database us shown in Figure 8.

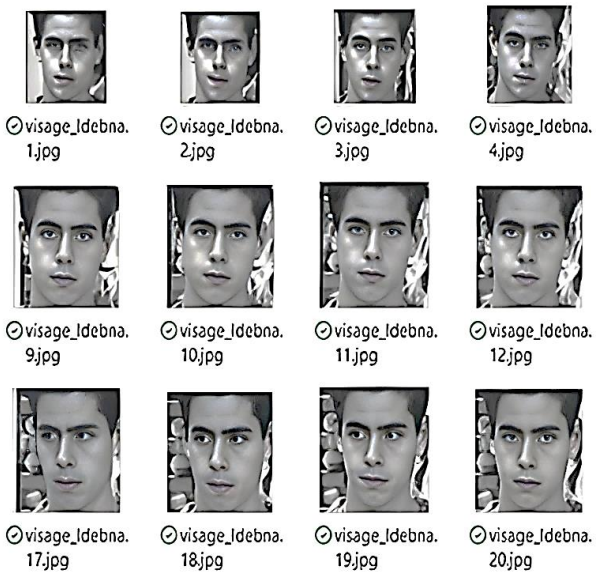

Figure 8. Face detected by Viola-Jones algorithm

4.2 Image pre-processing

The pre-processing images step in computer vision data processing is crucial to prepare the information suitably for training the model. In our work, transformations and pre-processing applied to images include resizing, tensor conversion, and normalization.

- **Resizing**: Input images can have various dimensions. To

ensure that all images have a uniform size, which is necessary for the data batches and convolutional layers of our neural network, they are resized to a fixed size. This ensures that all images will have the same dimension before being passed into our model. In the pre-processing phase, we set all the face detected to a fixed resolution of 100 × 100 pixel. Ensuring that all images of faces detected are the same size, so that the CNN model can process the data consistently. This may involve resizing images to a specific resolution.

- **Normalizing pixel values:** Normalization helps to stabilize and accelerate neural network training. It guarantees that pixel values have a consistent scale and are centered on zero. This improves model convergence. In our case, with a mean of 0.5 and a standard deviation of 0.5, values of pixels in the [0, 1] range are transformed to be [-1, 1].
- **Converting to tensor:** Our neural network in PyTorch requires tensors (torch. Tensor objects) as input. Normalizes pixel values from range [0, 255] to range [0, 1] by dividing by 255. Rearranges image dimensions from the format (H, W, C) to the format expected by PyTorch (C, H, W) (for example, 3 for RGB).

4.3 Feature extraction

The CNN architecture is used for feature extraction from detected faces in the Face96 database. In this stage, a CNN model is used to extract features from faces that have already been detected and pre-processed. The CNN used to extract facial features is composed of four convolutional layers; each followed by an activation function (ReLU) and batch normalization, as well as a pooling layer (Max Pooling) to reduce the dimensionality of the features. The first convolutional layer (Conv1) applies the input image's convolutional filters to extract low-level features such as edges and textures. The results of the convolution are then passed through a ReLU activation function to introduce non-linearity, followed by batch normalization to stabilize and accelerate the network's learning. Finally, a pooling operation (Max Pooling) is performed to reduce the spatial dimensionality of the extracted features.

The second convolutional layer (Conv2) follows a similar process to Conv1, extracting mid-level features from the low-level features produced by the first layer. The third convolutional layer (Conv3) continues the same process, extracting high-level features from the mid-level features produced by the second layer. Finally, the fourth convolutional layer (Conv4) extracts high-level features, which are considered more abstract representations of the faces in the image. This CNN is therefore designed to obtain vector representations of features progressively extracted. These features become increasingly abstract and discriminative of faces as we move from one layer to the next. The vector representations of the extracted features will then be used as input for the fully connected layers of the FaceNet model to perform the final classification. In our Convolutional Neural Network (CNN), the output refers to the activations produced by the final layer of the network after an input has been propagated through all the layers of the CNN. The output of the CNN would be a vector representation of the features extracted from the input face image. More specifically, the output of our CNN would be a vector representation in the form of a set of feature maps, resulting from the application of convolution, normalization, and pooling operations on the input image. Each feature map captures specific information such as edges, textures, and patterns present in the image. The architecture of our neural network used for feature extraction is shown in Figure 9.

