# 

Face attendance system report

Group 3

Tran Hoang Hai Anh, 104177513

Lai Gia Khanh, 104222015

Truong Minh Son, 104221795

Table of contents

[**Contribution 4**](#_wwaqkygdd8z9)

[**Abstract 4**](#_k075zvfbk9c4)

[**Introduction 6**](#_gaoghrvml7ip)

[Overview 6](#_fjfhtd3ec7y)

[Objective 6](#_6h5ddq432zxg)

[Main Objectives 6](#_iz9r8iwxdy7w)

[Face Registration 6](#_nqyeu5hdxexd)

[Face Verification 6](#_tc3y2091hl4)

[Anti Spoofing 6](#_wj45rffx7782)

[Modules 7](#_1w4hi56oaewd)

[List of main modules 7](#_2tnwbs9irsi)

[Dataset and Preprocessing Files (data\_loader.py) 7](#_uenfgtt20ljx)

[Model definition (model.py) 7](#_supurl8pv129)

[Training script (train.py) 7](#_ru2x09qxs9vq)

[Evaluation Script (evaluate.py) 7](#_ksvvocprm5y)

[Utility Files (utils.py) 7](#_7bn6ohi049cb)

[Main Script (main.py) 7](#_93pgkfc94bdf)

[**Methodology 7**](#_5bak3p4s5w28)

[Face Embedding 8](#_k6uhcqsxek4t)

[InceptionV3 8](#_xe9zvtdzpasv)

[Triplet Loss Function 8](#_a2slaz4v2tpi)

[Similarity Metric 9](#_u9o9o7hxfcbp)

[Anti-Spoofing Module 10](#_6s6xruashvpv)

[UI Development 11](#_r1ipfw3u7ku3)

[**Implementing 13**](#_vaql8mfmrvi)

[File Structure 13](#_9taze6lutbd3)

[Training process 13](#_jq04xpusplgk)

[Model learning 13](#_2zmxabxw5nyr)

[Model validated 14](#_y5imdnd3pr3c)

[Testing process 15](#_1051aps3f8wo)

[ROC Curve 15](#_y5ir05kfukel)

[Evaluating script 15](#_vvz46iu7jonm)

[**Results and Discussion 17**](#_kr3kfb5fcijv)

[Challenges in training 17](#_f6yv8uwb8qiv)

[Tuning 17](#_38fzu0b9zd0u)

[Improvements after tuned 17](#_tqew35atsxzy)

[Dataset sampled 18](#_ykhckv63k1yp)

[Adding a new face 18](#_aj50egy0eaby)

[Benchmark 20](#_s46lh52gmd1)

[Anti spoofing 21](#_5aqt2b8nuajm)

[Enhancements 22](#_a05fs1havo0o)

[Hyperparameters tuning 22](#_1czta8gvea54)

[Anti spoofing strategies 22](#_h9y1j4w8wxcj)

[Distance Measurement to Enhance Confirmation 22](#_gkvv4ul1meul)

[UI (User interface) 22](#_cqqb6ts0ewfw)

[Flexibility 22](#_4ayvgftrhsbs)

[**Conclusion 22**](#_8sfqrm79jqd4)

[**Future Improvements 23**](#_a0qg081irq06)

[**References 23**](#_505e67s0tk0z)

# 

# **Contribution**

| Name | Workload |
| --- | --- |
| Tran Hoang Hai Anh | * Researching about the technologies used in this project. * Implementing, training and evaluating the face recognition with anti spoofing model. * Writing the Abstract, Modules, Implementing, Results and Discussion and Future Improvements parts in the report. |
| Lai Gia Khanh | * Researching about the technologies used in this project. * Testing face recognition with anti spoofing models with real faces. * Writing the Methodology part in the report. |
| Truong Minh Son | * Testing face recognition with anti spoofing models with real faces. * Testing other groups' models in the live demonstration session in class. * Writing the Introduction and Conclusion parts in the report. |

# **Abstract**

This study describes the design and testing of a facial recognition attendance system with anti-spoofing features. Face verification is performed using Convolutional Neural Networks (CNNs) and the Triplet Loss function. Adjustments such as raising the margin to 1.2 and decreasing the LR to 0.0001 improved the performance of the model. The system attained a Receiver Operating Characteristic (ROC) AUC score of 0.94 after training on a 1000-sample dataset in 100 epochs.

To further security, the system contains an anti-spoofing module that detects blinks with Eye Aspect Ratio (EAR) and uses sophisticated face analysis to detect deep fakes and analyze emotions. The Chebyshev distance was found to be the most efficient method for face verification. Streamlit's user-friendly interface allows for simple GUI face registration and authentication.

The system provides high accuracy in facial recognition, effective anti-spoofing, and an easy-to-use interface, making it dependable for attendance verification. Future enhancements will focus on improved triplet selection, optimized hyperparameters, sophisticated data augmentation, and real-life usage to increase the overall processing time and accuracy.

