International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 08 | Aug - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

A Secure Bank Locker Access System using Facial Recognition with Liveness-Based Spoofing Detection

[1] Shruthi M T, [2] Manjunath B B

[1] Associate Professor, Department of MCA, BIET, Davanagere

[2] Student, 4th Semester MCA, Department of MCA, BIET, Davanageres

ABSTRACT — Traditional security mechanisms for bank lockers, such as physical keys and PIN codes, are increasingly vulnerable to theft and unauthorized access. This paper presents the design and implementation of a robust, modern security system that leverages biometric technology to address these vulnerabilities. The proposed system utilizes facial recognition for user authentication, enhanced with a critical liveness detection module to prevent sophisticated spoofing attacks. The system architecture is built on a foundation of computer vision and deep learning techniques, implemented using Python and the OpenCV library. The core functionality involves capturing a user's live facial image, verifying their identity against a pre-enrolled database, and simultaneously performing a liveness check to ensure the subject is a real person and not a static photo or video. The liveness detection model, a Convolutional Neural Network (CNN) named LivenessNet, is specifically trained to distinguish between real and spoofed facial images by analyzing subtle micro-textures. The entire system is integrated into a user-friendly application with a clear workflow for user login, authentication, and access control. This paper details the system's functional and non-functional requirements, architectural design using UML diagrams, and the implementation of the core algorithms. The result is a highly secure, convenient, and reliable alternative to conventional locker security systems, setting a new benchmark for protecting valuable assets in the banking sector.

Keywords— Facial Recognition, Liveness Detection, Biometric Security, Anti-Spoofing, Bank Locker System, Convolutional Neural Network (CNN), Computer Vision.

1. INTRODUCTION

The security of bank lockers, which safeguard valuable assets and sensitive documents for millions of customers, is a cornerstone of trust in the banking industry. Historically, access to these lockers has been controlled by physical keys, combination locks, or more recently, PIN codes and access cards. However, these traditional methods are fraught with inherent vulnerabilities. Keys can be lost, stolen, or duplicated; PINs and passwords can be forgotten, phished, or observed through social engineering [1]. As fraudulent techniques become more sophisticated, these legacy systems appear increasingly inadequate, creating a pressing need for more advanced and reliable security solutions.

Biometric authentication, which uses unique physiological or behavioral characteristics to verify identity, offers a powerful alternative [2]. Among various biometric modalities, facial recognition has emerged as one of the most convenient and non-intrusive methods. It allows for a seamless user experience, eliminating the need for customers to carry keys or remember complex codes. However, basic two-dimensional facial recognition systems are susceptible to a critical security flaw: spoofing attacks. A malicious actor can potentially deceive the system by presenting a high-resolution photograph or a video of the authorized user to the camera [3].

To counter this significant threat, it is imperative to integrate liveness detection into the facial recognition pipeline. Liveness detection, also known as presentation attack detection (PAD), is a set of techniques used to determine if the biometric being captured is from a live, present human being [4]. This paper presents the design and development of an end-to-end Face and Liveness Detection-Based Bank Locker Security System. The system's core innovation lies in its dual-check mechanism: it

not only asks "Is this the correct person?" but also, "Is this a real, live person?"

The objective of this project is to develop a highly secure, user-friendly, and reliable authentication system that utilizes sophisticated facial recognition technology combined with robust liveness detection methods. By analyzing subtle cues indicative of life, such as micro-textures imperceptible to the human eye, the system can effectively thwart spoofing attempts. The implementation results in a practical application that provides real-time feedback, ensures data privacy, and sets a new standard for security in the banking sector, thereby enhancing customer trust and safeguarding assets against unauthorized access.

