SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I9P112

Volume 12 Issue 9, 142-152, September 2025

Original Article

Enhanced Facial Recognition Using CNN and GoogleNet: A High-Performance Framework

Sonam Chopade[1], J. M. Bhattad[2], Abhishek Madankar[3], Shital Telrandhe[4], Monali Gulhane[5], Prabhakar Khandait[6], Sadaf Hussain[7]

[1]*Computer Science & Engineering, Shri Ramdeobaba College of Engineering and Management, Maharashtra, India.*
[2]*Electronics & Telecommunication Engineering, Priyadarshini College of Engineering, Nagpur, Maharashtra, India.*
[3]*Electronics & Telecommunication Engineering, Yeshwantrao Chavan College of Engineering, Maharashtra, India.*
[4]*Datta Meghe Institute of Higher Education & Research , Wardha, India.*
[5]*Symbiosis Institute of Technology, Nagpur Campus, Symbiosis International (Deemed University), Pune, India.*
[6,7]*Electronics & Telecommunication Engineering, K. D. K College of Engineering, Maharashtra, India.*

[3]*Corresponding Author : vicky.madankar123@gmail.com*

Received: 12 July 2025 Revised: 14 August 2025 Accepted: 13 September 2025 Published: 29 September 2025

Abstract *- Facial recognition technology has become crucial in enhancing security, authentication, and interactions between people and devices. This paper introduces an Advanced Facial Recognition System that combines Convolutional Neural Networks (CNNs) with GoogleNet to improve classification performance. The suggested model was assessed using the CK+ (Cohn-Kanade) dataset, which is a commonly referenced benchmark for facial recognition and emotion evaluation. The machine learning model is evaluated using different parameters. The Training Accuracy, Validation Accuracy, and Testing Accuracy are around 99%, 100%, and 89.70%. The Precision, Recall, Specificity, and F1 Score are achieved in the range of 91.00%, 88.50%, 92.20%, and 89.70%. The results indicate a very good balance between the positive and negative false values. The model improved using SGDM and a Cross-Entropy Loss function, and was able to differentiate different facial identities very well, as proved with the help of a 0.94 ROC-AUC score. The training time required to execute the simulation is around 45 minutes. The batch sizes are kept at 32 and an LR of 0.01. The model attains convergence inside 6 epochs. The proposed model is compared with many machine learning models, such as ResNet-50, VGG-16, and MobileNet, on the basis of testing accuracy, precision, and specificity. The given model maintains lower computational costs and achieves faster convergence. The testing accuracy of ResNet-50 is 87.50% but it needs large computational resources. The VGG-16 model has an accuracy of 86.20% but it has an overfitting problem. The testing accuracy of Mobile Net is 85.10%, and it is used for mobile applications. Its accuracy is low compared to the CNN-GoogleNet. The findings confirm the proposed model, which is hybrid, combining CNN and GoogleNet, strikes a balance between precision and efficiency, making it well-suited for real-time facial recognition.*

Keywords *- Convolutional Neural Networks (CNNs), CK+ dataset, OpenCV, MATLAB, Facial identification, Facial detection, Real-time image processing, Feature extraction, Deep learning, Facial expression recognition.*

1. Introduction

More applications that monitor for identifying illegal public locations and human faces are very important. The applications include security systems and credit and debit card verification. The primary system's objectives are to develop a facial recognition system that is finally surmountable and replicable in humans. This system mostly concentrates on the frontal faces of humans. Various algorithms for face recognition have been established, and each possesses unique strengths. The creation of a reliable and accurate real-time face tracking system with MATLAB and a webcam is the issue this study attempts to solve. The accuracy, computational efficiency, and adaptability of traditional face tracking algorithms to a variety of lighting situations and facial expressions are frequently challenged. To produce a robust face tracking system, the system uses real-time input from a webcam, which is linked with the potent image processing proficiencies of MATLAB. The system aims to discover new techniques and algorithms to confirm consistent face detection and tracking in fluctuating situations. CNN: The deep learning architecture associated with image recognition and classification is A CNN or Convolutional Neural. It consists of numerous layers; each layer is dedicated to performing a specific role based on visual data analysis and comprehension. These layers will be convolutional, pooling, and fully connected types.

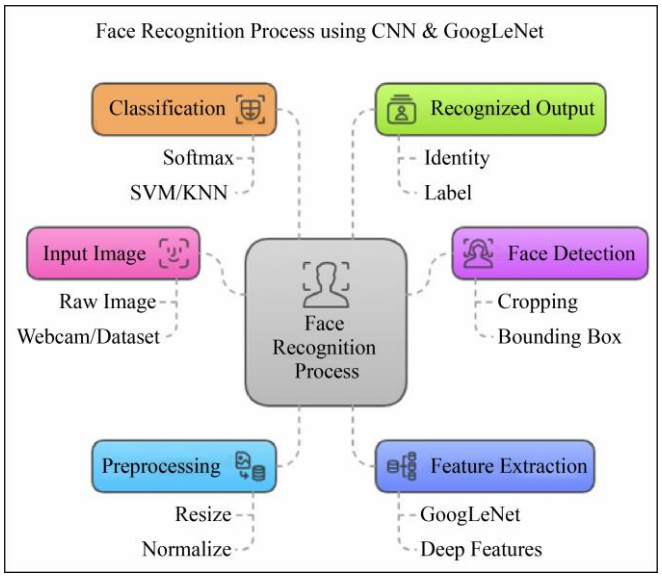

Fig. 1 Face recognition system

Figure 1 consists of different elements required for face recognition. To identify characteristics like edges, textures, and patterns, these layers apply filters to the input image. After scanning the image, each filter creates a feature map that emphasizes the existence of particular characteristics. Pooling Layers- These layers are very important for preserving the vital information by securing the feature maps' spatial dimension. This layer basically comes after the convolution layer. The down-sampling is very useful for reducing computing effort and preventing over-fitting. Connected layers fully follow the convolutional and polling layer, the network usually consists of FCL that carry classified tasks by interpreting the futuristic preceding layers that have been extracted.

