LIVENESS VERIFICATION FOR FACE RECOGNITION SYSTEM

Sarvesh Chaudhari[1], Abhirav Gote[2], Faizan Kadri[3], Samruddhi Gadakh[4], Prof. A. A. Kotalwar[5]

[1, 2, 3, 4] *Students, Department of Computer Engineering, Sinhgad College of Engineering, Pune, Maharashtra, India*
[5]*Assistant Professor, Department of Computer Engineering, Sinhgad College of Engineering, Pune, Maharashtra, India*

ABSTRACT

The popularity of face recognition is increasing due to its ability to recognize human faces, making it a popular topic for computer vision research. Despite its usefulness in image analysis and understanding, face recognition systems face challenges with fraudulent identities, commonly known as spoofing attacks. To address this challenge, face liveness detection is an important biometric authentication method that distinguishes between authentic and fake faces. Liveness detection can be implemented using software or hardware means. However, the use of additional hardware can be costly. The project aims to develop a liveness verification system for face recognition that can accurately distinguish between live and spoofed faces. Spoofing attacks on face recognition systems are becoming increasingly common, and therefore, it is necessary to develop robust liveness verification techniques. The proposed system employs a combination of physiological and behavioral biometric modalities to detect fake faces, including eye blinking, head movement, and facial expressions. The system also utilizes machine learning algorithms to analyze and classify live and spoofed faces based on various features extracted from the biometric data. The effectiveness of the proposed system is evaluated using a large dataset of live and spoofed faces, and the results show a high accuracy in detecting fake faces. The developed liveness verification system has the potential to improve the security of face recognition systems and prevent unauthorized access to sensitive information. Therefore, this paper proposes a CNN-based model for liveness verification and a web-based application to demonstrate its real-time implementation.

Keywords : - *Face Recognition, Computer Vision, Image Analysis, Spoofing Attack, Face Detection, Liveness Verification, CNN*

1. INTRODUCTION:

Face recognition has become the second most widely used biometric authentication method globally, after fingerprints. Biometrics involves measuring and mapping specific biological traits, such as fingerprints, face, and palm veins, for use as an individualized recognition code. Face recognition identifies individuals from digital images or video frames and has rapidly spread across various fields, including mobile device authentication, payments, attendance systems, forensics, and security access. However, face recognition systems are vulnerable to spoofing attacks, as images or videos of a person can be easily obtained from social media or the internet, and pictures can be taken from a distance. Nonetheless, several solutions exist to overcome such security threats.

Liveness verification is a crucial aspect of face recognition systems that aims to ensure the authenticity of the person being identified. It involves detecting whether the face presented in front of the system is a live, physical face or a static image, videos. Liveness detection is an essential part of modern biometric authentication systems, as it prevents fraudulent activities such as identity theft, spoofing attacks, and impersonation. In this project, we will be developing a liveness verification module for a face recognition system. The module will use advanced computer vision techniques and machine learning algorithms to distinguish between live faces and fake ones. Our goal is to build a robust and accurate system that can provide an additional layer of security to face recognition systems, making them more secure and reliable.

2. RELATED WORK:

Tejashree Dhawle, Urvashi Ukey, Rakshandha Choudante[1] describes “Face Detection and Recognition using OpenCV and Python”. In this paper according to author Face detection is basically an image segmentation problem as the image is to be segmented into two parts: one containing faces and the other containing non-face regions. Face detection takes images/video as input and locates face within these images. This is done by separating face areas from non-face background regions. Facial feature extraction locates important feature (eyes, mouth, nose and eyebrows) positions within a detected face.

Lih-Heng Chan, Sh-Hussain Salleh and Chee-Ming Ting[2] presented a facial biometric method that utilizes Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). PCA is used for dimensionality reduction by projecting the original face into a lower-dimensional space. This method preserves image variation while reducing image dimensionality, making it widely used for face images. It transforms image data into a feature space of known images, where the features are the principal components of the set of faces. It identifies an individual face by a weighted sum of the Eigenface features. Therefore, to identify an individual face, one only needs to compare these weights to those of individuals. The Eigen object recognizer class applies PCA to each image, resulting in Eigen Values.