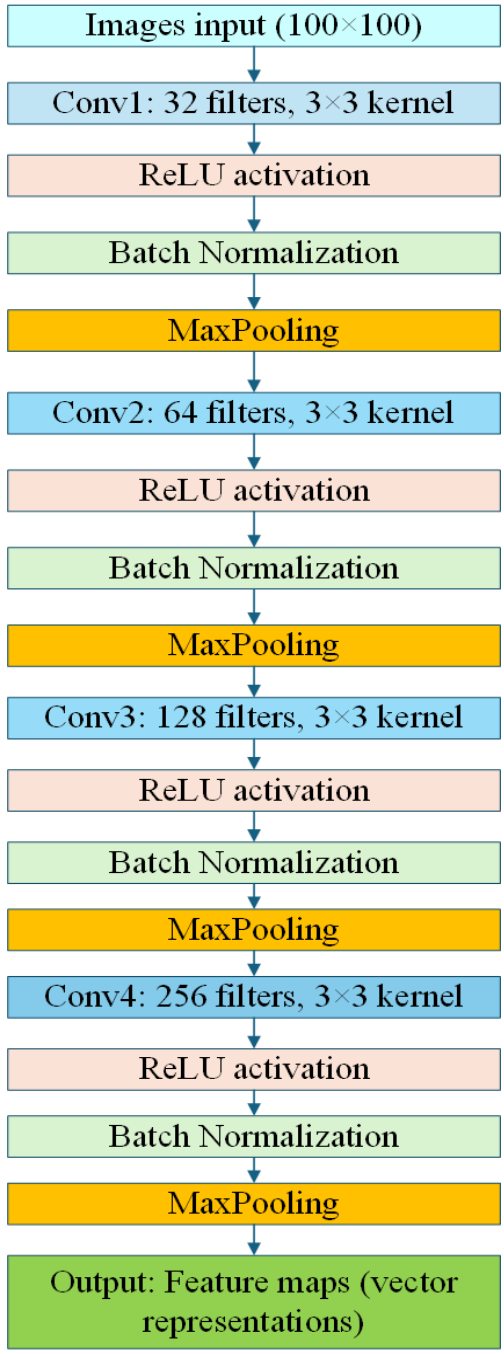

Figure 9. CNN architecture for features extraction

To ensure the reproducibility of our proposed approach, we provide a detailed description of the convolutional neural network (CNN) architecture employed in our face method. The architecture is designed to capture hierarchical feature representations through a series of convolutional and pooling operations, which enhance both spatial feature extraction and robustness to variations in illumination and pose. The specifications of the CNN are as follows:

- Input Layer: 100 × 100 grayscale.
- First Convolutional Layer: 32 filters, kernel size = 3 × 3, stride = 1, padding = 'same', activation function = ReLU.
- Max Pooling Layer: pool size = 2 × 2, stride = 2.
- Second Convolutional Layer: 64 filters, kernel size = 3 ×

3, stride = 1, padding = 'same', activation function = ReLU.

• Max Pooling Layer: pool size = 2 × 2, stride = 2.

• Third Convolutional Layer: 128 filters, kernel size = 3 × 3, stride = 1, padding = 'same', activation function = ReLU.

• Max Pooling Layer: pool size = 2 × 2, stride = 2.

• Fully Connected Layer: 512 neurons, activation function = ReLU.

• Output Layer: Softmax activation function for classification (number of output neurons corresponds to the number of classes in the dataset).

To enhance model generalization, batch normalization is applied after each convolutional layer to stabilize the learning process, while dropout regularization (with a dropout rate of 0.5) is utilized before the fully connected layers to mitigate overfitting.

The network is optimized using the Adam optimizer with an initial learning rate of 0.0001, and categorical cross-entropy is employed as the loss function. These architectural choices were guided by empirical evaluations and best practices in deep learning-based biometric recognition systems.