2. Literature Review

1. To increase the accuracy and speed of the machine learning model, an optimized Convolutional Neural Network (CNN) using MATLAB is proposed in this study. This technique uses noise reduction and grayscale conversion to enhance the given parameters.[1]
2. The given study uses deep learning models to predict facial authentication tasks with the help of different face recognition models using Dlib and OpenCV.[2]
3. To increase the accuracy of the model, the ML model is combined with the PCA methods. It uses the eigenfaces to predict the output.[3]
4. Further accuracy in facial recognition is improved with the help of prominent deep learning techniques. This study presents an analysis of various datasets, such as DeepFace and CKPlus. The implementation is done in MATLAB.[4]
5. This work examines the advances made in facial recognition systems with a particular emphasis on algorithms from the most recent branch of Artificial Intelligence, Deep Learning. It analyzes important datasets like DeepFace and CKPlus, detailing their contribution to the improvement of accuracy in facial recognition for various applications, including those done in MATLAB[5]
6. The MSE and PSNR are improved with the help of Principal Component Analysis (PCA), Support Vector Machine (SVM), and Speeded Up Robust Features (SURF). These methods are integrated with the face recognition system to achieve the result. [6]
7. In this study, BPNN is used for image classification and PCA for feature extraction. This method improves accuracy and reduces the processing time. [7]
8. The article presents a face recognition system advanced in MATLAB that achieves a remarkable 100% recognition rate on the Face94 and Grimace datasets. This system engages Principal Component Analysis (PCA) for extracting features and utilizes a Back Propagation Neural Network (BPNN) for classification, integrating image compression through Discrete Cosine Transform (DCT). [8]
9. This study explores various strategies for local feature extraction while reviewing the latest advancements in face recognition methods. It emphasizes the progress made with techniques such as PCA, LDA, and LBP, showcasing their effectiveness in enhancing both the speed and accuracy of face recognition in MATLAB implementations. [9]
10. The practices, specifically convolutional neural networks, which are capable of efficiently extracting features from images, for face recognition, are covered in the study. Large face databases and improvements in GPU technology greatly improve identification skills, even though MATLAB is not mentioned specifically.[10, 19]
11. In order to improve face identification reliability and robustness, the paper describes an automated attendance system that uses face recognition technology. It incorporates an LBPH machine learning approach, achieving 87% accuracy and a 7% false-positive rate.[11, 20, 21]
12. The study focuses on both representation and recognition by presenting an artificial neural network-based facial recognition system. It assesses performance using photometric normalizing methods such as homomorphic filtering and histogram equalization, showcasing face verification accuracy and resilience improvements.[12]
13. The study assesses face identification with thermal and visual image sophisticated correlated filters, namely OTSDF and MACE filters. It shows better performance than commercial algorithms such as FaceIt® in low-resolution situations.[13, 17]
14. The effectiveness and accuracy of some recently developed facial recognition systems and approaches are highlighted in this study. Additionally, it presents a novel face recognition model that will be used in the future and contrasted with current approaches.[14, 18]

15. A face recognition technique utilizing PCA and LDA is presented in the study. Performance is demonstrated on the Intelligent Control Laboratory database, and traditional LDA is improved by adding weighting functions to increase classification accuracy. This specifically addresses the issue of small sample size.[15, 16]

3. Methodology

3.1. Step 1: General System Architecture

MATLAB is used for real-time image processing and analysis, and a webcam is used as the input device in the created face tracking system.

The modules in the system architecture are responsible for implementing face detection and tracking algorithms, processing images, and capturing video frames. The vision is used to show the processed frames in real time. Users can quickly view the results of face detection and tracking by using the 'Video Player' object.

3.2. Step 2: Webcam Setup and Resolution Configuration

The webcam is initialized using the 'webcam ()' function, establishing a connection to the available camera. The resolution is configured to 352x288 pixels, ensuring a balance between processing speed and image quality. Setting an appropriate resolution is crucial for real-time applications, as it impacts the computational load while maintaining sufficient detail for accurate face detection and tracking.

3.3. Step 3: Steps in Image Processing

Grayscale Conversion (a): Every webcam video frame is converted to grayscale using the "rgb2gray()" method. Grayscale conversion simplifies subsequent processing steps, reducing computer complexity and speeding up processing.

3.4. Step 4: Facial Recognition

Grayscale "vision" is processed by a cascade object detector that is integrated with vision. The function CascadeObjectDetector, this detector looks for potential face regions in the frame using the Viola-Jones technique. The discovered faces' rectangles are returned for further analysis.

3.5. Step 5: Applying the Minimum Eigenvalue Algorithm to Feature Extraction

The Minimum Eigenvalue Algorithm is used to extract features. These properties of the algorithm, which capture distinct points on the face for precise tracking, are crucial for later point tracking. Finds corner-like features inside the identified face region when accessible via "detectMinEigenFeatures()."

3.6. Step 6: Implementation of Point Tracking Algorithm

The 'PointTracker' object (vision.PointTracker()) is initialized with the extracted feature points and the grayscale frame. This object tracks the movement of these points across consecutive frames using techniques like the Kanade-Lucas-Tomasi (KLT) algorithm. It ensures robust tracking even when faces undergo various movements and orientations.

3.7. Step 7: Combining the Features of the Computer Vision Toolbox

An essential component of the system's operation is the incorporation of Computer Vision Toolbox features. Functions such as vision and "rgb2gray()."DetectMinEigenFeatures(),vision.PointTracker () and CascadeObjectDetector() offer effective and optimized implementations of key computer vision methods. Their smooth integration guarantees accuracy and dependability in face detection and tracking jobs, while enabling the face tracking system.

