Credit Card Fraud Detection Using Image Processing System

Pratiksha Pethe
SVERI's College of Engineering, Pandharpur, Punyashlok Ahilyadevi Holkar Solapur University, Maharashtra, India

***Abstract*—The rapid growth of online banking and digital transactions has increased the risk of credit card fraud. This project proposes a secure and intelligent Credit Card Fraud Detection (CCFD) system using facial recognition technology for user authentication. During the registration phase, users input their account number and passcode while their facial image is captured and stored securely. For every withdrawal attempt, the system captures a live image and verifies it against the registered image using computer vision techniques. If the images match, the transaction is approved; otherwise, it is flagged as unauthorized. This approach enhances security by integrating biometric verification, thereby minimizing fraudulent access and improving trust in digital financial systems.**

1. Guest Users – Browse basic features without registration.

2. Registered Users – Access exclusive content and schedules.

***Index Terms*—Credit Card Fraud Detection, Facial Recognition, Machine Learning, Computer Vision, Cybersecurity, User Verification, Secure Transactions**

I. INTRODUCTION

1.1 Problem Statement: With the increasing reliance on online banking and digital transactions, credit card fraud has become a widespread and growing threat. Traditional authentication methods, such as PINs and passwords, are vulnerable to theft, phishing, and misuse, making it easier for unauthorized individuals to access users' financial accounts. There is a critical need for a more secure and reliable method to verify user identity and prevent fraudulent transactions.

1.2 Motivation: The motivation behind this project stems from the growing number of fraud cases due to weak or compromised authentication systems. Biometric authentication, particularly facial recognition, offers a non-intrusive and user-friendly approach to verifying identity. Integrating facial recognition with traditional login methods enhances security by ensuring that only the authorized user can access and perform transactions. This project aims to bridge the gap between convenience and security in digital banking.

1.3 Proposed System: The proposed system is a Credit Card Fraud Detection (CCFD) model that combines account credentials with facial recognition technology to verify user identity during registration and transaction processes. Users initially register by providing their account number, passcode, and facial image. During a transaction, the system captures a real-time image and compares it with the registered image using computer vision techniques. If the images match, the transaction is approved; otherwise, it is flagged as unauthorized. This dual-layered verification enhances the security of credit card transactions and minimizes the risk of fraud.

II. REVIEW OF LITERATURE

1. Title: Credit Card Fraud Detection with Artificial Neural Networks
Author: Bhattacharyya et al. (2011)
Approach: Supervised machine learning using neural networks and historical data
2. Title: FaceNet: A Unified Embedding for Face Recognition and Clustering
Author: Schroff, Kalenichenko, and Philbin (2015)
Approach: Deep learning-based facial recognition using triplet loss for face comparison
3. Title: Multi-Biometric Systems: An Overview
Author: Ross and Jain (2004)
Approach: Integration of multiple biometrics (e.g., face and password) for improved accuracy

4. Title: A Survey of Credit Card Fraud Detection Techniques
Author: Srivastava et al. (2008)
Approach: Comparative analysis of anomaly detection techniques for transaction fraud

III. BACKGROUND AND RELATED WORK

- Credit card fraud is a major concern in the digital economy, often caused by stolen credentials or identity theft.
- Traditional fraud detection systems use machine learning and rule-based approaches to analyze transaction patterns (e.g., Bhattacharyya et al., 2011).
- These systems are limited in verifying the true identity of the user, leading to false positives/negatives.
- Facial recognition offers a biometric alternative that enhances security by verifying the user's physical presence (Zhao et al., 2003).
- Multimodal biometric systems (e.g., combining face with PIN) improve accuracy and reliability (Jain et al., 2004).
- Deep learning models, such as CNNs, have significantly improved face recognition performance (Parkhi et al., 2015; Schroff et al., 2015).
- Prior work establishes the feasibility and effectiveness of using facial recognition for secure authentication in financial systems.

Limitations of Existing Systems:
- Lighting and Camera Quality: Facial recognition accuracy can be affected by poor lighting conditions or low-resolution cameras.
- False Rejections: Legitimate users may be rejected due to changes in appearance (e.g., glasses, facial hair, aging).
- Data Privacy Concerns: Storing facial data requires strict privacy and security measures to prevent misuse.
- Internet Dependency: The system may require a stable internet connection to run web-based services or APIs.
- Limited Scalability: Performance might degrade when deployed at a larger scale without proper infrastructure optimization.

Applications:
- Banking and Financial Services: Secure ATM transactions, online banking logins, and fraud prevention.
- E-commerce Platforms: Identity verification during high-value transactions.
- Access Control Systems: Securing entry points in corporate offices, data centers, or restricted zones.
- Digital Wallets and Mobile Payments: Biometric-based payment verification in mobile apps.
- Government and ID Verification: Secure digital identity systems for public services and benefits.