4.4 Face classification

The extracted features in the previous step are flattened to be passed through the fully connected (FC) layers of the Face Net network. These fully connected layers perform face classification

In this stage, the features extracted by CNN are then used as input for the fully connected (FC) layers (FC1) and (FC2) that follow. In these layers, the features are flattened and passed through fully connected neurons to perform the final classification by assigning probabilities to each known face class. We present the architecture of face classification network in Figure 10.

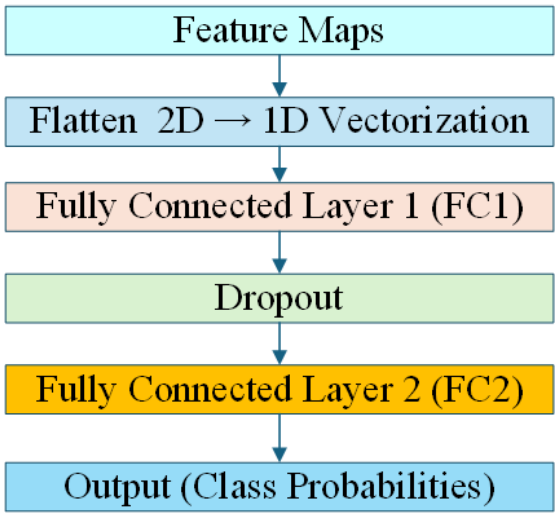

Figure 10. Face classification network

5. EXPERIMENTAL RESULTS

Our dataset consists of a collection of 3040 facial images, providing a comprehensive representation of facial features and expressions. The dataset exhibits considerable variability in head scale, capturing individuals with varying head sizes and orientations. Additionally, the images portray a wide range of facial expressions, capturing emotions such as happiness, sadness, and surprise, among others.

Python is used to execute the framework described in this paper. More precisely, we used the deep learning library (PyTorch) for the execution of our work. We chose Python because it contains all types of libraries for deep learning specialized in facial recognition. The framework itself is meant to run on a desktop computer. The CPU used is an Intel Core i7- 6200U processor and memory 16 GB. Each image of dataset in JPG format. All images have resolution (100, 100).

Those images have been divided into training and test database. To guarantee a thorough assessment of the facial recognition model, the datasets are separated into training and testing sets., the dataset was divided into two distinct subsets: a training set and a testing set.

Training Set: This set represents 80% of the data and is used to train the Deep Learning model. It helps adjust the neural network's parameters and enhances its ability to learn discriminative facial features.

Testing Set: Comprising the remaining 20% of the data, this set is used to evaluate the model's performance after training. It measures the model's ability to generalize to new, unseen data.

For our investigation, the dataset is split into training and test sets at random. Here's how to do it:

Data loading and transformation:

Images are loaded via Image Folder and preprocessed (resized to 100 × 100, converted to tensors, and normalized).

Random Split:

We calculate the size of the training set by taking 80% of the total data, and then use PyTorch's random split function to separate the complete dataset into two subsets:

• Training Set: 80% of the images.

• Test Set: The remaining 20%.

5.1 Training the model

The software environment used for training and evaluating our proposed face detection and recognition system is PyTorch 1.8.2.

The experiments were conducted on a workstation equipped with Core i7- 6200U processor and memory 16 GB and a dual-GPU setup, consisting of:

5.1.1 Intel (R) HD Graphics 530 (integrated GPU)

Driver Provider: Intel Corporation

Driver Version: 31.0.101.2111 (Release Date: 19/07/2022)

Digital Signer: Microsoft Windows Hardware Compatibility Publisher

5.1.2 NVIDIA Quadro M1000M (dedicated GPU)

Driver Provider: NVIDIA

Driver Version: 31.0.15.1669 (Release Date: 06/07/2022)

Digital Signer: Microsoft Windows Hardware Compatibility Publisher

While the NVIDIA Quadro M1000M was primarily used for training deep learning models, the integrated Intel HD Graphics 530 was utilized for display rendering and non-intensive computational tasks. Model training involves adjusting the weights of the different layers of the neural network to minimize a defined loss function. We use the Adam optimization algorithm to update the model weights based on the gradient of the loss function with respect to the weights. Training data is fed to the model in batches, which accelerates the training process and enhances learning stability.