# **Introduction**

## Overview

Face recognition technology is a critical component of current security and authentication solutions. It is separated into two main tasks: face categorization and face verification.

Face Classification recognizes and categorizes a face using a predetermined set of IDs. This is a closed-set issue in which the system learns to recognize faces from a predefined database. For example, in the workplace, the system associates an employee's face with their matching ID based on training data.

Face Verification checks if two facial photos belong to the same individual. Unlike classification, it is an open-set issue in which the system may meet new faces that were not in the training dataset. The purpose is to compute a similarity score to determine if the faces match. This is utilized in a variety of purposes, including device unlocking, security, and attendance verification.

Understanding the distinction between face categorization and verification is critical. Each has its own set of methodologies and issues that must be addressed in order to create good face recognition systems.

## Objective

This project intends to develop a comprehensive Face Recognition Attendance System for business contexts. The system will use convolutional neural networks (CNNs) for exact face verification and include an anti-spoofing module to detect and block fraudulent efforts to use phony or non-real faces. This assures excellent accuracy and strong security.

### Main Objectives

#### Face Registration

The technology will assist in the registration of new face IDs when employees join the business. This function maintains the database of recognized faces up-to-date and adaptive to personnel changes.

#### Face Verification

A CNN-based verification method will be used to compare two facial photos and calculate their similarity score. This guarantees precise and reliable identification of individuals.

#### Anti Spoofing

An anti-spoofing technique will be implemented to identify and prevent spoofing efforts, such as those involving photographs or videos of faces. This feature guarantees that the system only detects live, actual faces.

## Modules

### List of main modules

#### Dataset and Preprocessing Files (data\_loader.py)

Provides custom dataset classes to manage training, validation, and testing data for face verification. Resizes photos, normalizes pixel values, and applies data augmentations to assist model training. Loads data efficiently using PyTorch's DataLoader, allowing for batch processing and data shuffling.

#### Model definition (model.py)

Uses the InceptionV3 architecture, which includes convolutional, pooling, inception modules, and fully linked layers for face embedding. The last layer is replaced with a fully linked layer and batch normalization to get normalized embeddings that capture distinct face traits.

#### Training script (train.py)

Includes a bespoke training loop for handling model training, validation, and assessment across numerous epochs.

TensorBoard is used to track data like loss and accuracy as well as to visually represent training progress.

Saves the top-performing model based on validation loss for optimal deployment.

#### Evaluation Script (evaluate.py)

This file assesses a face verification model's performance by calculating similarity scores between pairs of face photos and producing metrics such as the Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC). It allows for both individual comparisons and batch evaluation.

#### Utility Files (utils.py)

Contains tools for loading photos, creating face embeddings, and comparing embeddings for face recognition. Includes routines for drawing face frames, cropping identified faces, and doing model inference for embedding prediction. Improves security by providing anti-spoofing capabilities such as expression analysis, deep fake detection, and valence-arousal estimate.

#### Main Script (main.py)

Creates an interactive interface using Streamlit to allow for easy interaction with the face recognition technology. Enables live face identification and verification via camera, providing real-time feedback. Liveness detection with Eye Aspect Ratio (EAR) detects blinks and prevents spoofing with static pictures or movies. Supports face registration, identity verification, and result presentation for a more seamless user experience.

# **Methodology**

## **Face Embedding**

### InceptionV3

The face embedding module utilizes a Convolutional Neural Network (CNN), specifically InceptionV3 architecture as the base model, to convert facial images into low-dimensional feature embeddings. The InceptionV3 model is fine-tuned by replacing its final layers with a custom fully connected layer and a batch normalization layer to produce a 128-dimensional output. This adjustment helps the model focus on unique facial characteristics required for face verification.

During the forward pass, the input image is resized to fit the InceptionV3’s expected input dimensions. The model extracts features and the output embedding is normalized to ensure that its values lie on a unit sphere. This normalization helps maintain consistency in embeddings, making it easier to compare similarities across different images.

The embeddings are implemented through the “create\_embeddings” function which processes images in the “known\_face” folder to store embeddings of known faces for later comparisons in face verification.

