Face Authentication System:

Secure Login with Facial Recognition

Suppakorn Pojsomphong, Yotsawan Sangkavaree, Paranee Wannasorn

1. Introduction

In today's digital era, security in authentication and identity verification is of paramount importance. Traditional authentication methods, such as passwords and PINs, are susceptible to various cyber threats, including phishing, brute force attacks, credential leaks, and replay attacks. As the reliance on online systems continues to grow, there is an increasing need for robust, secure, and user-friendly authentication mechanisms.

Biometric authentication has emerged as a more secure alternative, as it leverages unique biological characteristics, making it significantly harder to forge or steal. Among various biometric modalities, facial recognition is widely used due to its non-intrusive nature and ease of implementation. However, implementing a secure face authentication system requires addressing key challenges, such as spoofing attacks, storage security, and model accuracy.

This project presents a Face Authentication System that integrates deep learning and computer vision techniques for secure user authentication. The system utilizes a Convolutional Neural Network (CNN) for feature extraction and implements cosine similarity for identity verification. Additionally, security measures such as bcrypt for password hashing and prepared statements in MySQL are incorporated to mitigate security vulnerabilities, particularly SQL injection attacks.

2. Technology Used

The system is developed using a combination of machine learning frameworks, database management systems, and web technologies to ensure efficiency, accuracy, and security.

2.1 Software & Frameworks

- **OpenCV**: Used for face detection and image preprocessing, including grayscale conversion and resizing.
- **TensorFlow/Keras**: Used for training and deploying the deep learning model for face recognition.
- **MySQL**: Serves as the backend database to store user credentials and facial embeddings securely.

2.2 Programming Languages

- **Python**: The primary language used for backend development, image processing, and machine learning model integration.

2.3 Algorithm & Security Measures

To ensure robust security, multiple techniques are integrated into the system:

- **CNN-based Face Recognition**: A convolutional neural network is trained to extract facial embeddings, capturing unique identity features.
- **Cosine Similarity**: Used to measure the similarity between stored and captured face embeddings for verification.

- **Password Hashing (bcrypt)**: Ensures secure storage of user passwords by using salted hashing to prevent credential leaks.
- **Prepared Statements (MySQL)**: Prevents SQL injection attacks by using parameterized queries.

3. System Design & Implementation

3.1 Face Recognition Model

The Face Authentication System is built on a CNN-based facial recognition model that extracts high-dimensional embeddings from facial images. The key steps in model training and deployment include dataset selection, preprocessing, architecture design, and feature extraction.

Dataset: Georgia Tech Face Database

The Georgia Tech Face Database is used to train and evaluate the system. This dataset contains images of 50 individuals, with 15 color JPEG images per subject. The images include variations in facial expressions, lighting conditions, and background settings, ensuring the model's robustness.

- **Source**: Georgia Tech Face Database [6].
- **Dataset Details**:
    - 50 subjects
    - 15 images per subject (**total: 750 images**)
    - Cropped face images
    - **Training & Validation Split**:
        - 20 subjects selected
        - 10 images per subject for training
        - 5 images per subject for validation

Model Training Steps

1. **Data Preprocessing**: Images are converted to grayscale, resized to 100x100 pixels, and normalized between 0 and 1 for consistent input representation.
2. **CNN Architecture**: The model comprises convolutional layers for feature extraction, followed by fully connected layers to generate embeddings.
3. **Feature Embeddings**: Instead of classifying faces, the model generates unique feature vectors (embeddings) representing each individual.
4. **Cosine Similarity for Verification**: The stored and newly captured embeddings are compared using cosine similarity, with authentication granted if similarity exceeds a predefined threshold (e.g., 0.8).

3.2 User Authentication Process

1. Registration Process

1. **User Input**: The user provides a username and password.
2. **Face Capture**: The system captures an image of the user via a webcam.
3. **Face Detection**: OpenCV's Haar cascade classifier detects and extracts the face from the image.
4. **Embedding Generation:** The preprocessed face is passed through the CNN model to obtain a feature embedding.