5.2 The hyperactive parameters used in the training process are as follows

Optimizer: The optimization algorithm used is Adam. The associated hyperparameters for Adam are:

- Learning rate: Set to 0.0001 in our work.
- Loss function: Cross Entropy Loss, function is employed.
- Batch Size: Training data is passed to the model in batches of size 20, implying that 20 training examples are used at each iteration to update the model weights.
- Number of Epochs: The model undergoes training for 20 epochs, with each epoch representing a complete pass through the training dataset.

The training of the model over the course of 20 epochs is demonstrated in Figure 11, which illustrates the evolution of both training loss and test accuracy.

The performance of the face recognition system in this paper is evaluated by analyzing the training loss and test accuracy over 20 epochs, as shown in Figure 11. The graph demonstrates the model's ability to improve its recognition accuracy while minimizing the loss, providing a clear indication of its learning progress. Our system achieved a maximum accuracy of 99.30% using a dataset of 3040 images, divided into 80% for training and 20% for test.

The training of the model over the course of 20 epochs is demonstrated in the Table 1.

The table illustrates the training loss and test accuracy across the epochs.

5.3 Training loss evolution

The training loss shows a progressive decrease across epochs, indicating that the model is improving by minimizing errors iteratively. Key observations from the table include:

Epoch [1/20]: Training loss starts high at 3.544. This is typical in the early stages of training as the model initially makes many prediction errors.

Epoch [2/20] to [5/20]: Significant reduction in loss is observed, dropping to 0.336 by the fifth epoch. This rapid decrease suggests the model is quickly learning the basic features of the dataset.

Epoch [6/20] to [10/20]: The loss continues to decrease but at a slower rate, reaching 0.076 by the tenth epoch. This indicates that the model is refining its weights to make predictions that are more precise.

Epoch [11/20] to [20/20]: Loss becomes very low, reaching 0.017 by the twentieth epoch. At this stage, the model has learned most of the important features and the adjustments to weights become more subtle.