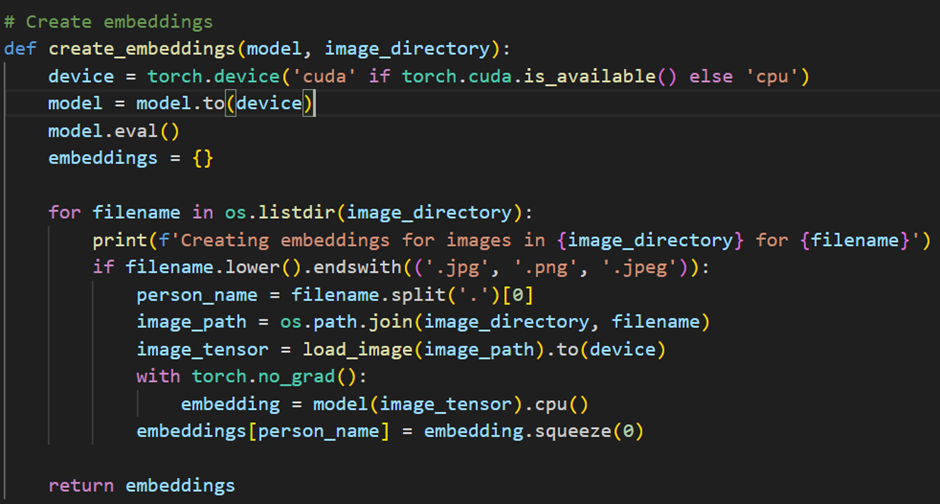


Figure 1: The create embeddings function

### Triplet Loss Function

* The model is trained using the Triplet Loss function, which ensures that embeddings for the same person are grouped closely together in the feature space, while embeddings for different individuals are well-separated.
* How Triplet Loss works:
  + Triplet Loss aims to minimize the distance between the anchor and positive embeddings while maximizing the distance between the anchor and negative embeddings. The loss function is defined as:

***Loss = max{d(A, P) – d(A, N) + margin, 0}***

Where:

* + - **Anchor (A):** A reference image of a person.
    - **Positive (P):** Another image of the same person.
    - **Negative (N):** An image of a different person.
    - **d(A, P):** Distance between anchor and positive embeddings.
    - **d(A, N):** Distance between anchor and negative embeddings.
    - **Margin:** A hyperparameter specifying the minimum required difference between the two distances. This ensures a significant separation between embeddings of different people.
* The system incorporates Triplet Loss as a key component to optimize the face embedding process which can improve the overall performance and robustness of the face recognition system.

## **Similarity Metric**

The system employs multiple distance metrics to compare face embeddings and determine similarity:

* **Cosine Similarity:** Measures the cosine of the angle between two embedding vectors, where higher scores indicate greater similarity. The system uses the PyTorch cosine similarity function to calculate the score. This is a suitable metric used for face verification because of its effectiveness with normalized embeddings.

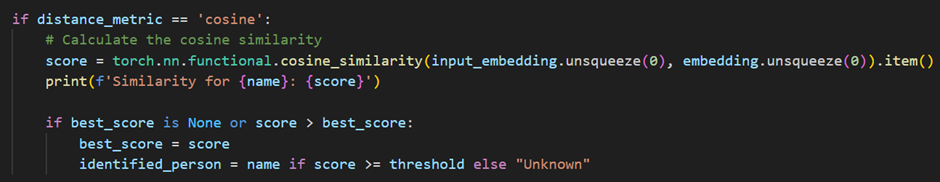


Figure 2: The cosine similarity calculation

* **Euclidean Distance:** Calculates the straight-line distance between embeddings. The metric is implemented with “pairwise\_distance” in PyTorch, measuring the absolute distance between two points where lower scores indicate greater similarity.

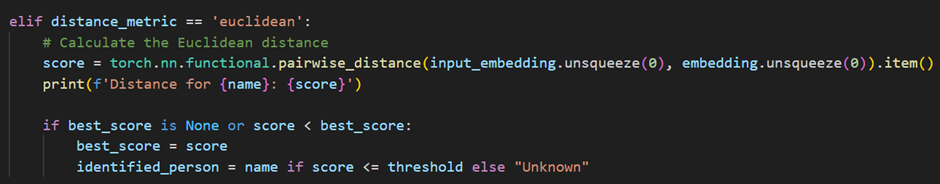


Figure 3: Euclidean distance calculation

* **Manhattan Distance:** Computes the sum of absolute differences along each feature dimension, offering an axis-aligned measure of similarity. This metric is useful when the difference along each feature dimension contributes independently to overall similarity. Lower scores imply a closer match.

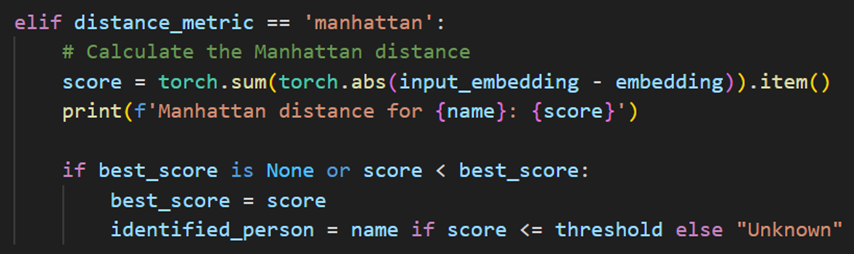


Figure 4: Manhattan distance calculation

* **Chebyshev Distance:** Uses the maximum absolute difference across all dimensions, focusing on the largest single-dimension difference, which may be suitable when significant differences in even one dimension suggest dissimilarity.