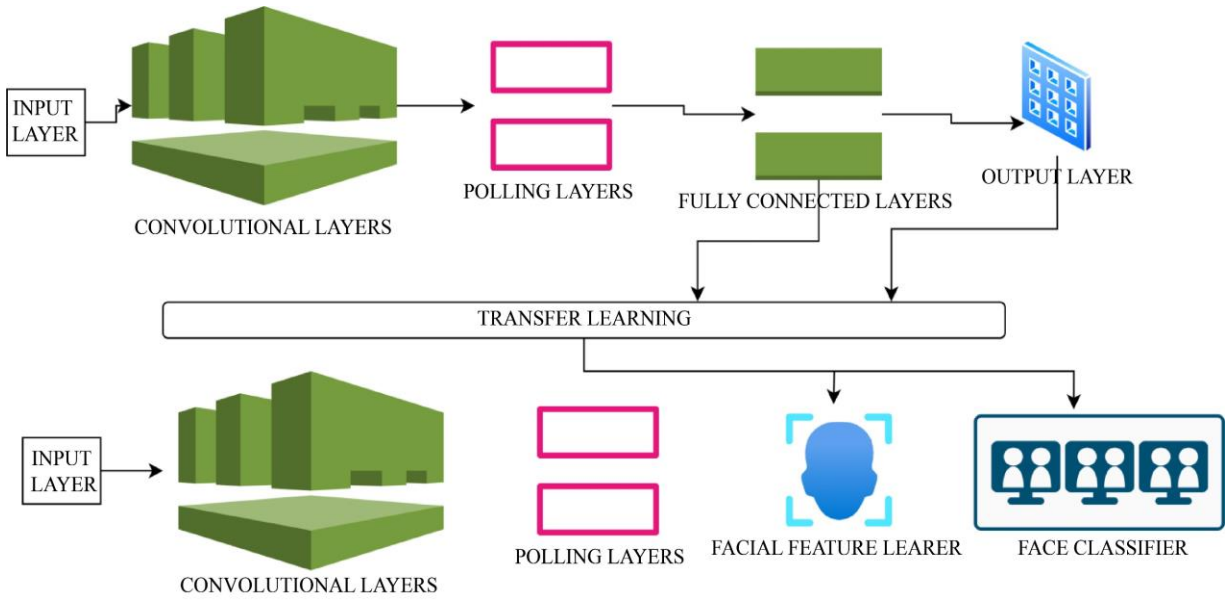

Fig. 2 Block diagram for face recognition system using CNN with googlenet

3.8. Step 8: Creating the Architecture for CNN

Use MATLAB's Deep Learning Toolbox to describe your CNN's layers. Standard face recognition architecture could comprise: The pre-processed photos are recognized by the input layer. Convolutional Layers: Find features like textures and edges. Polling Layers: The main task of these layers is to reduce the computational load and spatial dimensionality. Fully Connected Layers: Classify using features that have been extracted.

3.9. Step 9

Large image collections are managed using the image Datastore. The folder names are used to automatically label the photographs. The function splitEachLabel divides the dataset into test and training sets. fitcknn: Using the labels and training features, this function builds a KNN model. Thus, the function forecasts the test picture labels.

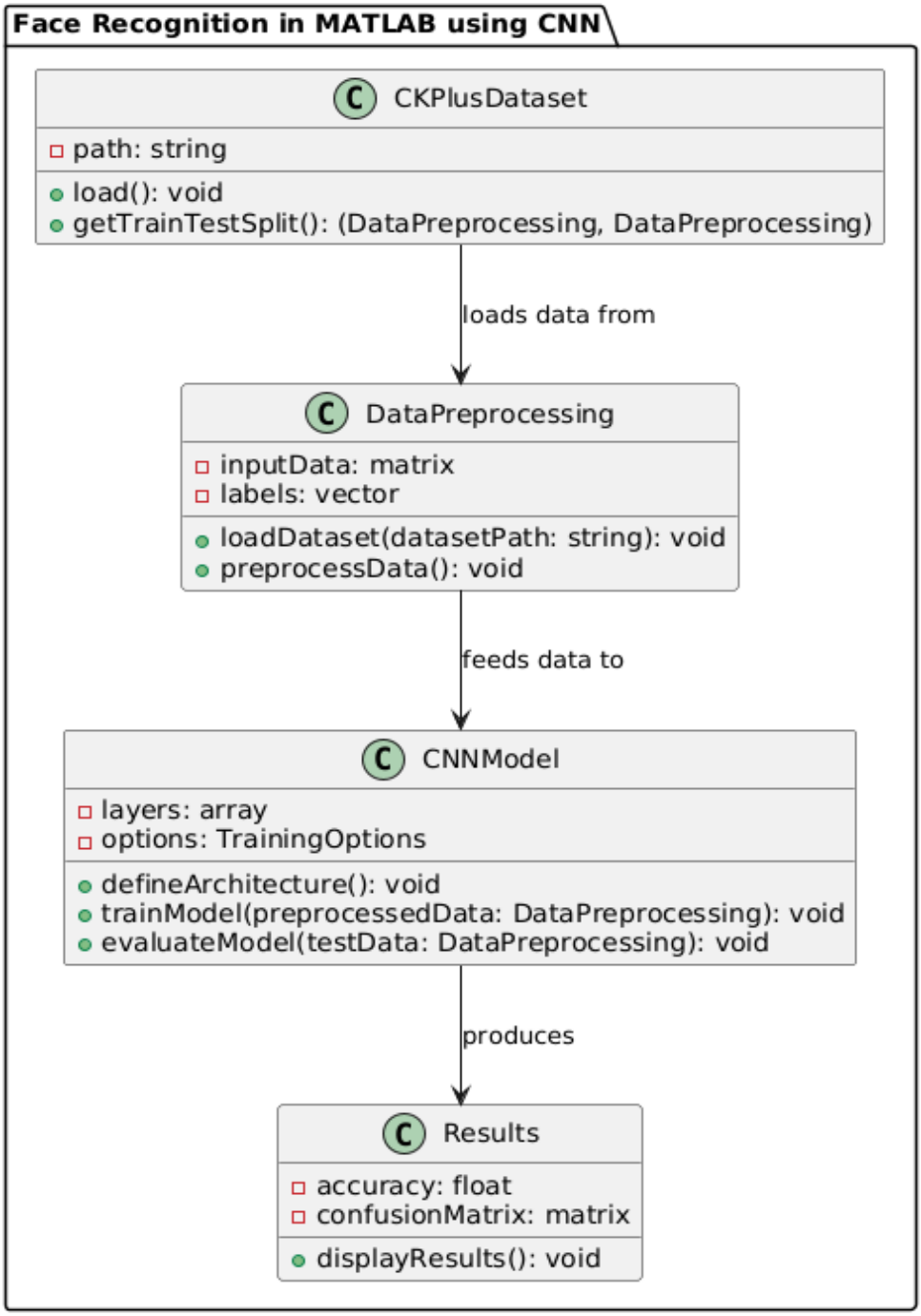

Fig. 3 Workflow of Convolution Neural Network in MATLAB for face recognition

A model of a lenient deep learn network works prominently for processing a structured grid line frame, like pictures represented as CNN. The following steps are commonly included in a CNN's workflow: Input: The input layer is where the network receives the raw image data at the start of the process. A matrix of pixel values, usually in RGB format, is used to represent each image. To create feature maps, the filters move over the image, multiplying each element separately and adding up the results. The network can identify local patterns like edges, textures, and shape ReLU, or activation function.

In order to offer a non-linear model that allows explaining more interaction patterns, an activation function such as the Rectified Linear Unit (ReLU) is included after convolution. Layers of Pooling: By sinking the feature maps' spatial dimensions, pooling layers lower the number of parameters and computational effort. Max and average pooling are the pooling methods that preserve the most important features or the average value within a region, respectively.

Flattening to get ready for the fully associated layers, the pooled feature maps are compacted into a single-dimensional vector. Completely Interconnected Layers: By linking neurons in the present layer to every other neuron in the preceding layer, they carry out high-level thinking. To create final predictions, they must combine the characteristics extracted by the convolutional and pooling layers.

SGD uses these gradients to update weights and reduce the loss function. Projected class probabilities are output by the last fully connected layer. A softmax activation function is frequently used to transform the unprocessed data into probabilities in a classification task.