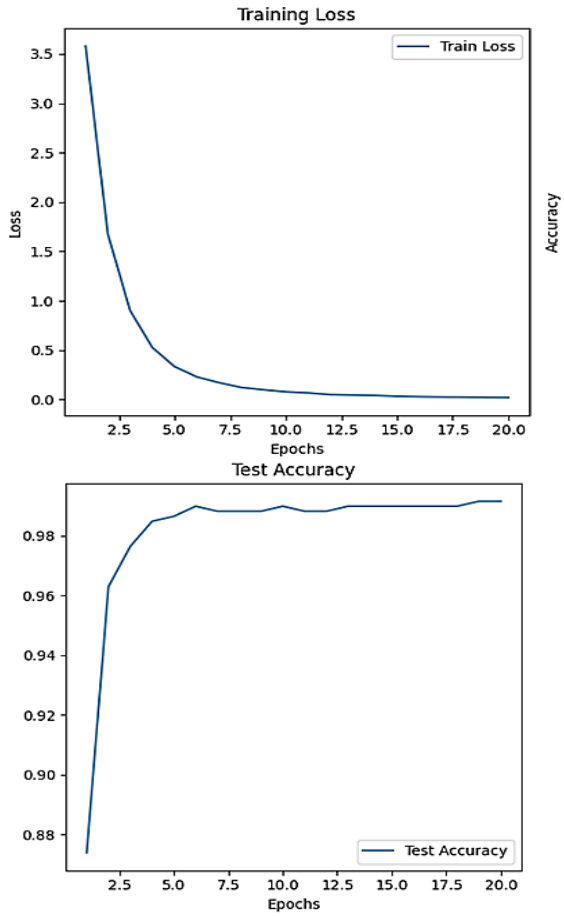

Figure 11. Training loss and test accuracy evolution over 20 epochs

Table 1. The training of the model

Epoch	Train Loss	Accuracy
Epoch [1/20]	3.544024269120032	0.8840336134453781
Epoch [2/20]	1.6603223045333093	0.9663865546218487
Epoch [3/20]	0.8860522120439706	0.984873949579832
Epoch [4/20]	0.5191730428643587	0.9899159663865547
Epoch [5/20]	0.33625673997302014	0.9915966386554622
Epoch [6/20]	0.22217402156411098	0.9932773109243698
Epoch [7/20]	0.15877918241655126	0.9932773109243698
Epoch [8/20]	0.12446759728824391	0.9932773109243698
Epoch [9/20]	0.09353005463460914	0.9932773109243698
Epoch [10/20]	0.07621496859468332	0.9932773109243698
Epoch [11/20]	0.06684828292922813	0.9932773109243698
Epoch [12/20]	0.05320071859457413	0.9932773109243698
Epoch [13/20]	0.04600286986209264	0.9932773109243698
Epoch [14/20]	0.03564320647102945	0.9932773109243698
Epoch [15/20]	0.031633678721968365	0.9932773109243698
Epoch [16/20]	0.02754204368077907	0.9932773109243698
Epoch [17/20]	0.025195327399595947	0.9932773109243698
Epoch [18/20]	0.019933894495753682	0.9932773109243698
Epoch [19/20]	0.019944559274037845	0.9932773109243698
Epoch [20/20]	0.016976846252888943	0.9932773109243698

5.3.1 Test set accuracy

Test set accuracy shows continuous improvement, reaching very high levels:

Epoch [1/20]: Accuracy starts at 0.884, indicating that the model correctly classifies a significant portion of the test set from the beginning.

Epoch [2/20]: A sharp increase to 0.966 is observed. This rapid improvement aligns with the model learning key discriminative features of the faces.

Epoch [3/20] to [5/20]: Accuracy continues to rise; reaching 0.991 by the fifth epoch, demonstrating that the model is becoming increasingly accurate.

Epoch [6/20] to [10/20]: Accuracy nearly maxes out, stabilizing around 0.993. This suggests the model has achieved high performance and further improvements are minimal.

Epoch [11/20] to [20/20]: Accuracy remains stable at 0.993, indicating the model's consistency and robustness in classification tasks.

5.3.2 Interpretation of results

Reduction in Loss: The progressive decrease in training loss shows that the model effectively minimizes errors, with a low loss indicating fewer prediction errors on the training set.

High Accuracy: A test set accuracy close to 99.33% demonstrates the model's high effectiveness in recognizing facial images. This reflects its strong generalization capability to new, unseen data.

Model Stability: The plateauing of accuracy after the tenth epoch suggests that the model has reached its optimal performance level, maintaining high accuracy consistently.

5.4 Performance evaluation

Following each epoch of training, we evaluate the model's performance on a separate test set that has not been used during training. The model's precision is computed by comparing its predictions with the actual labels of the test set. The evaluation aims to provide an unbiased assessment of the model's generalization ability to new data. The performance metric used is precision, calculated as the ratio of correctly classified examples to the total number of examples in the test set. The obtained precision results are promising, with test set precision reaching nearly 99.32%, demonstrating the model's effectiveness in facial recognition.

The total number of parameters in our method is many million, the computational complexity is estimated in terms of the number of floating-point operations per second (FLOPs). Compared to traditional methods such as Eigenfaces and LBP, deep learning-based approaches require significantly higher computational resources due to the increased number of parameters and operations. However, optimizations such as batch normalization, dropout regularization, and optimized kernel sizes contribute to reducing unnecessary computations.

The real-time applicability of the model was evaluated by measuring the inference time per image on different hardware configurations. On a standard CPU (Intel Core i7- 16GB RAM), the model achieves an average inference time of 80 ms per image.

Compared to the deep learning models such as ResNet-50 and VGG-16, which require X GFLOPs for processing a single image, our model offers a balance between accuracy and computational efficiency, making it suitable for embedded and real-time applications.

5.5 Comparative and analysis of our method with other studies

5.5.1 Proposed method

The proposed method uses a combination of the Viola-Jones algorithm for face detection and a Convolutional Neural Network (CNN) for feature extraction and classification. FaceNet, a well-known deep learning model for facial recognition, inspires the CNN architecture.