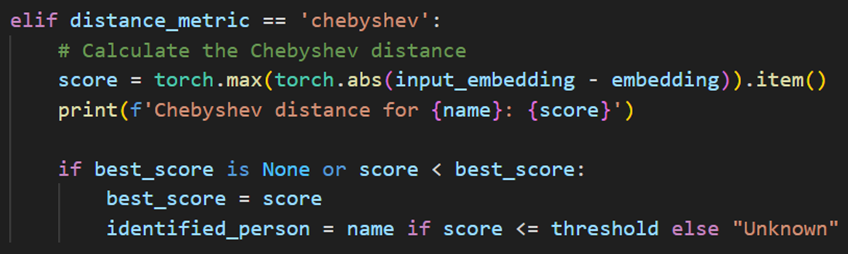


Figure 5: Chebyshev distance calculation

## **Anti-Spoofing Module**

The Anti-Spoofing Module is implemented using liveness detection techniques to prevent spoofing attacks.

* **Eye Aspect Ratio (EAR):** The system detects blinks as a form of live detection. The EAR is computed to measure the eye’s openness level, where rapid changes in the ratio (indicating blinks) signify that the user is present and alive.
  + Implementation:
    - **Blink Detection:** The EAR threshold is set at 0.275 and a count is maintained to detect a blink pattern. If the user completes a specified number of blinks within a time frame, they are considered “real” and allowed to proceed.
  + Purpose: This module effectively counters photo or video-based spoofing attempts. Since static images lack blinking action, they fail to meet the liveness criteria, adding a layer of security to the system.

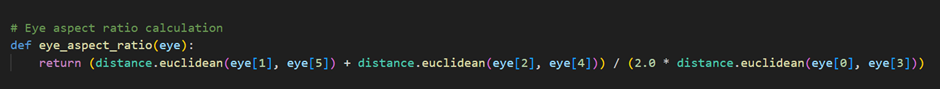


Figure 6: Eye aspect ratio function

* **Facial Expression Recognition and Deepfake Detection:** These two techniques are also applied to strengthen the project’s anti-spoofing capabilities, ensuring that the system can differentiate real people and spoofing attempts more effectively.

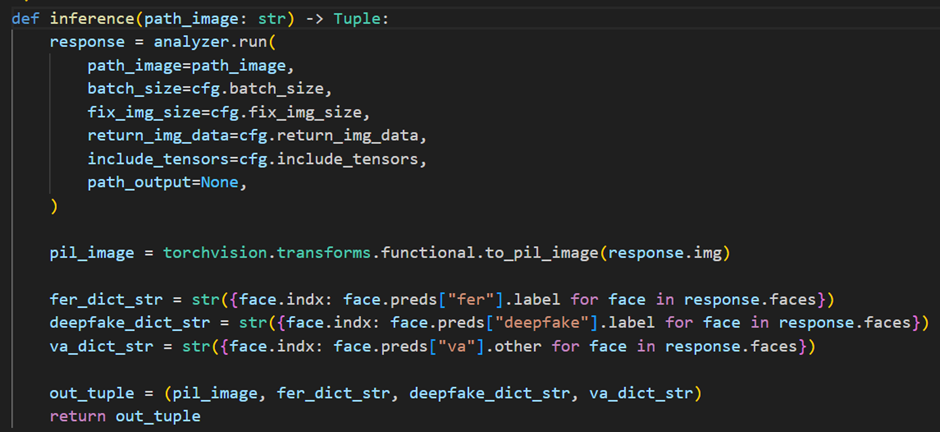


Figure 7: Inference function

## **UI Development**

* Streamlit, a widely-used Python framework for building interactive web applications, is utilized to develop the website for our face attendance system project, ensuring a seamless and user-friendly experience.
* Interactive UI with Streamlit:
  + Streamlit enables real-time updates and controls, providing immediate feedback during face verification or registration.
  + Users can easily perform tasks such as face verification, resetting the system, or registering new faces through simple, accessible buttons.
* Live Webcam:
  + The system captures video frames using the webcam and processes them in real-time.
  + The application continuously performs face detection, frame drawing, and liveness verification using Dlib and OpenCV.
  + The results such as liveness status or recognition success, are displayed instantly, enhancing user engagement.
* New Face Registration:
  + Users can register new faces by uploading images directly through the website.
  + The system detects and processes the uploaded images, storing their embeddings for verification tasks.

