Gan For Fingerprint Image Generation

Vijay S[1], Naveenrajj[2], Kalaiyarasan K[3]
[1, 2, 3]Dept of Computer Science And Engineering
[1, 2, 3] TRICHY ENGINEERING COLLEGE

Abstract- *Fingerprint-based biometric systems are widely used for secure personal identification and authentication. However, the limited availability of high-quality fingerprint datasets and privacy concerns often restrict model training and evaluation. To address this, this project proposes a Generative Adversarial Network (GAN)-based approach for synthetic fingerprint image generation. The GAN model is trained using real fingerprint images to learn complex ridge patterns, textures, and minutiae features. Once trained, it generates realistic artificial fingerprints that closely resemble original ones in both structure and quality. A similarity analysis between the real and generated images is performed using feature extraction and matching techniques to evaluate authenticity and visual fidelity. Experimental results demonstrate that the generated AI fingerprints achieve high similarity scores with real samples, showing the potential of GANs in biometric data augmentation and secure system testing without compromising user privacy.*

I. INTRODUCTION

Biometric authentication plays a vital role in modern security systems, with fingerprints being one of the most reliable and widely used forms of identity verification. Every individual's fingerprint pattern is unique and remains unchanged throughout life, making it an ideal parameter for personal identification. However, with the growing need for large-scale biometric systems, acquiring, storing, and using real fingerprint data raises major challenges related to privacy, security, and data scarcity.

In recent years, **Artificial Intelligence (AI)** and particularly **Generative Adversarial Networks (GANs)** have emerged as powerful technologies capable of generating synthetic yet realistic fingerprint images. These AI-generated fingerprints can be used for system training, evaluation, and testing without compromising user privacy. The concept of **"GAN for Fingerprint Image Generation"** focuses on creating artificial fingerprints that closely resemble real ones in texture, ridge flow, and minutiae structure. By training a GAN model using real fingerprint data, the system learns to produce highly realistic synthetic fingerprints that can enhance dataset diversity and robustness in biometric applications.

The main aims to design and implement a **GAN-based fingerprint generator** that can generate lifelike fingerprints and then verify how closely these generated samples match real ones. Through comparative analysis and quality assessment, the demonstrates how GANs can contribute to secure, ethical, and data-efficient biometric system development.

1.1 Artificial Intelligence (AI) in Biometrics

Artificial Intelligence refers to the development of computer systems that can perform tasks requiring human-like intelligence such as learning, reasoning, and decision-making. In the field of biometrics, AI helps in automating the identification and verification process by analyzing biological and behavioral traits like fingerprints, facial features, or iris patterns. AI algorithms can recognize minute variations, identify patterns, and make decisions with remarkable precision.

AI is used to train a generative model (GAN) that learns the patterns of real fingerprints and generates synthetic ones that mimic authentic biometric data. This reduces dependency on real samples, ensuring enhanced privacy and scalability in biometric applications.

1.2 Computer Vision

Computer Vision is a field of Artificial Intelligence that enables machines to interpret and understand visual data from the real world, such as images and videos. It allows systems to analyze patterns, shapes, and textures, making it an essential component in fingerprint analysis and generation.

Computer vision techniques are applied to preprocess and analyze fingerprint images — identifying ridge flow, edges, and minutiae points that are crucial for generating realistic synthetic prints. The system leverages these visual patterns to ensure that the GAN learns accurate structural details during training.

1.3 Generative Adversarial Networks (GANs)

Generative Adversarial Networks are a class of deep learning models introduced by Ian Good fellow in 2014. A

GAN consists of two main components — a **Generator** and a **Discriminator**. The Generator creates synthetic images, while the Discriminator attempts to distinguish between real and fake ones. Through continuous adversarial learning, the generator improves its ability to produce highly realistic data that the discriminator can no longer differentiate from real samples.