Gahvun Kim[3] proposed a method for detecting fake faces that uses frequency and texture analysis. Extracting frequency information from a given image can involve several steps. Firstly, the facial image is transformed into the frequency domain using discrete Fourier transform. The resulting Fourier-transformed image is shifted so that the zero-frequency component is centered in the spectrum. The image is then divided into several categories in the form of concentric rings, with the difference of two neighboring rings is set to 1. From an image, a set of 32 concentric rings is generated, where each ring represents a corresponding region. Rings with smaller radii contain lower frequency information of the image.

Hyung-Keun Jee, Sung-Uk Jung, and Jang-Hee Yoo[4] proposed a strategy for embedded face recognition systems that is based on analyzing eye movement. The main assumption is that due to the flickering and uncontrolled movements of the pupils in the eyes, there should be significant shape variations. Firstly, the center point of both eyes is identified in the input face image. Using the identification of both eyes, the face region is normalized, and the eye regions are extracted. After extracting the eye regions, each binarized eye region is compared and variations are noted. If the result exceeds the threshold, the input image is recognized as a live face; otherwise, it is rejected as a photograph.

Raden Budiarto Hadiprakoso, hermawan Setiawan, Girinoto[5] describes “Face Anti- Spoofing Using CNN Classifier and Face Liveness Detection”. In this paper according to author the widespread use of facial recognition biometric authentication has highlighted the necessity for face anti-spoofing measures. This study advocates for using a CNN analysis model for image input and combining it with a face liveness detection module to enhance security. The testing of this module resulted in outstanding outcomes for preventing multiple types of face spoof attacks. Our experimentation involved testing various spoof face attacks, including static attacks such as masks, photo posters, and digital photos, and dynamic attacks such as video replays. Additional studies can investigate parallel programming methods that can expedite the facial recognition program's processing time.

Anu Priya, Dr. Hardayal Singh Shekhwat, Mr. Maninder Singh Nehra[6] describes “KNN Classification for the Face Spoof Detection”. According to author technique for detecting spoofed faces, which are added due to unauthorized access to data. Additionally, it uses Discrete Wavelet Transform (DWT) to identify textual characteristics from input data. The classification of spoofed and non-spoofed faces is done using traditional methods such as SVM classifiers. The study utilizes the KNN classifier for classification performance and analyzes the accuracy and execution time. The results show an improvement in accuracy and a decrease in execution time by using this proposed novel approach

3. ALGORITHM:

3.1 Convolutional Neural Network:

Convolutional Neural Network or CNN, is a type of artificial neural network that is widely used in computer vision and image recognition applications. It is designed to process data with a grid-like topology, such as images, and consists of multiple layers of neurons that apply mathematical operations to the input data. The first layer usually performs a convolution operation to extract features from the input image, while subsequent layers may apply pooling, normalization, and activation functions to further process the features. CNN is a powerful and versatile deep learning technique that has revolutionized the field of computer vision. CNN typically consist of several types of layers mainly Convolutional Layer, Pooling Layer, Activation layer and Fully Connected Layer.

3.1.1 Convolutional Layer:

A Convolutional Layer is a key component of Convolutional Neural Networks (CNNs) used in computer vision and image processing applications. It applies a set of learnable filters to the input image to extract features such as edges, shapes, and textures. The convolutional layer works by sliding the filters over the input image, computing the dot product between the filter and the input pixel values at each position. This operation generates a feature map for each filter. Multiple filters can be applied to the input image to produce multiple feature maps, which capture different aspects of the input image. The size, number, and stride of the filters can be adjusted depending on the specific problem and the desired output. The weights of the filters are learned through the backpropagation algorithm during training, which adjusts the weights to minimize the difference between the predicted output and the actual output. In summary, the convolutional layer plays a critical role in extracting relevant features from the input image, which can then be processed by subsequent layers to perform tasks such as object recognition and classification.