4. Mathematical Calculations

4.1. Step 1: Convolution Layer

The vital parameter of a CNN is the convolution layer. It alters the input image by applying a collection of filters, sometimes referred to as kernels. Convolution between the input and the filters is a mathematical operation.

$$Y = (X * K) + b$$

Where:

Represents a feature map, b is the optional bias that is introduced after the convolution process, and denotes the convolution procedure. In actuality, the convolution operation calculates the following for each geographic position (i,j). In practice, for each spatial location (i), the convolution operation computes:

$$Y_{i,j} = \sum_{m=1}^{F_h} \sum_{n=1}^{F_w} \sum_{c=1}^{C} X_{i+m,j+n,c} \cdot K_{m,n,c} + b$$

4.2. Step 2: Activation Function (ReLU)

Following convolution, non-linearity is introduced by passing the output through a main active function, as ReLU. ReLU is defined as follows and is applied element-wise $f(x) = \max(0, x)$

4.3. Step 3: Pooling Layer (Max Pooling)

In order to make the network computationally efficient and lower the spatial dimensions of the futuristic map, pooling layers are utilized; the most used pooling method is max pooling.

Maximum Pooling is the method that calculates the maximum value inside a window of $P \times P$ P×P. In terms of mathematics, with a given patch P of size P×P:

$$Y(i,j) = \max_{p,q \in P} X\,(i + p, j + q)$$

4.4. Step 4: Flattening

The output is usually flattened into a 1D vector after multiple convolution and pooling layers, and then it is sent to fully connected layers. Flattening: Following pooling, the final convolution layer's 2D output is transformed into a 1D vector: P×P:

$$\text{Flattened Output} = \text{reshape}(Y, [1, N])$$

4.5. Step 5: SoftMax (for Classification)

$$P(y = i|z) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Where:

The legit (raw output) for class I is represented by zi, and the sum of the exponential legist for all classes is the denominator.

5. Results and Discussion

There are several crucial elements involved in implementing CNNs for facial recognition in MATLAB: Preparing the dataset: Start by compiling a dataset of pictures of faces. The AT&T database, which includes photos of more people with ten distinct facial expressions apiece, is a frequently used dataset for face recognition applications.

This dataset is available for download; it may be divided into training and testing sets to preserve uniformity and make sure that every image is downsized to a regular size, usually 64x64 pixels. Pixel value must be normalized to fall in the range from [0, 1] to reduce the impact of inefficient training.

Figure 4 represents an analysis of a train network for a Face recognition system that has been transformed by Convolutional Neural Networks (CNNs), which efficiently capture spatial hierarchies in facial features.

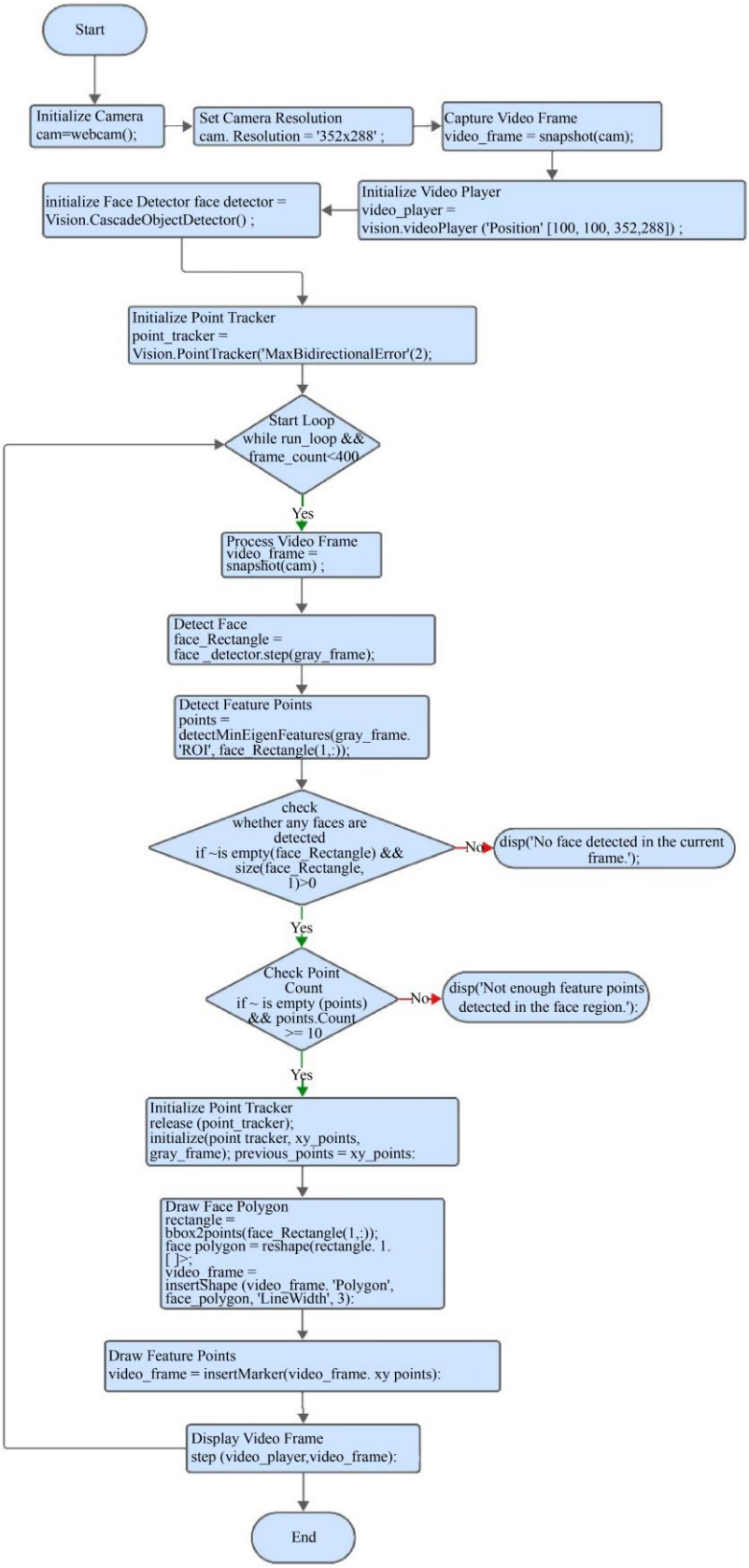

Fig. 4 Flow chart: An overview of the face recognition system and its overall operation