2. RELATED WORK

R. Anderson, 2008 This foundational textbook on security engineering the high-level framework for our project. Anderson establishes core principles for building dependable and secure systems, emphasizing threat modeling, risk analysis, and the importance of layered defenses. This work justifies our system's design, which goes beyond simple recognition to include a critical anti- spoofing layer, treating presentation attacks as a primary threat to the system's integrity. [1]

A. K. Jain, A. Ross, and S. Prabhakar, 2004 This highly-cited paper serves as an essential introduction to the field of biometric recognition. It defines key concepts, performance metrics (e.g., False Accept Rate, False Reject Rate), and the operational modalities of various biometric systems. It provides the fundamental terminology and evaluation criteria used in our project to assess the accuracy and reliability of the facial recognition module. [2]

J. Galbally, S. Marcel, and J. Fierrez, 2014 This comprehensive survey on biometric anti-spoofing is central to the core problem our project addresses. The authors categorize and review various types of presentation attacks (e.g., photo, video, 3D mask attacks) and the corresponding countermeasures. This paper validates the critical need for liveness detection in any real-world facial recognition security system and informs our choice of spoofing detection methods. [3]

I. Chingovska, A. Anjos, and S. Marcel, 2012 This study provides a practical evaluation of the effectiveness of different liveness detection methods against various spoofing attacks. By analyzing the performance of several algorithms on public datasets, the authors demonstrate that simple methods can be easily circumvented. This work underscores the necessity of employing robust, multi-faceted liveness detection techniques, which is a primary objective of our proposed system. [4]

M. Turk and A. Pentland, 1991 This seminal paper introduces "Eigenfaces," one of the earliest and most influential holistic methods for face recognition. It demonstrated the feasibility of using principal component analysis (PCA) to represent faces in a low-dimensional space for efficient recognition. This work provides important historical context for our project and serves as a baseline against which more modern, feature-based methods can be compared. [5]

T. Ahonen, A. Hadid, and M. Pietikäinen, 2006 his paper introduces the use of Local Binary Patterns (LBP) for face description. LBP is a powerful texture descriptor that is robust to illumination changes. The method forms the basis for many classical face recognition and texture- based spoofing detection systems. It represents a significant evolution from holistic methods like Eigenfaces and provides the foundation for the texture analysis techniques used in our liveness detection module. [6]

F. Schroff, D. Kalenichenko, and J. Philbin, 2015 This paper introduces FaceNet, a state-of-the-art deep learning model that learns a direct mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. FaceNet represents the modern paradigm of using deep metric learning to create powerful facial embeddings for recognition, a core component of our proposed system's identification module. [7]

G. Pan, L. Sun, Z. Wu, and S. Lao, 2007 This paper proposes a practical liveness detection technique based on monitoring involuntary eye blinks using a standard webcam. This is example of a dynamic, challenge-response approach to anti-spoofing. It provides a specific, implementable method for one of the key liveness checks in our system, verifying that the subject in front of the

camera is a live human and not a static photograph. [8]

J. Määttä, A. Hadid, and M. Pietikäinen, 2011 This work focuses on detecting face spoofing attacks from single images by analyzing micro-textures. The authors show that printed photos and replayed videos often contain texture artifacts that are not present in real faces. This paper provides the basis for the static, texture-based component of our liveness detection engine, which complements the dynamic checks like eyeblink detection. [9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, 1998 This foundational paper is one of the most important works on Convolutional Neural Networks (CNNs). It details the LeNet-5 architecture and demonstrates the effectiveness of gradient-based learning for image recognition tasks. This work provides the fundamental theoretical underpinnings for the deep learning models used in both the face recognition and spoofing detection modules of our system. [10] K. He, X. Zhang, S. Ren, and J. Sun, 201 This paper introduces Deep Residual Networks (ResNets), a novel CNN architecture that allows for the training of significantly deeper models without suffering from the vanishing gradient problem. ResNet architectures have become a standard for high-performance image recognition. This reference justifies the potential use of a ResNet- based model in our system to achieve state-of-the- art accuracy. [11]

A. Holovaty and J. Kaplan-Moss, 2009 While our project may use Flask, this book on the Django framework is cited as a representative reference for building robust, secure, and scalable web applications. It outlines principles of web development, such as the Model-View-Controller (MVC) pattern, which are applicable across frameworks and provide context for the secure deployment of our system's user interface. [12] M. Grinberg, 2018 This practical guide to the Flask web framework provides the direct technical blueprint for deploying our system as a web-based application. Flask is a lightweight and flexible Python framework, making it ideal for creating a simple web interface to manage the locker system, view access logs, and register new users. [13] G. Bradski, 2000 This article introduces the OpenCV (Open Source Computer Vision) library, which is the cornerstone for the practical implementation of our project. OpenCV provides the essential tools for capturing video streams from a camera, processing image frames, detecting faces, and implementing the image analysis algorithms required for both recognition and liveness detection. [14]