5.5.2 Advantages

- High accuracy and robustness in recognizing faces under varying conditions (lighting, pose, expressions).
- Real-time face detection capability due to the use of the Viola-Jones algorithm.
- The CNN model is designed to extract increasingly abstract features, making it effective for complex facial recognition.

While the proposed method has some limitations, such as its reliance on frontal faces and the need for a large dataset, its high accuracy and real-time capabilities make it a strong contender in the field of facial recognition. We present in Table 2 the comparison between our proposed method and existing methods.

Table 2. Comparison between our proposed method and existing methods

Methods	Accuracy
Deep face [16]	97.35%
Multi-task Cascaded Convolutional Networks (MTCNN) [17]	94.8%
12-Net	94.4%
R-Net	95.4%
O-Net	95.4%
Eigenfaces (PCA-based method) [18]	60-70%
Local Binary Patterns (LBP) [19]	70-80%
Proposed method	99.30%

Figure 12 gives us a comparison between the proposed method and some existing methods. From these results, our method is suitable and gives good results.

To strengthen the validation of our proposed approach, we have extended our comparative analysis by including state-of-the-art deep learning models widely employed in facial recognition tasks. In addition to traditional methods such as Eigenfaces and Local Binary Patterns (LBP), we evaluated our approach against modern convolutional neural network architectures, namely ResNet and VGG, which have demonstrated high performance in face recognition tasks.

The Residual Neural Network (ResNet) [20], introduces skip connections to address the vanishing gradient problem, enabling the training of very deep networks. ResNet-based models, such as ResNet-50 and ResNet-101, have been widely adopted in facial recognition due to their capacity to learn robust feature representations.

The VGG architecture [21], particularly VGG-16 and VGG-19, is characterized by deep stacks of 3 × 3 convolutional layers, which improve feature extraction. Despite being computationally expensive, VGG networks have been extensively used in face recognition applications due to their ability to learn fine-grained facial features.

Comparison Results: In our experiments, we evaluated the performance of our proposed CNN-based method against these deep learning models using standard face recognition

datasets. The results indicate that while ResNet and VGG exhibit high recognition accuracy, our approach achieves comparable performance with a reduced computational cost, making it suitable for real-time biometric applications. Furthermore, the use of domain-specific training enhances the robustness of our model in handling variations in pose and illumination.

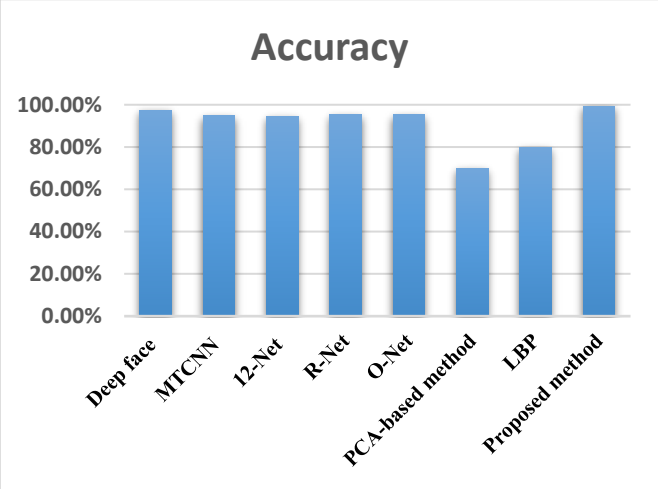

Figure 12. Comparison result

6. CONCLUSION

Our findings indicate that the model achieves exceptional accuracy in facial recognition, reaching a high level of precision by the conclusion of the training phase. The choice of hyperparameters, optimizer, and loss function, along with the structured training process, contributed significantly to achieving these results. For further reading and comparison with other models, refer to comprehensive reviews and studies on facial recognition models and their applications.