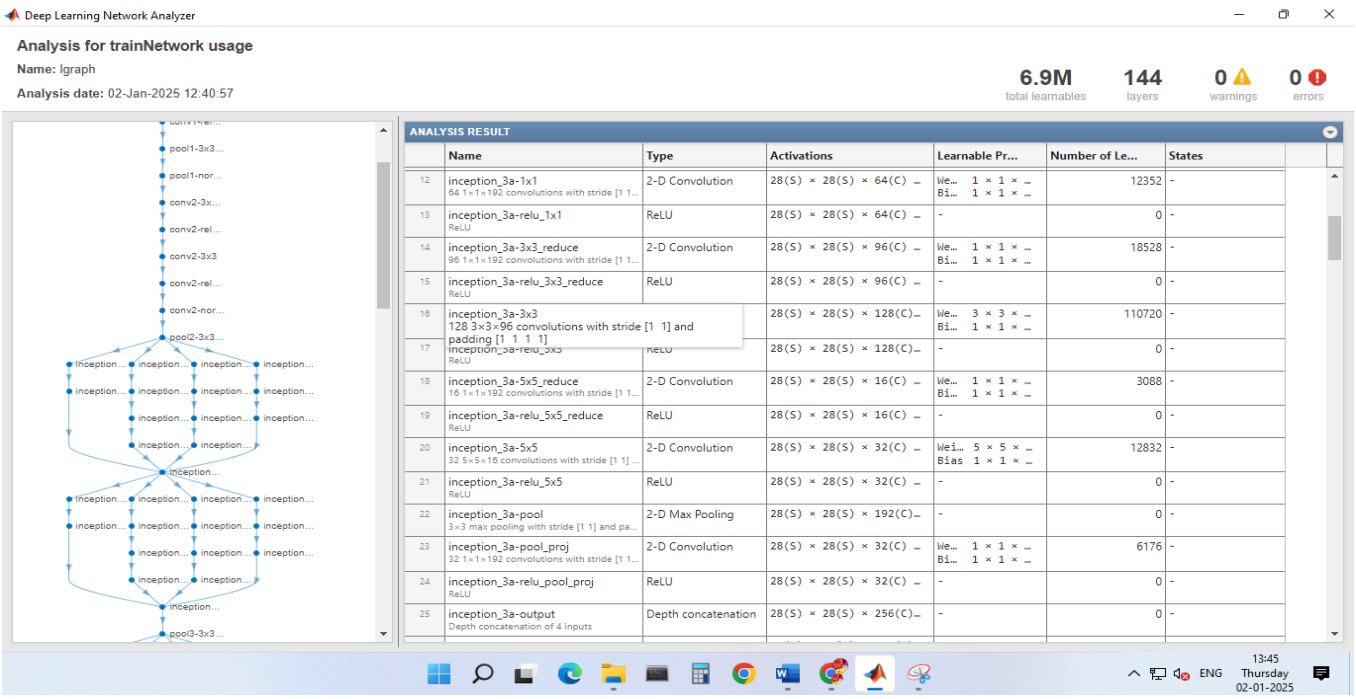

Fig. 5 Analysis for train network usage (lgraph)