In fingerprint image generation, GANs play a crucial role by learning the complex ridge structures and minutiae features from real data. The outcome is a set of synthetic fingerprint images that exhibit real-like characteristics and can be safely used for model training and testing.

1.4 Fingerprint Recognition and Matching

Fingerprint recognition is the process of identifying individuals based on the unique patterns of ridges and valleys on their fingertips. Traditional fingerprint recognition systems involve steps such as image acquisition, feature extraction, template creation, and matching.

once the synthetic fingerprints are generated by the GAN model, they are compared against real fingerprints to evaluate quality and similarity. Feature matching algorithms like **Minutiae Matching**, **Structural Similarity Index (SSIM)**, or **Cosine Similarity**are used to determine how closely the generated fingerprints resemble authentic ones.

1.5 Image Processing in Fingerprint Generation

Image processing plays a key role in improving fingerprint image quality and extracting relevant features. Preprocessing techniques such as noise removal, edge detection, and contrast enhancement are applied to both real and generated images to ensure clarity and uniformity.

Techniques like **Gabor filters**, **Histogram Equalization**, and **Binarization**are commonly used to highlight ridge details and minutiae structures. These processed images are then used as input for the GAN model, allowing it to learn detailed fingerprint patterns accurately.

1.6 Real-Time Fingerprint Image Comparison

The generated fingerprints are compared with real ones using automated systems that can perform real-time analysis. These systems apply image matching algorithms to calculate the degree of similarity.

Such real-time comparison is useful in evaluating GAN performance and can also be extended to biometric authentication scenarios, where instant fingerprint validation is crucial. This ensures that synthetic data generated by the GAN is realistic enough for real-world security applications.

1.7 Applications of Synthetic Fingerprints

Synthetic fingerprints have multiple practical applications. They can be used**to:**

Augment training datasets **for AI-based biometric systems.**
Evaluate and benchmark fingerprint recognition algorithms **without exposing real user data.**
Test security vulnerabilities in biometric systems by simulating various fingerprint variations.
Enhance privacy by using synthetic instead of real biometric data during research and system development.
Overall, GAN-generated fingerprints support ethical data handling while improving the efficiency and robustness of biometric technology.

1.8 Merits and Demerits

Merits

Privacy Preservation: Reduces the need for collecting sensitive biometric data.
Cost Efficiency: Generates large datasets without expensive data collection.
Scalability: Easily expands dataset size to train deep learning models effectively.
Realism: Produces high-quality, authentic-looking fingerprints useful for research and testing.

1.9 Future Scope

The future of fingerprint generation using GANs is promising. Advanced architectures such as **StyleGAN**, **CycleGAN**, and **Diffusion Models** can further improve image realism, texture consistency, and diversity. Combining GANs with **feature-based learning** and **biometric encryption** can enhance both security and performance.

In addition, the integration of **explainable AI (XAI)** can help understand how GANs generate specific ridge patterns, leading to more transparent and controllable model behavior. Future research can also explore generating **3D fingerprints** for high-end security systems, expanding the scope of AI-driven biometrics.

II. LITERATURE SURVEY

2.1 Finger Recovery Transformer: Toward Better Incomplete Fingerprint Identification-2024

Fingerprint recognition is a crucial biometric technology extensively used in identity verification, including areas like criminal investigations, security systems, and biometric authentication. This technology encounters greater challenges when dealing with incomplete fingerprint images, especially those with significant background noise or substantial portions of the fingerprint missing. Existing incomplete fingerprint recognition technologies struggle with extensive data loss, primarily due to the significant reduction and difficulty in extracting usable features from incomplete fingerprint images. Current image processing methods or deep learning models are unable to comprehensively reconstruct fingerprint features with limited information. To address these challenges, we introduce the Finger Recovery Transformer (FingerRT), an innovative network specifically designed for recovering incomplete fingerprint information. FingerRT can simultaneously complete ambient noise cancellation and fingerprint feature information recovery, resulting in a complete and clean fingerprint image. FingerRT combines the most critical feature information in fingerprints, directional field, and minutiae, as supervision information. FingerRT inherits the denoising ability of the fingerprint enhancement networks and the powerful generative ability of the Vision Transformer architecture, enabling high-quality and robust fingerprint information recovery. By imposing constraints at multiple levels, including fingerprint features, fingerprint images, and multi-stage generation, FingerRT can complete fingerprint information accurately and effectively. Experiments demonstrate that FingerRT significantly enhances fingerprint recognition accuracy after recovery across various fingerprint datasets, including rolled, snapped, and latent fingerprints.