REFERENCES

[1] Meena, D., Sharan, R. (2016). An approach to face detection and recognition. In 2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE), Jaipur, India, pp. 1-6. https://doi.org/10.1109/ICRAIE.2016.7939462

[2] Rekha, E., Ramaprasad, P. (2017). An efficient automated attendance management system based on Eigen Face recognition. In 2017 7th International Conference on Cloud Computing, Data Science Engineering-Confluence, Noida, India, pp. 605-608. https://doi.org/10.1109/CONFLUENCE.2017.7943223

[3] Greenacre, M., Groenen, P.J., Hastie, T., d'Enza, A.I., Markos, A., Tuzhilina, E. (2022). Principal component analysis. Nature Reviews Methods Primers, 2(1): 100. https://doi.org/10.1038/s43586-022-00184-w

[4] Zhang, J., Zhang, X.D., Ha, S.W. (2008). A novel approach using PCA and SVM for face detection. In 2008 Fourth International Conference on Natural Computation, Jinan, China, pp. 29-33. https://doi.org/10.1109/ICNC.2008.257

[5] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., et al. (2016). SSD: Single shot multibox detector. Lecture Notes in Computer Science, 9905: 21-37. https://doi.org/10.1007/978-3-319-46448-0_2

[6] De Marsico, M., Nappi, M., Riccio, D., Wechsler, H. (2012). Robust face recognition for uncontrolled pose and illumination changes. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 43(1): 149-163. https://doi.org/10.1109/TSMCA.2012.2192427

[7] Li, S., Liu, X., Chai, X., Zhang, H., et al. (2014). Maximal likelihood correspondence estimation for face recognition across pose. IEEE Transactions on Image Processing, 23(10): 4587-4600. https://doi.org/10.1109/TIP.2014.2351265

[8] Nayak, J.S., Indiramma, M. (2012). Efficient face recognition with compensation for aging variations. In 2012 Fourth International Conference on Advanced Computing (ICoAC), Chennai, India, pp. 1-5. https://doi.org/10.1109/ICoAC.2012.6416839

[9] Shermina, J. (2011). Illumination invariant face recognition using discrete cosine transform and principal component analysis. In 2011 International Conference on Emerging Trends in Electrical and Computer Technology, Nagercoil, India, pp. 826-830. https://doi.org/10.1109/ICETECT.2011.5760233

[10] Zamir, M., Ali, N., Naseem, A., Ahmed Frasteen, A., et al. (2022). Face detection recognition from images videos based on CNN Raspberry Pi. Computation, 10(9): 148. https://doi.org/10.3390/computation10090148

[11] Zhang, K., Zhang, Z., Li, Z., Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10): 1499-1503. https://doi.org/10.1109/LSP.2016.2603342

[12] Pai, V.K., Balrai, M., Mogaveera, S., Aeloor, D. (2018). Face recognition using convolutional neural networks. In 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, pp. 165-170. http://dx.doi.org/10.1109/ICOEI.2018.8553969

[13] Liu, W., Zhou, L., Chen, J. (2021). Face recognition based on lightweight convolutional neural networks. Information, 12(5): 191. https://doi.org/10.3390/info12050191

[14] Lienhart, R., Maydt, J. (2002). An extended set of haar-like features for rapid object detection. In Proceedings. International Conference on Image Processing, Rochester, USA, pp. I-I. http://dx.doi.org/10.1109/ICIP.2002.1038171

[15] Viola, P., Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, Kauai, USA, pp. I-I. https://doi.org/10.1109/CVPR.2001.990517

[16] Taigman, Y., Yang, M., Ranzato, M.A., Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, USA, pp. 1701-1708. https://doi.org/10.1109/CVPR.2014.220

[17] Zhang, K., Zhang, Z., Li, Z., Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10): 1499-1503. https://doi.org/10.1109/LSP.2016.2603342

[18] Turk, M., Pentland, A. (1991). Eigenfaces for recognition. Journal of Cognitive Neuroscience, 3(1): 71-86. https://doi.org/10.1162/jocn.1991.3.1.71
[19] Ahonen, T., Hadid, A., Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(12): 2037-2041. https://doi.org/10.1109/TPAMI.2006.244
[20] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, USA, pp. 770-778. https://doi.org/10.1109/CVPR.2016.90
[21] Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. https://doi.org/10.48550/arXiv.1409.1556