5. **Password Hashing**: The password is hashed using bcrypt before being stored.
6. **Database Storage**: The hashed password and face embedding are securely stored in the MySQL database.

2. Login Process

1. **User Input**: The user enters their username and password.
2. **Face Capture**: A live image of the user's face is captured.
3. **Embedding Extraction**: The captured face is preprocessed, and a feature embedding is generated.
4. **Password Verification**: The stored password hash is verified using bcrypt.
5. **Cosine Similarity Calculation**: The stored embedding and captured embedding are compared using cosine similarity.
6. **Authentication Decision**:
    - If similarity ≥ 0.8, access is granted.
    - Otherwise, authentication fails, and access is denied.

4. Model Training and Performance

4.1 Model Training Process

The facial recognition model was trained using a Convolutional Neural Network (CNN). The training process followed these steps:

1. **Dataset Preparation**: Images were preprocessed using OpenCV to detect faces and extract features.
2. **CNN Model Architecture**: The model consisted of three convolutional layers, batch normalization, and ReLU activation functions.
3. **Training Configuration**:
    - **Optimizer**: Adam
    - **Loss Function**: Categorical Cross-Entropy
    - **Batch Size**: 32
    - **Epochs**: 10
    - **Validation Split**: 20%

4.2 Training Accuracy

The accuracy curve represents training and validation accuracy over 10 epochs. The model quickly converged, reaching a high accuracy of approximately 99% within a few epochs, indicating strong feature extraction and classification capabilities as shown in Figure 1.

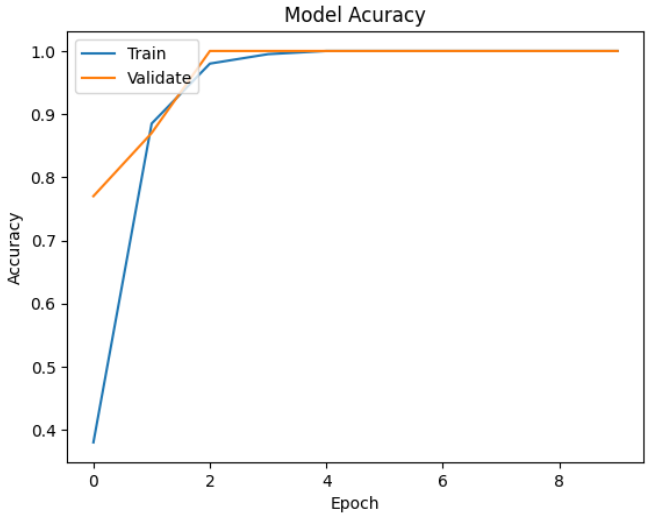

Figure 1: Training Accuracy

4.3 Training Loss

The training loss graph provides additional insight into the model's learning process. The rapid decrease in loss during the first few epochs suggests efficient feature learning. By epoch 4, the loss reaches a stable minimum close to zero, indicating that the model successfully learned to distinguish faces with minimal error as shown in the Figure 2.