# **Implementing**

## **File Structure**

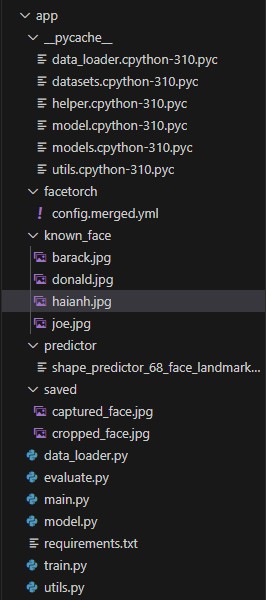


Figure 8: The structure of files

## **Training process**

### Model learning

The training loop iterates through the dataset for a predetermined number of epochs.

In each epoch, the model processes input in batches, creates embeddings, and calculates triplet loss. The optimizer modifies model weights to lessen the loss.

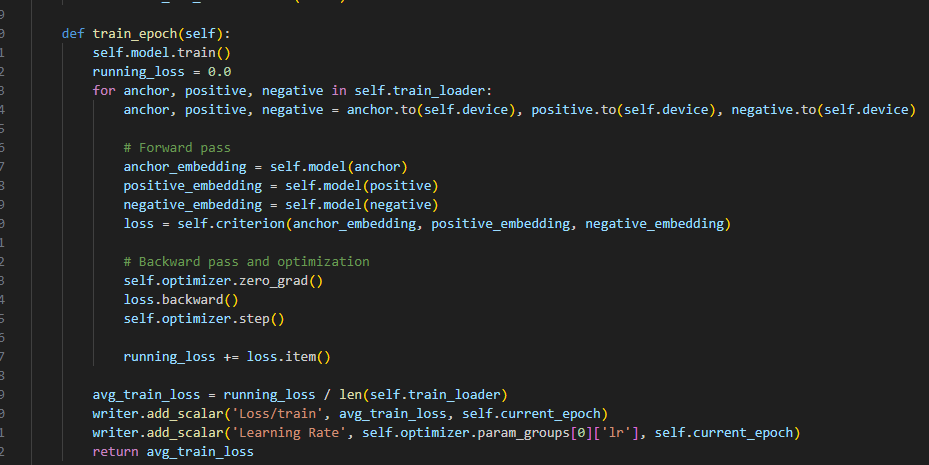


Figure 9: The train epoch function

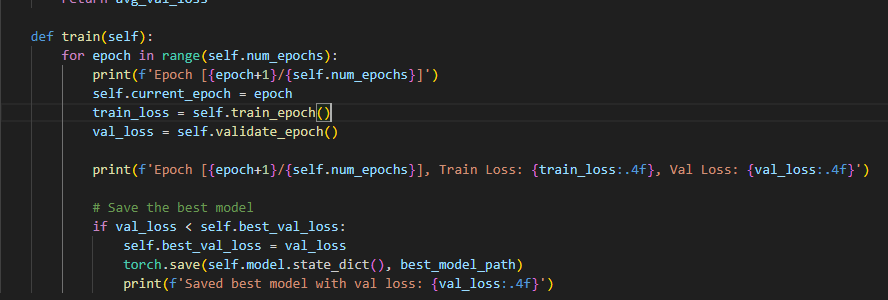


Figure 10: The train function

### Model validated

The model is assessed on the validation dataset after each epoch.

Validation loss is calculated to evaluate performance and prevent overfitting risks.

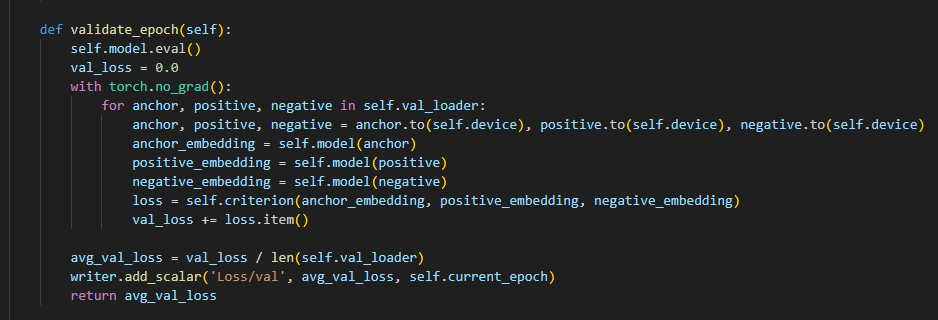


Figure 11: The validate epoch function

## **Testing process**

### ROC Curve

Evaluate the performance of a classification or prediction model, particularly for binary classification problems. It helps to see the trade-off between sensitivity (true positive rate) and specificity (false positive rate) at various threshold levels.



Figure 12: The ROC curve visualization

### Evaluating script

The evaluate.py script loads the training model and applies it to the test dataset.

The ROC curve and AUC are computed using predictions alongside actual labels.