2.2 Touch of Privacy: A Homomorphic Encryption-Powered Deep Learning Framework for Fingerprint Authentication-2025

Deep learning and fully homomorphic encryption (FHE) are integrated for privacy-preserving fingerprint recognition. Convolutional neural network (CNN) extract fingerprint features encrypted using the Cheon-Kim-Kim-Song (CKKS) FHE scheme. TenSEAL ensures all computations occur in the encrypted domain, preventing raw biometric data exposure during authentication. A subset of the SOCOFing dataset, comprising 7,200 altered fingerprints from 90 individuals across three difficulty levels, is used for training (80%), validation (10%), and testing (10%). One real fingerprint per user is encrypted and stored for authentication, reducing computational complexity. The CNN classifies encrypted features without decryption, ensuring secure authentication. The system, utilizing Euclidean similarity, achieves 99.06% test accuracy with a loss of 0.1692, a True Acceptance Rate (TAR) of 99.19%, a False Rejection Rate (FRR) of 0.81%, a False Acceptance Rate (FAR) of 0%, and a True Rejection Rate (TRR) of 100%, with a minimal Equal Error Rate (EER) of 0.40%. Encrypted templates require only 31 MB for 90 users, averaging 344 KB per fingerprint per user. Despite a 157.26× encryption overhead, remains feasible for real-time applications, completing an encrypted comparison in 0.025 seconds with a total processing time of 0.136 seconds per fingerprint. Adhering to ISO/IEC 24745 biometric protection standards, the system ensures irreversibility, unlinkability, and renewability of biometric templates. Decryption occurs only on the client side, keeping raw fingerprint data inaccessible to the server. The results confirm the feasibility of FHE for secure fingerprint authentication in cloud-based environments, benefiting applications in healthcare, banking, and access control.

2.3. Improved RF Fingerprint-Based Identity Verification in the Presence of an SEI Mimicking Adversary-2024

Specific Emitter Identification (SEI) is advantageous for its ability to passively identify emitters by exploiting distinct, unique, and organic features unintentionally imparted upon every signal during formation and transmission. These features are attributed to the slight variations and imperfections in the Radio Frequency (RF) front end; thus, SEI is being proposed as a physical layer security technique. Most SEI work assumes the targeted emitter is a passive source with immutable and difficult-to-mimic signal features. However, Software-Defined Radio (SDR) proliferation and Deep Learning (DL) advancements require a reassessment of these assumptions because DL can learn SEI features directly from an emitter's signals, and SDR enables signal manipulation. The investigates a strong adversary that uses SDR and DL to mimic an authorized emitter's signal features to circumvent SEI-based identity verification. The investigation considers three SEI mimicry approaches, two different SDR platforms, the application of matched filtering before SEI feature extraction, and selecting the most informative portions of the signals' time-frequency representation using entropy. The results show that “off-the-shelf” DL achieves effective SEI mimicry. Additionally, SDR constraints impact SEI mimicry effectiveness and suggest an adversary's minimum requirements. Our results show matched filtering results in the identity of all authorized emitters being correctly verified at a rate of 90% or higher, the rejection of all other authorized emitters-whose IDs are not being verified-at a rate of 97% or higher, and rejection of forty-five out of fortyeight SEI mimicry attacks. Based on the results presented herein, future SEI research must consider adversaries capable of mimicking another emitter's SEI features or manipulating their own.

III. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In existing biometric systems, fingerprint recognition primarily relies on large datasets of real fingerprint images collected from users. These fingerprints are used to train and test models for authentication and identification. Traditional systems focus on feature extraction techniques such as ridge detection, minutiae matching, and pattern classification to verify user identity.

However, the availability of high-quality fingerprint datasets is limited due to privacy concerns, data collection difficulties, and the risk of identity misuse. Existing image generation methods, such as data augmentation through rotation, scaling, and transformation, fail to produce entirely new and realistic fingerprints. As a result, model performance and dataset diversity remain restricted.

Furthermore, older generation fingerprint systems depend heavily on manual preprocessing and lack the ability to create synthetic biometric data that preserves the statistical features of real fingerprints. This limits research and testing in the field of biometric security, making it difficult to evaluate algorithms effectively without compromising sensitive information.

3.2 PROPOSED SYSTEM

The proposed system introduces a **Generative Adversarial Network (GAN)-based approach** for fingerprint image generation. The system uses real fingerprint images to train a GAN model that learns the structural patterns, textures, and ridge flows of authentic fingerprints. Once trained, the generator produces synthetic fingerprint images that are visually and statistically similar to real ones.

This system overcomes the limitations of traditional methods by creating **artificial fingerprints** that preserve the natural characteristics of real samples while ensuring data privacy. The generated fingerprints can be used to **augment training datasets**, **test biometric algorithms**, and **analyze model performance** without using real personal data.

Additionally, a comparison module is included to evaluate the similarity between real and generated images using various metrics such as **Structural Similarity Index (SSIM)**, **Minutiae Matching**, and **Cosine Similarity**. This helps in validating the quality and authenticity of the generated fingerprints.

The proposed system provides a scalable, privacy-preserving, and cost-effective solution for biometric data generation and testing.

IV. SOFTWARE DESCRIPTION

The **GAN for Fingerprint Image Generation** system relies on a collection of software tools, frameworks, and libraries that enable deep learning, image generation, feature extraction, and data visualization. The integration of these tools ensures that the system can effectively learn from real fingerprint data and generate realistic synthetic fingerprints while maintaining privacy and accuracy.
Below is a detailed description of the major software components used in this system.

4.1 Machine Learning Frameworks

Machine learning (ML) forms the foundation, as it enables the system to learn intricate fingerprint features such as ridge patterns, minutiae points, and texture variations from real images.

TensorFlow

TensorFlow is an open-source deep learning framework developed by Google. It is used to design, train, and deploy the GAN architecture efficiently. TensorFlow handles both the **Generator** and **Discriminator** networks, enabling stable model training and gradient updates. It provides tools for GPU acceleration, which speeds up the training process on large datasets.

PyTorch

PyTorch, developed by Facebook AI Research, is another popular deep learning framework used for flexible model building and experimentation. It supports **dynamic computation graphs**, making it ideal for testing different GAN architectures and custom loss functions. PyTorch's modular structure allows seamless integration with image processing libraries like OpenCV.

Keras

Keras serves as a high-level neural network API that simplifies model development. It runs on top of TensorFlow and is used for building and training GAN layers, defining activation functions, and managing optimizers. Its simplicity and readability make it perfect for rapid prototyping of deep learning models.

4.2 Image Processing and Computer Vision Libraries

Image processing is essential for analyzing real fingerprint images and evaluating generated outputs. These libraries help enhance image quality, extract features, and visualize ridge details.

OpenCV (Open Source Computer Vision Library)

OpenCV is a versatile library used for real-time image processing and computer vision tasks. it is used for preprocessing fingerprint images—such as resizing, noise reduction, grayscale conversion, and edge detection. OpenCV also assists in comparing structural similarities between real and AI-generated fingerprints.