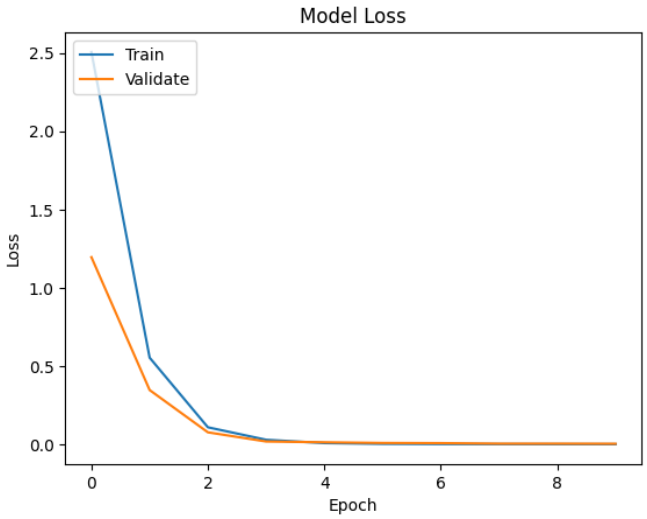

Figure 2: Training Loss vs. Validation Loss

4.4 Performance Analysis

- The rapid convergence of the accuracy and loss curves indicates a well-trained model.
- The gap between training and validation loss is minimal, meaning the model generalizes well to unseen data.
- Future improvements could include training on a larger dataset to further enhance robustness and avoid potential overfitting.

5. Demo & Results

5.1 System Workflow

1. User Registration:

- The user enters a username and password.
- The system captures an image of the user's face and extracts an embedding using the trained model.
- The password is securely hashed, and both the hashed password and embedding are stored in MySQL.

2. User Login:

- The user enters their credentials.
- The system captures a new image of the user's face, extracts its embedding, and compares it with the stored one using cosine similarity.
- If the face matches and the password is correct, access is granted.

3. Security Measures:

- Password hashing ensures secure credential storage.
- Cosine similarity provides accurate face authentication.
- Prepared statements protect against SQL injection attacks.

5.2 Authentication Success & Failure

- **Success Case**: If a registered user provides the correct password and a matching face, access is granted.
- **Failure Cases**:
  - If the password is incorrect, authentication fails.
  - If the face does not match, access is denied.

5.3 Sample Execution Output

Example 1:

```
Enter your choice: 2
Enter username: alice
Enter password: mysecurepassword
Look at the camera for login...
Cosine Similarity: 0.85
✅ Login successful!
```

Example 2:

```
Enter your choice: 2
Enter username: alice
Enter password: wrongpassword
❌ Incorrect password. Login failed.
```

Example 3:

```
Enter your choice: 2
Enter username: alice
Enter password: mysecurepassword
Cosine Similarity: 0.55
❌ Face does not match. Login failed.
```

6. Conclusion

This project successfully demonstrates a secure face authentication system using deep learning, OpenCV, and MySQL. By integrating facial recognition, password hashing, and SQL injection prevention, the system offers enhanced security over traditional authentication methods. The CNN-based model efficiently recognizes faces, and the authentication mechanism ensures secure login using both password verification and facial verification.

7. Future Development & Suggestions

1. **Enhancing Model Accuracy**: Train the model on a larger and more diverse dataset.
2. **Multi-Device Authentication**: Enable authentication across various devices.
3. **Mobile App Integration**: Extend the system to a mobile platform.
4. **2-Factor Authentication (2FA)**: Implement an additional security layer.
5. **Live Detection Mechanism**: Add liveness detection to prevent spoofing attacks

8. References

[1] TensorFlow/Keras Documentation, "TensorFlow," 2024. [Online]. Available: https://www.tensorflow.org/. [Accessed: 15-Mar-2025].

[2] OpenCV Documentation, "OpenCV Library," 2024. [Online]. Available: https://opencv.org/. [Accessed: 15-Mar-2025].

[3] MySQL Official Documentation, "MySQL 8.0 Reference Manual," 2024. [Online]. Available: https://dev.mysql.com/doc/. [Accessed: 15-Mar-2025].

[4] bcrypt Library, "bcrypt Python Package," 2024. [Online]. Available: https://pypi.org/project/bcrypt/. [Accessed: 15-Mar-2025].

[5] Wikipedia, "Cosine similarity," Wikipedia, The Free Encyclopedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Cosine_similarity. [Accessed: 15-Mar-2025].

[6] A. Nefian, "Georgia Tech Face Database," 2002. [Online]. Available: http://www.anefian.com/research/face_reco.htm. [Accessed: 15-Mar-2025].