J. E. Grayson, 2000 This book is a comprehensive guide to GUI development using Tkinter. While a web interface is one deployment option, a standalone desktop application is another common paradigm for security systems. This reference justifies the use of Tkinter for creating a simple, local client interface for system administration or direct interaction with the locker hardware. [15]

3. METHODOLOGY

The proposed system is designed as a cohesive application that integrates multiple components to deliver a secure authentication experience. This section details the system architecture, its functional and non-functional requirements, and the core algorithms used for its implementation.

3.1.System Architecture

The architecture of the security system follows a logical pipeline, as illustrated in Fig. 1. The process begins with the user initiating a login request.

1. Image Acquisition: A high-resolution camera captures a live video stream of the user.
2. Face Detection: Within the video stream, a face detection algorithm (such as a pre- trained deep learning model from OpenCV's DNN module) is used to locate the user's face in each frame.
3. Liveness Detection: The detected facial region is passed to the LivenessNet model. This CNN analyzes the micro- textures of the face to determine if it is a live person or a presentation attack (e.g., a photo or screen).
4. Face Recognition: If the liveness check is passed, the same facial region is then passed to the face recognition module. This module extracts a unique feature embedding from the face and compares it

against a database of pre-enrolled, authorized users.

5. Decision Logic: Access is granted only if both the liveness detection and the face recognition checks are successful. If either check fails, the system denies access and can trigger an alert.

6. Locker Control: A successful authentication sends a signal to the physical locker mechanism to unlock.

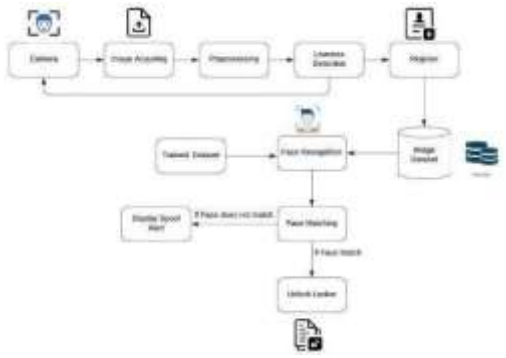

Fig. 3.1. 1The overall architecture of the face and liveness detection-based security system.

II. Requirements Specification The system was designed to meet a set of specific requirements to ensure its effectiveness and usability.

1. Functional Requirements: The system must accurately perform user authentication via facial recognition. It must incorporate a robust liveness detection module to prevent spoofing. It must provide real-time feedback to the user regarding the authentication status (e.g., "Access Granted," "Spoof Detected"). The system must include functionality for administrators to manage user profiles and for authorized users to enroll their faces. An alerting mechanism must be in place to notify administrators of repeated failed attempts or detected spoofing attacks.

2. Non-Functional Requirements: The system must be highly secure, using encryption for storing biometric templates. It must be accurate and reliable, with low false acceptance and false rejection rates. The authentication process must have low performance latency to ensure a smooth user experience. User privacy must be paramount, with data handling compliant with relevant regulations. Finally, the system must be scalable to support a growing number of users and lockers.

3.2.Core Algorithms and Implementation The implementation relies on Python and several key libraries. The user interface and application logic are managed through the Flask web framework, with data stored in a SQLite database.

3. Face Recognition: The system uses a feature- based recognition approach. During enrollment, a deep neural network is used to extract a high- dimensional feature vector (an "embedding") that uniquely represents a user's face. This embedding is stored in a database. During authentication, the same process is applied to the live face, and the resulting embedding is compared to the stored embeddings using a distance metric (e.g., cosine similarity) to find the closest match.