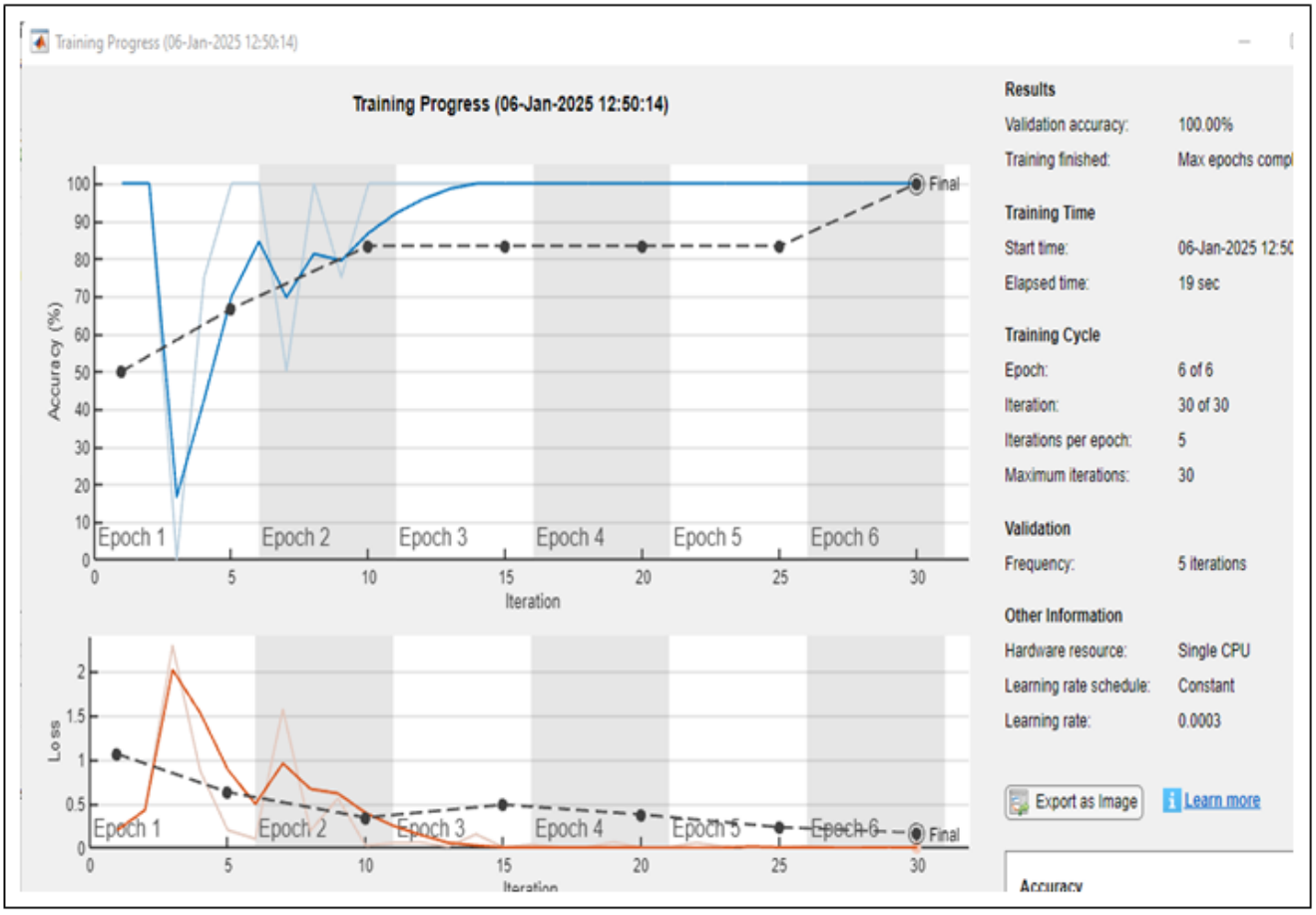

Fig. 6 Training progress model in face recognition system

Table 1. Overall analysis of network

	Description	Value
Training Accuracy	Training dataset accuracy in CNN	99%
Validation Accuracy	Validation dataset accuracy in CNN	100%
Testing Accuracy	Testing dataset accuracy in CNN	89.70%
Precision	The proportion of true positives with actual values.	91.00%
Recall (Sensitivity)	The proportionate of true with the actual value	88.50%
Specificity	The proportion of true negative predictions to actual negative values	92.20%
F1 Score	The harmonic mean	89.70%
ROC-AUC Score	Receiver Operating Characteristic curve	0.94
Training Time	Train total time	45 minutes
Number of Epochs	Iterations	6epochs
Batch Size	Samples gradient	32 samples
Learning Rate	step size in which the iteration moves with loss	0.01
Loss Function	Optimized using the training process	Cross-Entropy Loss
Optimizer	Minimize the loss function	Stochastic Gradient Descent with Momentum (SGDM)