Scikit-Image

Scikit-Image is used for applying advanced image transformation techniques like thresholding, skeletonization, and feature extraction. It helps analyze fingerprint ridge structures and minutiae points for quality evaluation of generated samples.

Pillow (PIL)

The Pillow library is used for basic image operations such as loading, cropping, converting, and saving fingerprint images. It supports multiple file formats and is efficient for preprocessing large datasets.

4.3 GAN Framework and Architecture

Generative Adversarial Networks (GANs) form the core of this system. The framework consists of two neural networks that work in opposition to generate highly realistic synthetic data.

Generator

The Generator network learns to produce artificial fingerprint images from random noise. It gradually improves by learning key fingerprint characteristics such as ridge orientation, spacing, and minutiae distribution.

Discriminator

The Discriminator acts as a classifier that distinguishes between real and generated fingerprints. Through adversarial training, it helps the Generator produce images that become increasingly indistinguishable from real samples.

Loss Functions

Custom loss functions such as **Binary Cross-Entropy** and **Adversarial Loss**are used to optimize training and maintain a balance between the Generator and Discriminator.

4.4 Feature Extraction and Image Matching Tools

To compare the authenticity of generated fingerprints, the system uses image similarity metrics and feature extraction techniques.

NumPy and SciPy

These Python libraries are used for mathematical and numerical operations. They handle large multidimensional arrays, compute similarity metrics, and perform transformations on image matrices.

Structural Similarity Index (SSIM)

SSIM is used to measure how closely the generated fingerprint matches the real image in terms of structure, luminance, and contrast.

Minutiae Extraction Algorithms

Minutiae-based comparison techniques are applied to detect unique ridge endpoints and bifurcations in both real and generated images, helping evaluate fingerprint realism.

V. SYSTEM DESIGN

System design defines the overall structure and workflow of the proposed model. It illustrates how different components such as data collection, preprocessing, GAN model training, image generation, and evaluation interact with one another to produce realistic fingerprint images. The design ensures that the system operates efficiently and achieves the desired objectives with accuracy and privacy.

5.1 SYSTEM ARCHITECTURE

The **system architecture** of the GAN-based fingerprint image generation model consists of several interconnected modules — data acquisition, preprocessing, GAN model training, image generation, and image similarity analysis. Each module performs a specific function that contributes to the generation of realistic synthetic fingerprints.

Description of System Architecture

Input Data (Real Fingerprint Dataset)

The system starts with a collection of real fingerprint images obtained from open-source biometric datasets. These serve as the training data for the GAN model.

Data Preprocessing

Input images are resized, converted to grayscale, and normalized. Noise removal and enhancement techniques are applied using OpenCV to ensure that the data is clean and consistent for training.

GAN Model Training

The**Generator** and **Discriminator** networks are trained simultaneously.
The *Generator* creates synthetic fingerprint images from random noise.
The *Discriminator* evaluates the images and provides feedback to the Generator to improve image realism.

Generated Fingerprint Images

Once trained, the Generator produces high-quality artificial fingerprint images that closely resemble real fingerprints in structure and texture.

Feature Extraction and Similarity Check

Generated and real fingerprints are compared using metrics such as **Structural Similarity Index (SSIM)** and **Minutiae Matching** to determine how closely the generated images match authentic samples.

Output Visualization

The results, including the generated fingerprint and similarity scores, are displayed through a Flask-based web interface or notebook environment.

5.2 DATA FLOW DIAGRAM (DFD)

The **Data Flow Diagram (DFD)** illustrates how data moves through the system. It shows the major processes, data inputs, outputs, and data stores that make up the GAN fingerprint generation system.

LEVEL 0 DFD (Context Diagram)

At **Level 0**, the system is represented as a single process that interacts with external entities such as the user and the fingerprint dataset.
Description:

User

Uploads fingerprint data or initiates the generation process.
Requests similarity analysis between real and generated images.