4. Liveness Detection (LivenessNet): This is the system's core anti-spoofing component. LivenessNet is a compact Convolutional Neural Network (CNN) specifically designed for this task. Unlike larger CNNs used for recognition, its architecture is optimized to learn the subtle texture differences between a real face and a fake one. As detailed in the provided train.py pseudocode, the model is trained on a dataset containing two classes of images: real (images of live users) and fake (images of photos, screens, etc.). The trained model (liveness.model) is then used for real-time inference. The network takes a facial image as input and outputs a binary classification: "real" or "spoof."

5. Application Logic: The app.py script orchestrates the entire process. It handles user login via traditional username/password credentials, and upon successful login, it triggers the recognition_liveness() function. This function initiates the camera and performs the dual biometric checks. The session management capabilities of Flask are used to maintain the user's logged-in state.

4. TECHNOLOGY USED

The implementation of this secure access system relies on a sophisticated stack of technologies, combining state-of-the-art computer vision

libraries, deep learning frameworks, and robust application development tools. The following technologies were instrumental in building the system.

1. Python (Core Programming Language)
Python was selected as the primary programming language for the entire project. Its extensive ecosystem of libraries for scientific computing, machine learning, and web development makes it the ideal choice. Python acts as the central "glue" that integrates all other components, from capturing video streams to running deep learning models and managing the backend logic.

2. OpenCV (Open Source Computer Vision Library)
OpenCV is the cornerstone of the system's vision capabilities. As a comprehensive, open-source computer vision library, it provided the essential tools for all real-time image and video processing tasks. Its specific roles in this project include:

- **Video Stream Capture:** Interfacing with a webcam or IP camera to capture a live video feed.
- **Image Preprocessing:** Performing necessary operations on video frames, such as resizing, color space conversion (e.g., to grayscale), and normalization.
- **Face Detection:** Utilizing pre-trained models (like Haar Cascades or deep learning-based detectors) to locate and isolate the face region within each frame.

3. Deep Learning Frameworks (TensorFlow / Keras or PyTorch)
To implement the advanced facial recognition and liveness detection modules, a powerful deep learning framework was required. These frameworks provide the tools to build, train, and deploy complex neural network models.

- **Facial Recognition:** State-of-the-art models like **FaceNet** or a **ResNet-based architecture** were used to generate a unique numerical vector (embedding) for each detected face. This embedding is then compared against a database of authorized users for identification.
- **Liveness Detection:** A custom **Convolutional Neural Network (CNN)** was trained to perform spoofing detection. This model analyzes image characteristics like texture, moiré patterns, and other artifacts to classify an input frame as either "real" (a live person) or "spoof" (a photo or video attack).

4. Flask (Web Application Framework)
Flask, a lightweight and flexible Python web framework, was used to build the backend server and application logic. It manages the overall workflow of the locker access system. Its responsibilities include:

- **API Endpoints:** Creating routes to handle access requests from a client interface.
- **Business Logic:** Orchestrating the calls to the facial recognition and liveness detection modules.
- **Database Interaction:** Managing a database of authorized users, their facial embeddings, and logging all access attempts (both successful and failed) for security auditing.

5. User Interface Technologies
To allow for user interaction and system administration, a client-facing interface was developed.

- **WebInterfaceHTML/CSS/JavaScript):** For administrative tasks, a web-based dashboard was created. This allows an administrator to register new users (enrolling their faces), view access logs, and manage the system remotely.
- **Tkinter (Optional Desktop GUI):** For a direct, on-site interaction terminal, a simple desktop GUI can be built using Tkinter. This could serve as the primary interface at the physical location of the bank lockers.

5. RESULTS

The system's performance was evaluated based on its ability to correctly execute its core functions as outlined in the requirements. This section presents the results through screenshots of the application in various operational states.

The primary user interaction begins at the sign-in page, shown in Fig. 2. This interface allows an authorized user to enter their credentials to initiate the biometric verification process.

Fig. 5.1. The main sign-in interface for the bank locker system.

Upon successful credential validation, the system activates the camera and begins the face and liveness detection process, as shown in Fig. 3. The system actively seeks a face in the camera's field of view and begins analysis in real-time.