3.1.2 Pooling Layer:

A Pooling Layer is a common layer in Convolutional Neural Networks (CNNs) used in computer vision and image processing applications. It is typically inserted after a Convolutional Layer and is used to reduce the spatial dimensionality of the output feature maps. The pooling function outputs a single value that summarizes the information in that subregion. By reducing the spatial dimensionality of the feature maps, pooling helps to make the output of the convolutional layer more robust to small variations in the input image, while also reducing the number of parameters in the network, which can help prevent overfitting. There are several types of pooling functions, including max pooling, average pooling, and L2-norm pooling, each of which has different characteristics and may be more appropriate for different types of data or tasks. In summary, the pooling layer is a useful layer in CNNs that helps to reduce the spatial dimensionality of the output feature maps, making the network more robust and less prone to overfitting.

3.1.1 Convolutional Neural Network:

A Fully Connected Layer is a layer in neural networks that connects every neuron in one layer to every neuron in the next layer. In Convolutional Neural Networks (CNNs), the Fully Connected Layer is typically located at the end of the network and is used to perform a classification or regression task based on the features extracted from the input image by the previous layers. The Fully Connected Layer takes the flattened output of the previous layer, which represents a high-dimensional feature vector, and applies a set of learnable weights to perform a linear transformation of the input. The output of the Fully Connected Layer is then passed through an activation function, such as softmax or sigmoid, to produce a probability distribution over the output classes. The Fully Connected Layer is an important component of CNNs as it allows the network to learn complex relationships between the input features and the output classes, and to perform accurate classification or regression tasks. The number of neurons in the Fully Connected Layer can be adjusted depending on the complexity of the problem and the size of the input feature vector. In summary, the Fully Connected Layer is a key component of CNNs that performs a classification or regression task based on the features extracted from the input image by the previous layers. It allows the network to learn complex relationships between the input and output and to produce accurate predictions.

4. IMPLEMENTATION:

In our project there is basically three main modules are there namely Face Detection, Face Recognition and Liveness Verification.

4.1.1 Face Detection:

Facial detection relies on Histogram of Oriented Gradients (HOG) technology. The input image is first converted into black and white as color data is not necessary. The pixels are then analyzed and the surrounding pixels are taken into consideration to create an arrow or line based on the gradient shift from light to dark. This process is repeated for each pixel in the image to create a plot of the flow of light known as Gradients. By only considering the direction of gradient shift, an image of the same person, whether light or dark, will be represented in the same way. To reduce the amount of detail and save space, the higher level of gradient is considered to generate a forest of trees. The image is divided into 16x16 pixel squares and in each square, the gradients are counted to determine the major direction of arrows, which are used to create one large arrow for the square. This produces a HOG representation that can be compared to known HOG patterns to detect faces.

4.1.2 F ace Recognition:

The Deep Convolutional Neural Network (CNN) is responsible for the facial recognition process after detection. It extracts relevant features from images to generate 128 measurements for each face. During training, the CNN is fed three face images:

- a training image of the known person,
- another picture of the same person,
- and an image of a different person's face.

The algorithm generates measurements for all three images and adjusts the neural network so that the measurements for images #1 and #2 are closer, while the measurements for #2 and #3 are further apart. This process is repeated millions of times on millions of images to teach the neural network to generate 128 measurements on faces of different people. The Dib library is trained on millions of faces, and the facial recognition module is built upon this neural network to provide accurate measurements in an efficient manner. The process of generating 128 measurements is referred to as Embedding.

4.1.3 Liveness Verification:

To verify liveness, we will utilize the CNN model, which consists of various parts such as the input layer, convolutional layer, pooling layer, full connection layer, and output layer. The convolution layer extracts image features, while the pooling layer decreases the number of parameters and the full connection layer outputs results. Typically, a CNN contains multiple convolution and pooling layers.