Figure 13: The main function of the evaluating script

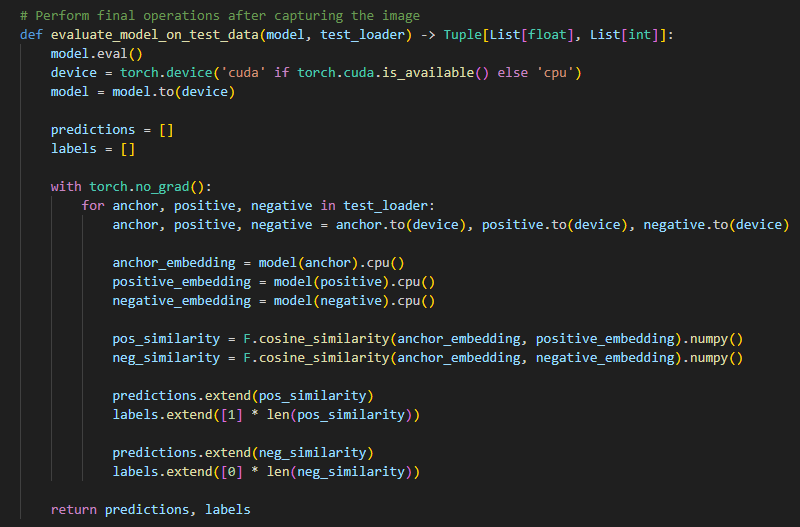


Figure 14: The evaluate on testing data function

# **Results and Discussion**

## Challenges in training

Triplet Selection: Good triplets are essential for learning. Easy triplets have little influence, however severe triplets may disrupt training.

Convergence issues: Triplet loss might result in large variation, rendering training unstable. Correct learning rates and margin values are required.

Data imbalance: Uneven class distribution has an impact on triplet selection and learning, decreasing generalization.

## Tuning

Margin Adjustment: Raised the Triplet Loss margin from 1.0 to 1.2 to improve separation of positive and negative pairings.

Learning Rate Reduction: Reduced the optimizer's learning rate from 0.001 to 0.0001, resulting in more steady and progressive modifications.

### Improvements after tuned

- A greater Triplet Loss margin improved separation between positive and negative pairings, hence accelerating learning.

- A decreased learning rate resulted in more steady updates, decreasing overshooting and helping convergence.

- Validation loss has decreased and stabilized, indicating greater generalization and less overfitting.

## Dataset sampled

The model was trained on a reduced dataset of 1,000 samples across 100 epochs to improve training and testing performance with low data resources.

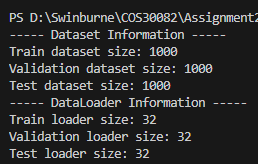


Figure 15: The information of the data loader

## Adding a new face

To recognize new faces that were not in the training dataset, the model must be able to generalize effectively.

It generates embeddings for fresh faces that may be compared to registered ones.

The identification process involves comparing embeddings using distance measures.

Generalization is checked on a validation collection of unseen faces.