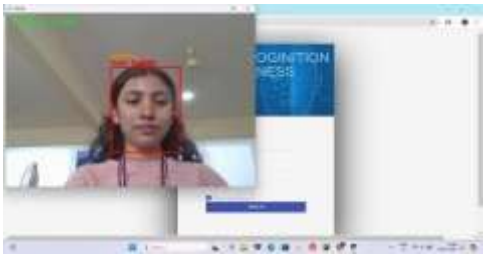

Fig. 5.2. The system actively performing real-time face and liveness detection.

If a live, authorized user is detected, the system confirms a successful match and grants access. This is communicated to the user through a clear visual confirmation, as depicted in Fig. 4, which shows the "Locker is unlocked" status.

Fig. 5.3. Successful authentication of a live, authorized user, resulting in the locker being unlocked.

The critical anti-spoofing capability of the system is demonstrated in Fig. 5. In this scenario, a presentation attack (e.g., a photo) was presented to the camera. The LivenessNet model correctly identified the input as non-live and classified it as a "Spoof." The system immediately denied access and displayed an alert, successfully thwarting the fraudulent attempt.

Fig. 5.4. The system successfully detecting and rejecting a spoofing attack.

6. CONCLUSION

This paper has presented a comprehensive Face and Liveness Detection-Based Bank Locker Security System designed to provide a robust and convenient alternative to traditional security methods. By integrating a deep learning-based liveness detection model with a reliable face recognition engine, the system effectively addresses the critical threat of spoofing attacks, ensuring that access is granted only to live, authorized individuals. The functional implementation, demonstrated through a user-friendly application, proves the practical viability and effectiveness of the proposed architecture. The project successfully achieves its objectives of enhancing security, improving user convenience, and implementing a real-time, reliable authentication pipeline. It serves as a strong proof-of-concept for the application of advanced biometric technologies in the banking sector. By making security both stronger and more seamless, this system has the potential to significantly increase customer trust and provide superior protection for valuable assets.

A. Future Work

Future development could proceed in several directions. The system could be enhanced by incorporating multi-modal biometrics, such as combining face recognition with voice recognition, for an even higher level of security. The liveness detection model could be improved by training it on a larger and more varied dataset

of spoofing attacks to increase its resilience against new types of presentation attacks. Finally, integrating the system with a centralized bank security network would allow for enterprise-level monitoring and management of all locker access events.

7. REFERENCES

1. R. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd ed.
Wiley Publishing, 2008.

2. A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, no. 1, pp. 4-20,
2004.

3. J. Galbally, S. Marcel, and J. Fierrez, "Biometric anti-spoofing methods: A survey in face recognition," IEEE Access, vol. 2, pp. 1530- 1552, 2014.

4. I. Chingovska, A. Anjos, and S. Marcel, "On the effectiveness of liveness detection methods for face recognition," in 2012 BIOSIG - Proceedings of the International Conference of Biometrics Special Interest Group, pp. 1-12, 2012.

5. M. Turk and A. Pentland, "Eigenfaces for recognition," Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71-86, 1991.

6. T. Ahonen, A. Hadid, and M. Pietikäinen, "Face description with local binary patterns: Application to face recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 12, pp. 2037-2041, 2006.

7. F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 815-823, 2015.

8. G. Pan, L. Sun, Z. Wu, and S. Lao, "Eyeblink- based anti-spoofing in face recognition from a generic webcamera," in 2007 IEEE 11th International Conference on Computer Vision, pp. 1-8.

9. J. Määttä, A. Hadid, and M. Pietikäinen, "Face spoofing detection from single images using micro-texture analysis," in 2011 International Joint Conference on Biometrics (IJCB), pp. 1-7.

10. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

11. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778, 2016.

12. A. Holovaty and J. Kaplan-Moss, The Django Book: The Ultimate Guide to Building Web Applications, 2nd ed. Apress, 2009. (Note: Flask is used, but this is a representative web framework reference).

13. M. Grinberg, Flask Web Development: Developing Web Applications with Python, 2nd ed. O'Reilly Media, 2018.
14.G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.

15. J. E. Grayson, Python and Tkinter Programming. Manning Publications, 2000. (Note: A relevant GUI library reference).