The architecture for face detection and liveness verification typically involves multiple components working together to detect and verify a live human face. Here is an overview of a typical architecture for a face detection and liveness verification system:

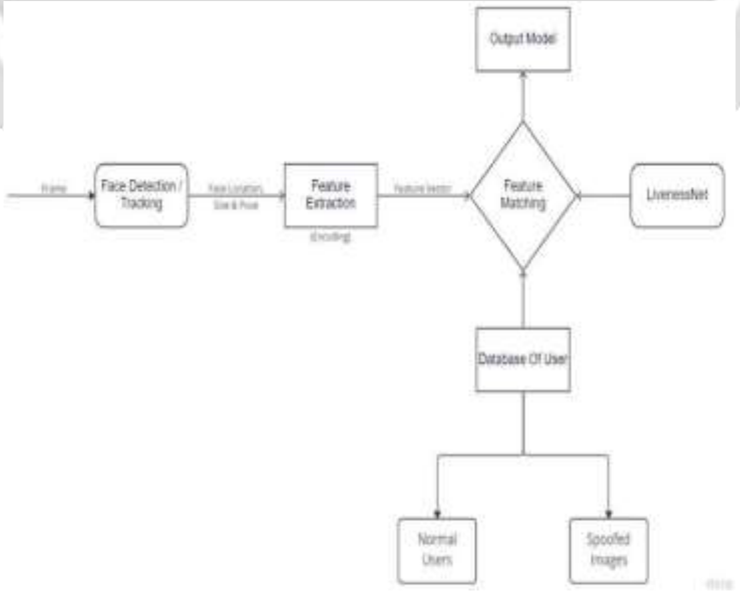

Fig 1. Face Detection and Liveness Verification Architecture

1. Image Capturing: The first step is to acquire an image or video of the user's face. This can be done using a camera or webcam.
2. Face detection: The system uses computer vision algorithms to detect the face in the image or video. This involves identifying the location and boundaries of the face within the image or video.
3. Face tracking: Once the face is detected, the system tracks the face in real-time. This ensures that the user's face is always in view, and any movement or changes in facial expressions can be captured.
4. Liveness detection: The system performs liveness detection, which involves analyzing various features of the face to determine if it is a live human face or a fake one. This can be done using various techniques such as analyzing eye movement, blink detection, head movement, or skin texture analysis.
5. Face recognition: Once liveness is confirmed, the system then compares the live face to the user's stored facial template to verify their identity. If the face is verified, the user is granted access to the system.

The architecture can be divided into two main parts: the front-end and the back-end. The front-end is responsible for capturing the live video or image of the user's face and performing face detection, tracking, and liveness detection. This is typically done using computer vision libraries and tools such as OpenCV or TensorFlow.

The back-end is responsible for storing the user's facial template, performing face recognition, and granting access to the system if the user is verified. This can be done using machine learning algorithms such as deep neural networks, which can be trained to recognize faces and identify individuals based on their unique facial features.
Overall, a robust face detection and liveness verification system requires a well-designed