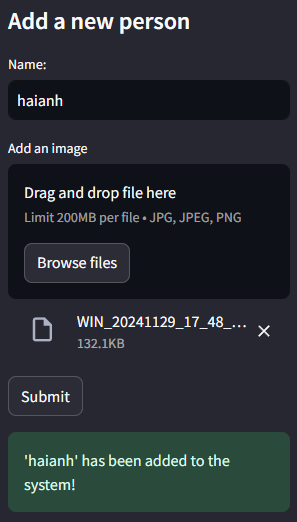


Figure 16: A new user has been successfully added to the system

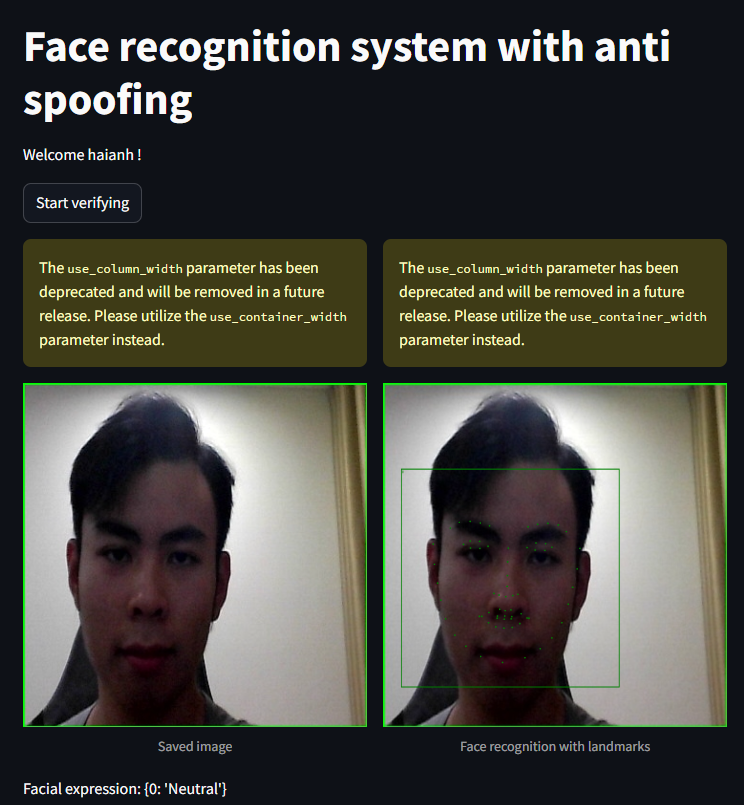


Figure 17: The system successfully recognized Hai Anh

## Benchmark

The model's performance is evaluated using the ROC curve and AUC.

The ROC curve depicts performance at various thresholds, and the AUC highlights the capacity to distinguish positive and negative pairings.

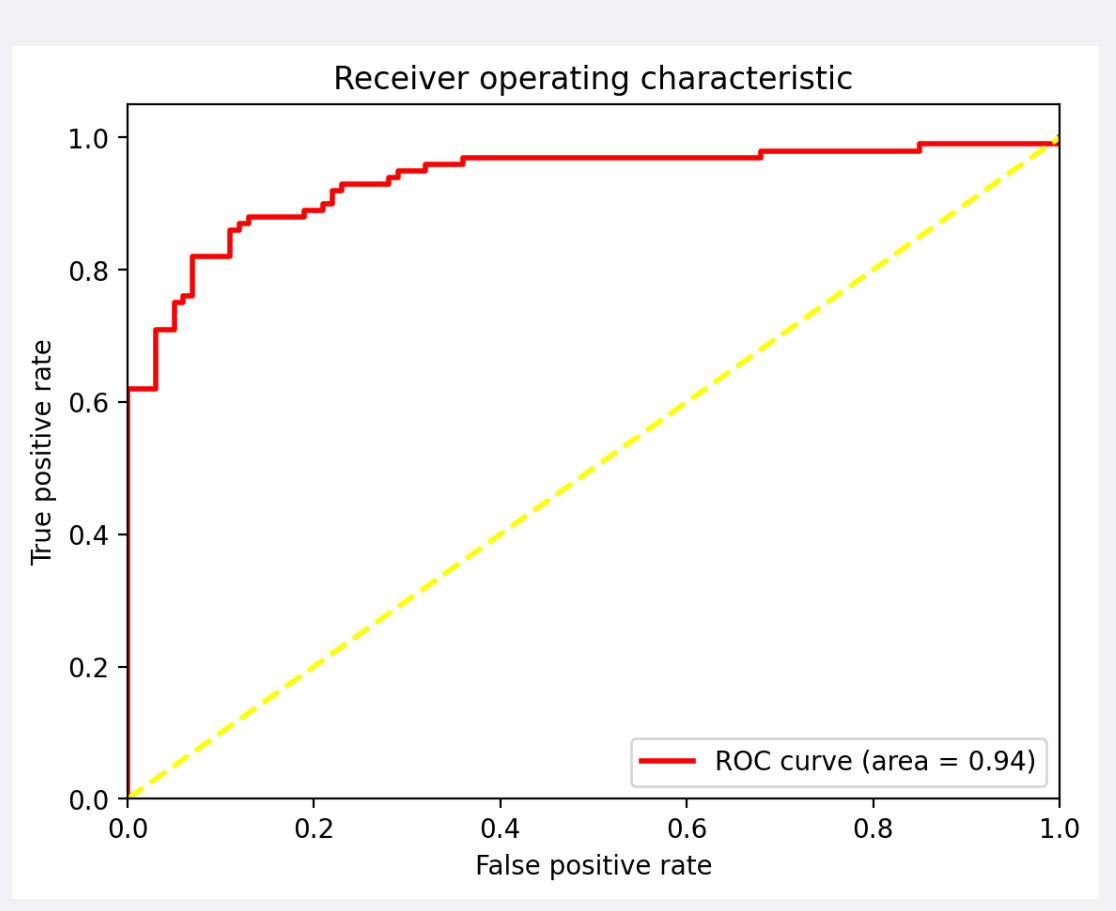


Figure 18: The receiver operating characteristic visualization with 100 samples

The results indicate the model's ability to differentiate between similar and dissimilar faces using the ROC curve and AUC. On a 100-sample test dataset, the ROC AUC score of 0.94 shows great accuracy in detecting positive and negative pairings.

## Anti spoofing

The anti-spoofing module detects and prevents spoofing attempts involving static pictures or videos. It employs the eye aspect ratio to detect blinks, guaranteeing that only live faces are detected and prevents the system from being misled by non-living images.