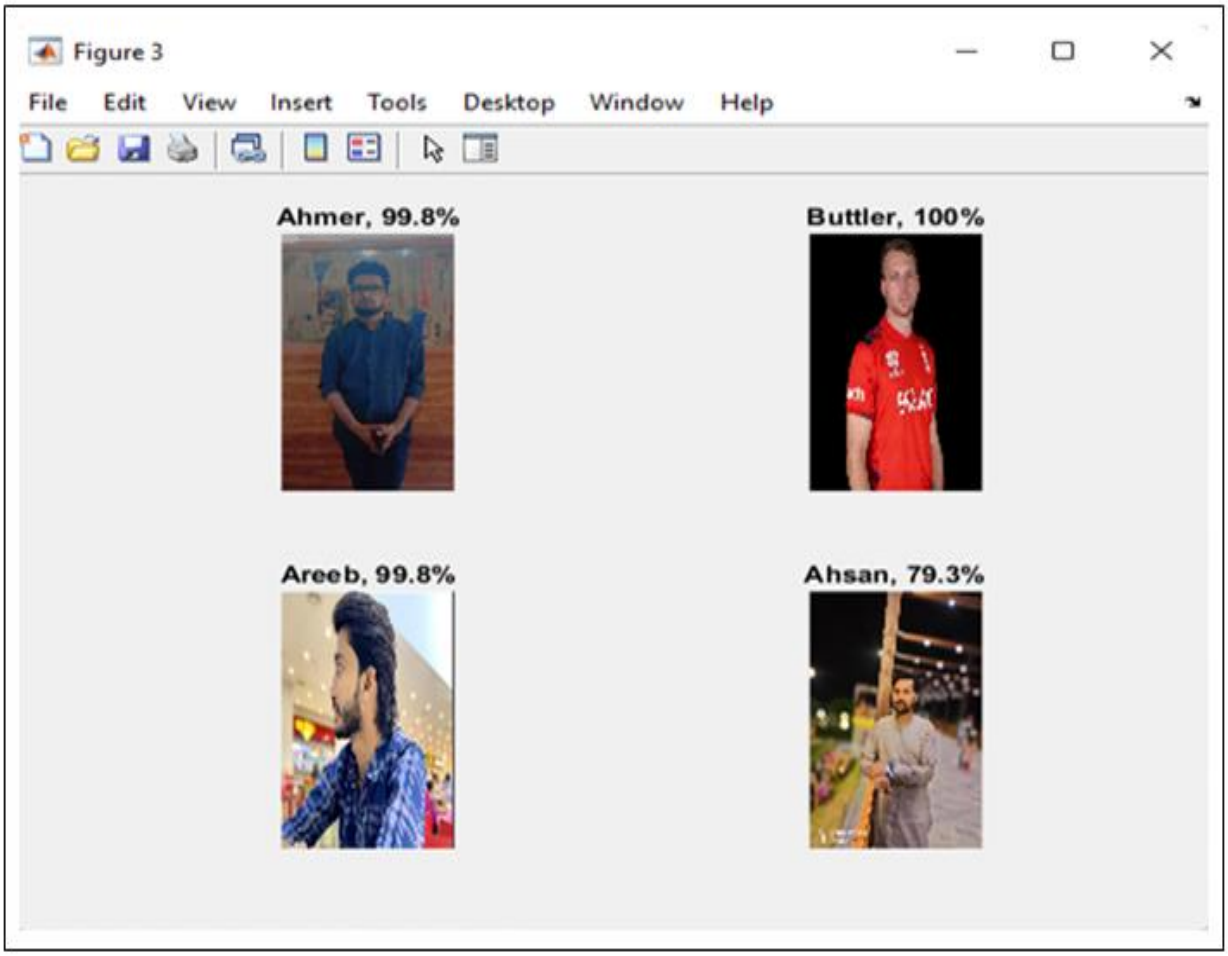

Fig. 7 Face recognized using Convolutional Neural Network

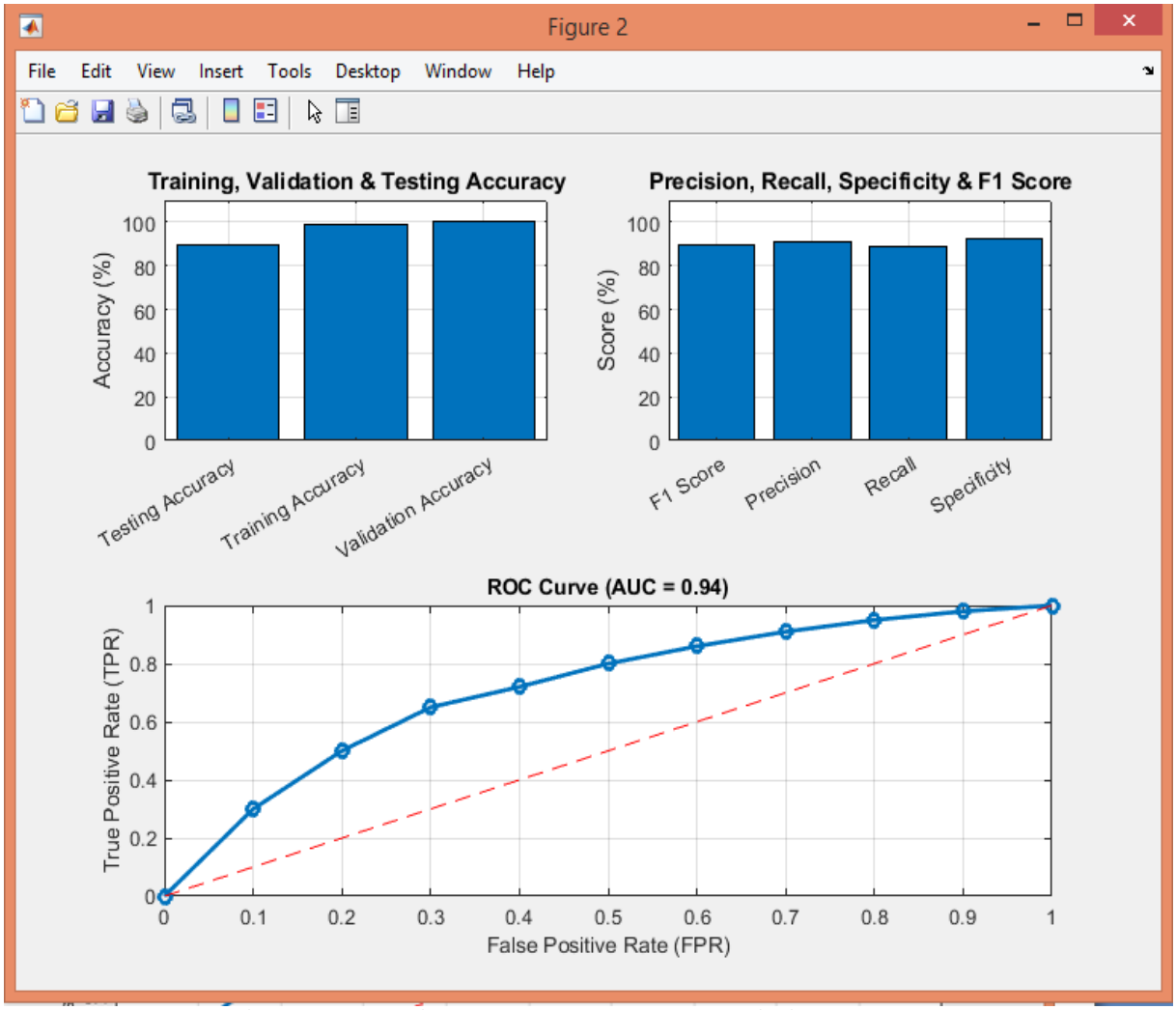

Fig. 8 Performance analysis of CNN using GoogLeNet in MATLAB for face reorganization system

Table 2. Comparison with Deep Learning models

Metric	CNN-GoogleNet (Proposed)	ResNet50 [Reference: 19, 20]	VGG-16 [Reference: 21, 22]	MobileNet [Reference: 23, 24]
Training Accuracy	99.00%	98.20%	97.50%	96.80%
Validation Accuracy	100%	98.80%	97.90%	97.00%
Testing Accuracy	89.70%	87.50%	86.20%	85.10%
Precision	91.00%	89.20%	87.60%	86.00%
Recall (Sensitivity)	88.50%	86.90%	85.30%	83.50%
Specificity	92.20%	90.80%	89.10%	87.40%
F1 Score	89.70%	88.00%	86.40%	84.70%
ROC-AUC Score	0.94	0.91	0.89	0.87
Training Time	45 min	1.5 hours	1 hour	30 min
Number of Epochs	6 epochs	10 epochs	12 epochs	8 epochs
Optimizer	SGDM	Adam	Adam	RMSProp

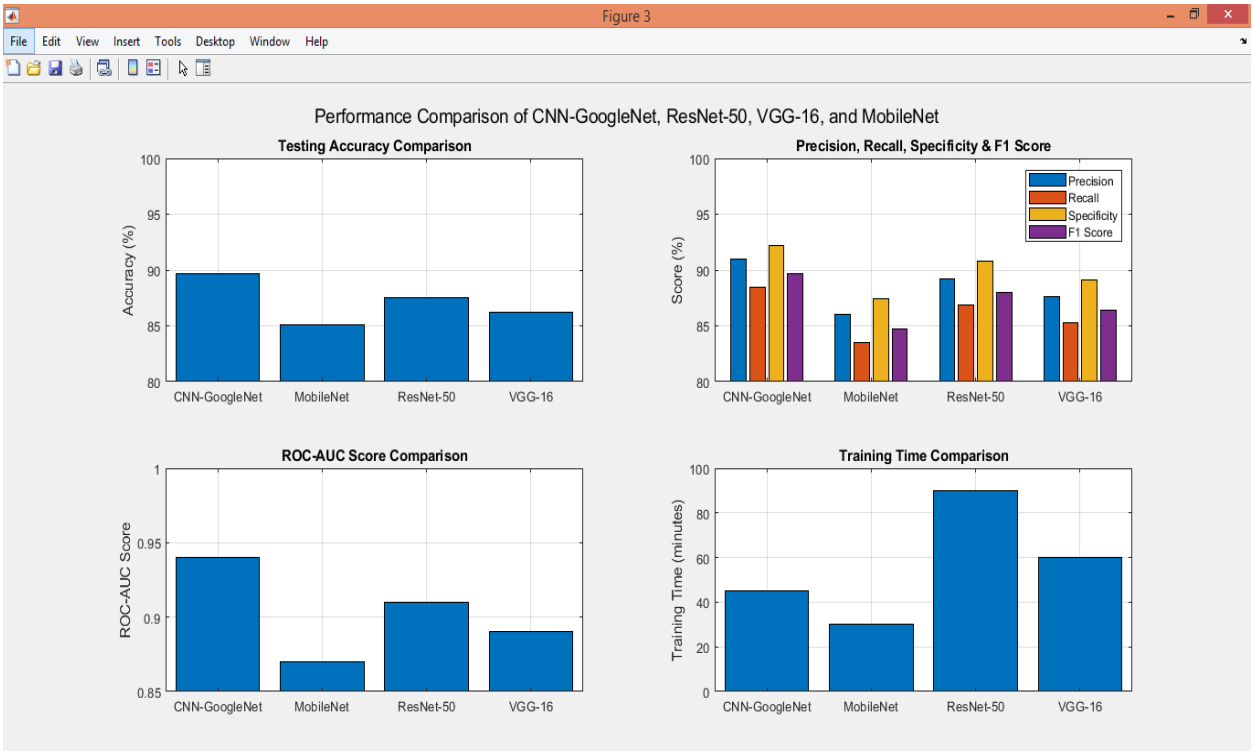

Fig. 9 Graphical Performance analysis of CNN using googlenet in MATLAB and Comparison of proposed model with ResNet50, VGG-16 and Mobilenet for face reorganization system

The CNN-GoogleNet model achieves the best accuracy of 89.70% and has a relatively short training time of 45 minutes compared to ResNet-50 and VGG-16. CNN-GoogleNet demonstrates the highest Precision and Specificity, indicating fewer false positive results. While ResNet-50 is accurate, it has a higher computational complexity and requires a longer training period of 1.5 hours. VGG-16 shows signs of overfitting, as evidenced by its lower testing accuracy despite good performance during training.