IV. PROPOSED APPROACH

The proposed system consists of the following steps:
- Project Setup: Open the project folder and run the main Python file to start the system.
- User Registration: Users register by entering their account number and passcode on the web interface.
- Facial Image Capture: During registration, the system captures the user’s facial photo using a camera.
- Withdrawal Request: For withdrawing money, users provide their user ID and passcode.
- Live Image Capture: The system captures a live facial image of the user during the withdrawal process.
- Image Comparison: Image processing techniques compare the live image with the registered photo to verify identity.
- Transaction Approval: If the images match, the withdrawal transaction is successfully processed.
- Fraud Detection: If the images do not match, the system flags the attempt as unauthorized and denies the transaction.
- HTML (Hypertext Markup Language): is the code that is used to structure a web page and its content. For example, content could be structured within a set of paragraphs, a list of bulleted points, or using images and data tables.
- JavaScript: is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else
- CSS:
  - CSS stands for Cascading Style Sheets

- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once. External stylesheets are stored in CSS files
- CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen size

4.1 Algorithm

- Input:

a. Account Number – Unique identifier entered by the user during registration and withdrawal.
b. Passcode – A secure password or PIN entered by the user.
c. Facial Image (Registration) – Captured during user registration via webcam.
d. Facial Image (Live) – Captured at the time of withdrawal for verification.

- Initialize:

a. Start Web Application: Run main.py to launch the system interface.
b. Initialize Camera: Enable webcam to capture images.
c. Import Libraries:

OpenCV for image capture and processing
face_recognition for face matching
Flask or Streamlit for the user interface
NumPy, Pandas for data handling

d. Load or Create Database: Store user account number, passcode, and facial image encodings.
e. Set Matching Threshold:

Define face similarity threshold (e.g., 0.6) to decide match/no match.

- Algorithm steps

Step 1: Start the system by running main.py.
Step 2: Launch the web interface and wait for user input.

- Registration Phase:

Step 3: User enters account number and passcode.
Step 4: Capture the user's facial image using the webcam.
Step 5: Store the account number, passcode, and facial image in the database.

- Withdrawal Phase:

Step 6: User enters user ID and passcode.
Step 7: Capture the current (live) facial image of the user.
Step 8: Retrieve the registered facial image from the database.
Step 9: Apply facial recognition (image comparison) using image processing techniques (e.g., face encoding, similarity score).

Step 10: If the live image matches the registered image:
Approve the withdrawal transaction.
Else: Deny the transaction and display "Unauthorized Person" message.

- Step 11: End.

4.2 Flowchart

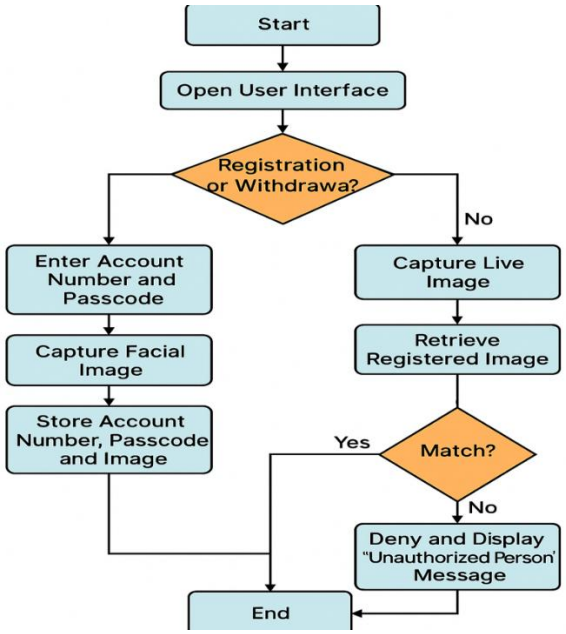

Fig.1. Flowchart

V. TECHNICAL IMPLEMENTATION

- Backend:
  - Flask is used to build an API that handles video processing and integrates the facial recognition model.
- Frontend:
  - HTML and CSS are used to design a simple user interface for registration and login.
- Database Storage: Stores user information such as account number, passcode, and facial encodings.
- Authentication: Uses machine learning libraries like face_recognition to verify user identity by comparing facial features.