To improve anti-spoofing capabilities, the system incorporates additional approaches. The FaceAnalyzer, which does image inference to identify facial expressions, deep fakes, and valence-arousal estimates, is configured using the config.merged.yml file. These strategies improve the system's capacity to detect different types of spoofing.

The efficiency of the anti-spoofing module is assessed by subjecting it to spoofing efforts such as printed pictures and movies. The system's ability to identify these efforts while still providing accurate face verification is evaluated. The findings indicate that the anti-spoofing module, when paired with other measures, effectively prevented unauthorized access, confirming the system's resiliency.

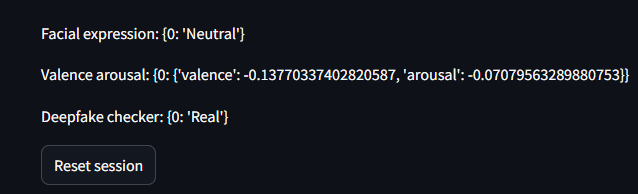


Figure 19: The facial expression, valence arousal and deep fake checking result

## Enhancements

### Hyperparameters tuning

Fine-tuning the hyperparameters for the Triplet Loss function increased the system's performance. Increasing the margin to 1.2 and decreasing the learning rate to 0.0001 improved stability, resulting in a 0.94 ROC AUC score on the test dataset. This tweaking was critical to increasing the model's capacity to discriminate between similar and distinct faces.

### Anti spoofing strategies

The anti-spoofing module combines blink detection based on eye aspect ratio with extensive facial analysis. It also includes deepfake detection, expression recognition, and valence-arousal estimation to successfully avoid spoofing efforts with great precision.

### Distance Measurement to Enhance Confirmation

The Chebyshev distance metric for face verification, which quantifies the greatest absolute discrepancies across embeddings, is presented in this research. It performed better than other indicators, increasing the efficacy of the system and boosting individual identification accuracy.

### UI (User interface)

Easy face registration, verification, and liveness recognition with real-time feedback are all made possible by the system's interactive, user-friendly Streamlit-based interface. This improves usability and makes using the system easy and effective.

### Flexibility

The system can handle big datasets and be implemented in the real world since it is scalable. Future improvements, such as sophisticated triplet mining and data augmentation methods, are made possible by its strong architecture.

# **Conclusion**

The Triplet Loss function and CNN-based face embeddings are used by the face recognition attendance system to effectively distinguish between similar and dissimilar faces. The system's performance and training stability were greatly enhanced by adjusting hyperparameters, such as raising the margin to 1.2 and decreasing the learning rate to 0.0001. For face verification, the Chebyshev distance measure worked especially well, outperforming other metrics in terms of performance.

The system has an anti-spoofing module that can identify and stop spoofing attempts in order to improve security. Streamlit's user-friendly interface guarantees a seamless and engaging face registration and verification process. Furthermore, following initial training, the model correctly identified unknown faces, demonstrating strong generalization. The system has great accuracy in differentiating between positive and negative pairings, as evidenced by its ROC AUC score of 0.94 on a test dataset consisting of 100 samples.

# **Future Improvements**

There are several chances to improve the system even further. To enhance triplet selection and maximize model training, sophisticated triplet mining techniques can be used. Training stability and performance might be further improved by ongoing hyperparameter adjustment, which includes experimenting with learning rates, margins, and batch sizes.

The danger of overfitting would be decreased by increasing the variety in training data through the use of sophisticated data augmentation techniques. Additional protections against complex spoofing efforts may be offered by improvements to the anti-spoofing module, such as motion detection techniques and texture analysis based on machine learning.

Lastly, one of the top priorities is to optimize the system for real-world deployment. Making sure the system is scalable and resilient in a variety of settings will increase its viability for broad adoption.

# **References**

Rosebrock, A. (2023, May 12). Face detection with dlib (HOG and CNN) - PyImageSearch. PyImageSearch. <https://pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/>

Hashmi, F., & Hashmi, F. (2021, September 27). Face Recognition using Deep Learning CNN in Python - Thinking Neuron. Thinking Neuron - Data Science application to real world problems! <https://thinkingneuron.com/face-recognition-using-deep-learning-cnn-in-python/>

Knowledge\_Share. (2024, January 5). # Face recognition using CNN Architecture in Python. Medium. <https://medium.com/@raguwing/face-recognition-using-cnn-architecture-in-python-f3c302c2164f>

Team, M. (2024b, February 21). Anti-spoofing techniques for liveness detection in face recognition. MobiDev. <https://mobidev.biz/blog/face-anti-spoofing-prevent-fake-biometric-detection>

Bluemel, J. (2024, February 20). Fraud Detection: How Anti-spoofing facial recognition works. IDnow. <https://www.idnow.io/blog/fraud-detection-anti-spoofing-facial-recognition/>