4.2 Program Flow:

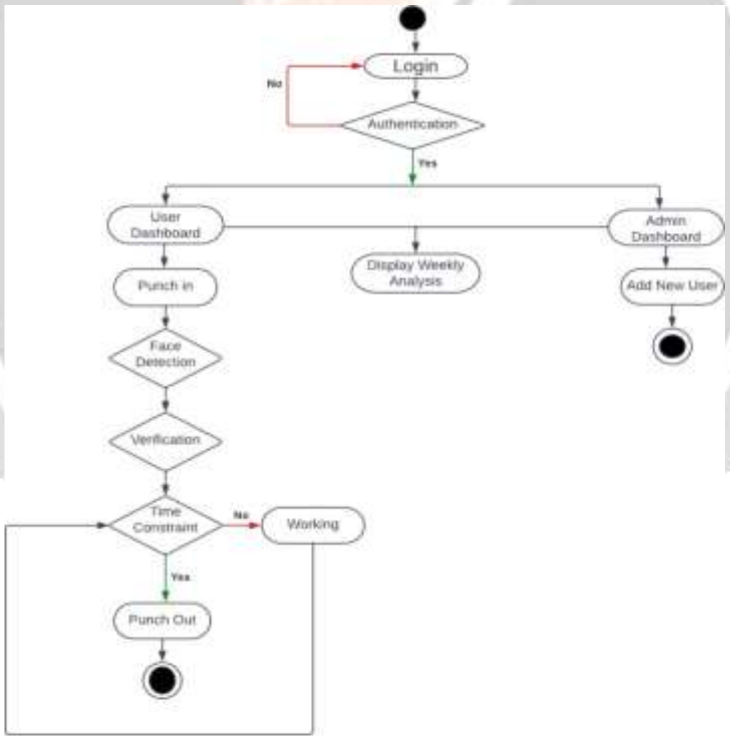

Fig 2. Activity Flow Diagram

4.2.1 F ace Detection:

- User will login through its proper credentials.
- The application will ask for camera access to detect his face.
- After the application has properly detected the user's face, further steps would beundertaken.

4.2.2 Face Recognition:

- The application will search its database for the matching face.
- Application will verify if the inputted face matches the one in the database.
- Simultaneously it will check for liveliness.

4.2.3 Liveness Verification

- Application will analyze the light effects on the face.
- Application will verify the liveliness based on light effects on the face.
- The Application will confirm that the user is live and not recorded.

4.2.4 Punch-in and Punch-out time:

- When the application has properly recognized the user's face and verified his liveliness, it will store and print the time of verification i.e., the punch-in time
- Punch-out button will be frozen until a specific required time.
- Users can not log out until the specified time.
- After the conditions for timestamp are satisfied, the punch-out button will be then active.
- Users can log out using this button.
- The timestamp of punch-out will be recorded and used for further analysis.

4.2.5 Weekly Chart:

- The punch-in and punch-out time of every employee for every day will be recorded.
- This stored data will be plotted into a bar graph for every week.
- After user authentication, each employee can see their weekly analysis chart on the dashboard

4.3 System Overview:

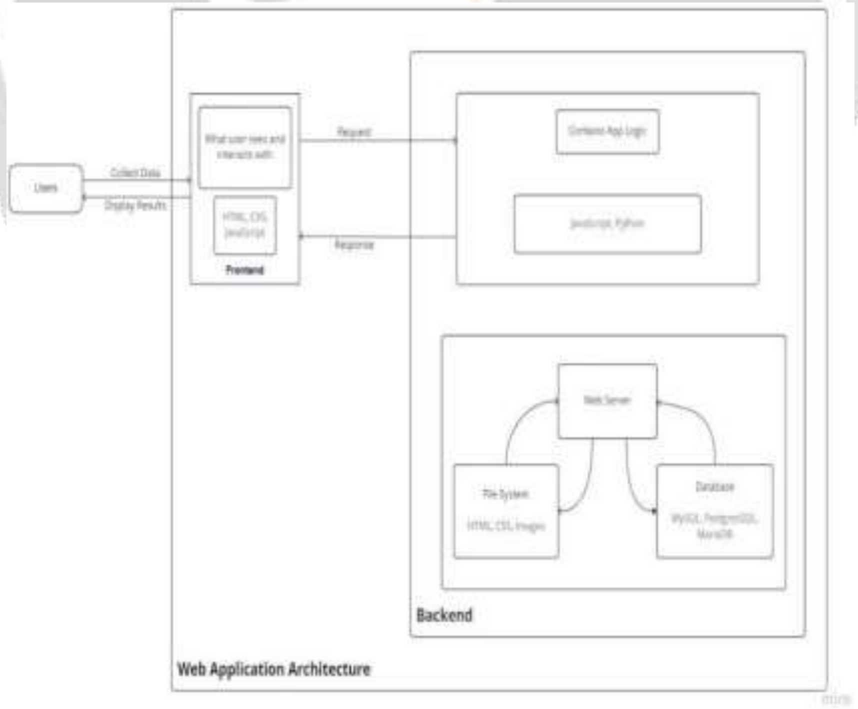

Fig 3. Web Application Architecture

5.CONCLUSION AND FUTURE SCOPE:

Face recognition systems have become popular among top tech companies and industries due to their ease of use and accessibility through the use of Python programming and OpenCV. This project proposes a user-friendly and cost-efficient face recognition system that can be customized for various purposes. To prevent face spoofing, many security systems have implemented face liveness detection. The proposed system uses a CNN approach, which achieved an acceptable testing accuracy of approximately 85%. The system is comprised of two stages: feature extraction and classification. However, future work intends to use new techniques, such as capsule neural networks, and compare pre- trained CNN models with regular CNN to achieve higher performance. Additionally, deep learning algorithms, such as RNN, will be used to detect face spoofing in videos, which is an important topic in face spoofing.

In the future, advancements in machine learning and computer vision can further improve the accuracy and speed of liveness verification. The development of more advanced sensors and hardware can also contribute to the effectiveness of liveness verification. Additionally, incorporating multimodal biometric authentication techniques, such as combining facial recognition with voice or fingerprint recognition, can further enhance the security of the system.

Overall, liveness verification is an essential component of any facial recognition system, and continued research and development in this field can contribute to improving the overall security and reliability of these systems.

6. REFERENCES:

[1]. “Face Detection and Recognition using OpenCV and Python”, Tejashree Dhawle, Urvashi Ukey, Rakshandha ChoudanteInternational Research Journal of Engineering and Technology (IRJET), ”, e- ISSN: 2395-0056, Volume: 07 Issue: 10 | Oct 2020

[2]. “Face Biometrics Based on Principal Component Analysis and Linear Discriminant Analysis”, Lih-Heng Chan, Sh-Hussain Salleh and Chee-Ming Ting, Journal of Computer Science 6 (7): 693-699, 2010, ISSN 1549-3636, © 2010 Science Publications

[3].“Face liveness detection based on texture and frequency analyses”, Gahyun Kim, Dong ik Kim, Sungmin Eum (School of Electrical and Electronic Engineering, Yonsei University, Republic of Korea, 978-1-4673-0397-2/12/$31.00 ©2012 IEEE

[4]. “Liveness Detection for Embedded Face Recognition System”, Hyung-Keun Jee, Sung- Uk Jung, and Jang-Hee Yoo , International Journal of Biological and Medical Sciences 1:4 2006

[5]. “Face Anti-Spoofing Using CNN Classifier and Face Liveness Detection”, Raden Budiarto Hadiprakoso, hermawan Setiawan, Girinoto, 2020 3rd International Conference on Information and Communications Technology (ICOIACT) | 978-1-7281-7356-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICOIACT50329.2020.9331977

[6]. “KNN Classification for the Face Spoof Detection”, Anu Priya, Dr. Hardayal Singh Shekhawat, Mr. Maninder Singh Nehra, International Journal of Research in Electronics and Computer Engineering, IJRECE VOL. 7 ISSUE 2 APR.-JUNE 2019 ISSN: 2393-9028, ISSN: 2348-2281

[7]. “Face Liveness Detection Using a sequential CNN technique” 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC) | 978-1- 6654-1490- 6/21/$31.00 ©2021 IEEE | DOI: 10.1109/CCWC51732.2021.9376030

[8]. “Liveness Detection Based on Improved Convolutional Neural Network for Face Recognition Security”, E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 12, Issue 08, August 2022

[9]. “Face Recognition System”, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181 Vol. 8 Issue 05, May-2019

[10]. “Insight on Face Liveness Detection: A Systematic Literature Review”, International Journal of Electrical and Computer Engineering (IJECE), Vol. 9, No. 6, December 2019, pp. 5165~5175, ISSN: 2088-8708, DOI: 10.11591/ijece.v9i6.pp5165-5175

[11]. “Liveness Detection Based on Improved Convolutional Neural Network for Face Recognition Security”, E-ISSN 2250-2459, Scopus Indexed, ISO 9001:2008 Certified Journal, Volume 12, Issue 08, August 2022