6. Conclusion

The process of implementing a Convolutional Neural Network (CNN) for face recognition in MATLAB includes crucial stages such as data pre-processing, specifying the CNN architecture, training the model, and assessing its performance. By adhering to these stages, we attained a satisfactory result. The confusion matrix gave a simple overview of how well the model performed, outlining its strong points and indicating where it could be improved.

The frequently vital metrics that helped measure the model performance include training accuracy, validation accuracy, testing accuracy, precision, recall, F1 score , and ROC-AUC score. The confusion matrix provided additional information on the numbers of true positives, true negatives, false positives, and false negatives, giving a complete view of the model's performance. Training Accuracy: confirming its effectiveness in practical applications with an accuracy of 89.70%. Precision: was 91.0% with true positive cases. Recall (sensitivity): The model achieved a recall of 88.5%, indicating that it correctly identified 88.5% of the true positive cases. Specificity: The specificity reached a value of 92.2%, meaning that the model correctly identified 92.2% of the true negative samples. F1 Score: The F1 score, being the harmonic mean of precision and recall, was measured to be 89.7%, indicating a balanced predictive performance. The area under the Receiver Operating Characteristic curve was 0.94, reflecting a strong ability to distinguish between positive and negative categories. The overall duration of training for the CNN was 45 minutes. Number of Epochs: The model underwent training for a total of 6 epochs. Batch Size: A batch size of 32 samples was utilized during training. The initial learning rate set was 0.01. Cross-Entropy Loss was the chosen loss function for this model. Optimizer: Stochastic Gradient Descent with Momentum (SGDM) was employed as the optimization method.

6.1. Future Scope

1. Deeper Architectures: To enhance the accuracy and robustness of the proposed system, deeper CNN architectures like ResNet, Inception, and DenseNet can be developed.
2. To create a further accurate and secure system, a dimensionality reduction technique can be used.
3. A more accurate, efficient system can be developed with the help of Cross-domain adaptation. In this technique, a CNN model trained with a dataset can also perform well on a diverse dataset without extensive retraining.

References

[1] Zhong Xiaolei, "Improving Face Recognition Accuracy through Optimization of Haar and LBP Features in MATLAB," *Scientific Journal of Technology*, vol. 6, no. 5, pp. 119-125, 2024. [CrossRef] [Publisher Link]

[2] E.L. Arjun et al., "Advanced Face Authentication Using Deep Learning Models," *IEEE Pune Section International Conference*, Pune, India, pp. 1-6, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[3] Mallika Kohli et al., "Identification and Recognition of Facial Images," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 6, no. 5, pp. 69-76, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[4] Medha Jha et al., "Face Recognition: Recent Advancements and Research Challenges," *13th International Conference on Computing Communication and Networking Technologies*, Kharagpur, India, pp. 1-6, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[5] Shilpa Sharma, and Kumud Sachdeva, "Face Recognition using PCA and SVM with Surf Technique," *International Journal of Computer Applications*, vol. 129, no. 4, pp. 41-46, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[6] Priyanka Dhoke, and M.P. Parsai, "A MATLAB based Face Recognition using PCA with Back Propagation Neural Network," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, no. 8, pp. 5291-5297, 2014. [Google Scholar] [Publisher Link]

[7] Nawaf Hazim Barnouti, "Face Recognition using PCA-BPNN with DCT Implemented on Face94 and Grimace Databases," *International Journal of Computer Applications*, vol. 142, no. 6, pp. 8-13, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[8] Zhiming Qian, and Dan Xu, "Research Advances in Face Recognition," *Chinese Conference on Pattern Recognition*, Nanjing, China, pp. 1-5, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[9] C.R. Vishwanatha et al., "Face Recognition and Identification Using Deep Learning," *Third International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies*, Bhilai, India, pp. 1-5, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[10] Al Mahmud Zayeef, and Rana Jyoti Chakma, "Face Recognition-Based Automated Attendance System for Educational Institutions Utilizing Machine Learning," *Information and Communication Technology for Competitive Strategies*, pp. 325-333, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[11] Shahrin Azuan Nazeer, Nazaruddin Omar, and Marzuki Khalid, "Face Recognition System using Artificial Neural Networks Approach," *International Conference on Signal Processing, Communications and Networking*, Chennai, India, pp. 420-425, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[12] Jingu Heo, M. Savvides, and B.V.K. Vijayakumar, "Performance Evaluation of Face Recognition using Visual and Thermal Imagery with Advanced Correlation Filters," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, San Diego, CA, USA, pp. 9-9, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[13] Musab Coşkun et al., "Face Recognition with Convolutional Neural Network," *International Conference on Modern Electrical and Energy Systems*, Kremenchuk, Ukraine, pp. 376-379, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[14] Lee Hui Kueh, and Lee Jon-Tark, "Face Recognition Using Linear Discriminant Analysis (LDA) of Principal Component Analysis (PCA)," *Proceedings of the 8th Symposium on Advanced Intelligent Systems*, pp. 941-944, 2007. [Google Scholar]

[15] Yaniv Taigman et al., "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701-1708, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[16] Florian Schroff, Dmitry Kalenichenko, and James Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815-823, 2015. [Google Scholar] [Publisher Link]

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2016. [Google Scholar] [Publisher Link]

[18] Kaiming He et al., "Deep Residual Learning for Image Recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[19] Priyansh Saxena, Akshat Maheshwari, and Saumil Maheshwari, "Predictive Modeling of Brain Tumor: A Deep Learning Approach," *arXiv preprint*, pp. 1-5, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[20] Karen Simonyan, and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv Preprint*, pp. 1-14, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[21] Wisal Hashim Abdulsalam et al., "Automated Glaucoma Detection Techniques: A Literature Review," *Engineering, Technology & Applied Science Research*, vol. 15, no. 1, pp. 19891-19897, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[22] Andrew G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv Preprint*, pp. 1-9, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[23] Prasun Roy et al., "Effects of Degradations on Deep Neural Network Architectures," *arXiv Preprint*, pp. 1-10, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[24] Douglas O'Shaughnessy, "Recognition and Processing of Speech Signals Using Neural Networks," *Circuits, Systems, and Signal Processing*, vol. 38, no. 8, pp. 3454-3481, 2018. [CrossRef] [Google Scholar] [Publisher Link]