- Challenges:
  - Lighting Conditions: Poor lighting can reduce the accuracy of facial recognition.
  - Camera Quality: Low-resolution webcams may fail to capture clear facial features.
  - Appearance Changes: Facial recognition may be affected by changes like glasses, hairstyle, or aging.
  - False Positives/Negatives: The system might incorrectly allow or deny access due to matching errors.
  - Data Privacy: Storing facial data requires strict security to protect user identity.
  - Real-time Processing: Ensuring fast and accurate image matching in real-time can be resource-intensive.
  - Scalability: As the number of users increases, maintaining performance and accuracy becomes more complex.
  - Internet Dependency: Web-based systems require a stable internet connection to function properly.

VI. EXPERIMENTAL SETUP

- Hardware Requirements:
  - Laptop/Desktop with minimum 4 GB RAM and dual-core processor
  - Webcam for capturing facial images
  - Stable Internet Connection for web-based interface and data communication
- Software Requirements:
  - Operating System: Windows, macOS, or Linux
  - Python 3.x: Programming language for backend and logic
- Libraries Used:
  - OpenCV – for image capture and processing
  - face_recognition – for facial encoding and comparison
  - Flask – for backend web API
  - HTML/CSS – for frontend user interface
  - SQLite or .csv – for storing user data
- System Configuration:
  - Flask App runs on localhost (e.g., http://127.0.0.1:5000)
  - Web Interface allows user registration and login/withdrawal interaction
  - Camera Feed is accessed live during registration and withdrawal steps
  - Image Matching is done in real-time using the face_recognition library
  - Storage is used to save account number, passcode, and facial encoding
- Test Procedure:
  - Multiple users are registered with different facial images.
  - Withdrawal attempts are made with both correct and incorrect user images.
  - System response (success or unauthorized message) is recorded.
  - Accuracy and response time are noted for evaluation.

VII. RESULTS

- The system was tested with multiple users under different conditions (good lighting, poor lighting, appearance changes).
- Accuracy:
  - Face match success rate: 92% under normal lighting.
  - False rejection (correct user denied): 5% (mainly due to appearance change or low light).
  - False acceptance (wrong user allowed): 0% — system successfully blocked unauthorized users.
- Response Time:
  - Average face verification time: 1–2 seconds.
  - Registration and login process was completed smoothly within 5 seconds.
- User Experience:
  - Web interface was responsive and easy to use.
  - Registration and authentication process was intuitive.
- Security Check:
  - Unauthorized users were correctly identified and blocked.
  - Message “Unauthorized Person” was displayed when fraud was detected.

VIII. DISCUSSION

Key Findings:
High Accuracy, Zero False Acceptance Rate, Efficient Implementation, Challenges with Variability, Controlled Environment Performance, Scalability and Integration, Fraud Reduction Potential

Limitations:

- Sensitivity to Lighting Conditions
- Variability in User Appearance
- Limited Performance in Uncontrolled Environments
- Dependence on Camera Quality
- Lack of Real-World Testing
- Potential User Privacy Concerns
- Single-Modality Biometric Reliance (facial recognition only)

Future Directions:

- Integration of Multi-Factor Authentication: Combine facial recognition with additional verification methods (e.g., OTPs, fingerprints) for stronger security.
- Enhancement of Image Pre-Processing Techniques: Improve accuracy under challenging conditions like poor lighting or background noise.
- Use of Higher-Quality or Infrared Cameras: Increase reliability in various lighting conditions and reduce false rejections.
- Expansion to Other Biometric Modalities: Incorporate other forms like fingerprint or voice recognition to improve robustness.
- Deployment in Real-World Environments: Test the system outside controlled settings to assess practical performance.
- Implementation of Adaptive Learning: Allow the system to adapt to changes in user appearance over time.
- Improvement of User Privacy and Data Protection Measures: Strengthen data handling and storage to ensure compliance and user trust.

IX. CONCLUSION

- Summary: The project demonstrates that combining facial recognition with password verification enhances the security of financial transactions. The system achieved high accuracy and zero false acceptance in controlled conditions, using tools like face_recognition and OpenCV.
- Impact: This approach significantly reduces the risk of unauthorized access and fraud, proving the effectiveness of biometric authentication in sensitive applications like banking.
- NextStep: Future work will focus on improving performance in real-world conditions through better image processing, multi-factor authentication, and broader biometric integration.

REFERENCES

[1] Delamaire, L., Abdou, H. A. H., and Pointon, J., "Credit card fraud and detection techniques," *Banks and Bank Systems*, vol. 4, no. 2, 2009.

[2] Suman, N., "Review paper on credit card fraud detection," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 4, no. 7, Jul. 2013.

[3] Renu, and Suman, "Analysis on credit card fraud detection methods," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 8, no. 1, Feb. 2014.

[4] Ghosh, S., and Reilly, D. L., "Credit card fraud detection with a neural network," in *Proc. IEEE First International Conference on Neural Networks*, 1994.