Fingerprint Dataset

Provides real fingerprint samples as input to the system.

GAN System (Main Process)

Accepts input data from the user or dataset.
Processes the data using the GAN model.
Generates synthetic fingerprint images.
Performs similarity analysis and sends results back to the user.

VI. IMPLEMENTATION

Implementation is the most critical stage of the development process, where the theoretical design is transformed into a working model.
The proposed **GAN-based Fingerprint Image Generation System**is implemented using **Python** as the core programming language, along with **Flask** for the backend framework, **TensorFlow/PyTorch** for model development, and **OpenCV** for image processing.

The main goal of the implementation is to train a **Generative Adversarial Network (GAN)** using real fingerprint images to generate artificial fingerprints that resemble real ones and validate their authenticity through similarity analysis.

6.1 OVERVIEW OF THE COMPONENTS

The system consists of two main parts:

Backend (Server-Side):

Handles dataset loading, image preprocessing, GAN model training, fingerprint generation, and image similarity analysis. It uses frameworks like TensorFlow, Keras, and Flask to manage computations and serve generated results.

Frontend (Client-Side):

Provides a simple and interactive web interface that allows users to upload fingerprint samples, view generated synthetic fingerprints, and see comparison results or similarity scores in real time.

6.2 BACKEND: GAN IMPLEMENTATION

The backend is the core of the system, responsible for performing model training, image generation, and evaluation. The backend integrates **Flask** with deep learning frameworks to handle data flow, model execution, and user interaction.

Flask Setup

A Flask web application is created to serve as the main server.
It provides endpoints that allow users to upload real fingerprint images and request AI-generated images.
The backend processes each image, passes it through the **trained GAN model**, and returns the generated fingerprint image to the frontend.
The server also computes similarity metrics between the real and generated fingerprints and displays the results.

GAN Model Implementation

The **Generative Adversarial Network (GAN)**is implemented using **TensorFlow** and **Keras**:
Generator Network: Takes random noise as input and produces synthetic fingerprint images.
Discriminator Network: Differentiates between real and generated fingerprints.
During training, both networks compete against each other until the generated fingerprints become indistinguishable from real ones.

Model Training Process

Load the real fingerprint dataset.
Preprocess images (resize, grayscale, normalize).
Train Generator and Discriminator iteratively.
Save model weights once optimal performance is achieved.
Generate and store synthetic fingerprints.

6.3 IMAGE PREPROCESSING AND FEATURE EXTRACTION

Before training the GAN, all fingerprint images are preprocessed to ensure consistency and quality.

Preprocessing Steps

Resizing: All images are resized to a fixed resolution (e.g., 128×128 pixels).
Grayscale Conversion: Fingerprints are converted to grayscale for simplicity.
Noise Removal: Gaussian filters and thresholdingare applied to remove background noise.
Normalization: Pixel values are normalized between 0 and 1 for better convergence.

Feature Extraction

After generation, fingerprint features such as ridge patterns, bifurcations, and minutiae points are extracted using **OpenCV** and **Scikit-Image**.
These features are used to compare real and synthetic fingerprints.

6.4 FRONTEND: USER INTERFACE

The frontend provides a simple web-based interface that connects to the Flask backend. Users can interact with the model without any coding knowledge.

Key Features

File Upload Interface: Allows users to upload real fingerprint images as input.

Generated Image Display: Displays the AI-generated fingerprint created by the GAN model.
Comparison Results: Shows similarity scores (e.g., SSIM or cosine similarity) between the original and generated fingerprints.

6.5 SIMILARITY ANALYSIS AND VALIDATION

The main purpose of the system is not just to generate fingerprints but also to verify how closely they resemble real fingerprints.

Similarity Checking Methods

Structural Similarity Index (SSIM): Measures the similarity in luminance, structure, and contrast between two fingerprint images.
Minutiae Point Comparison: Compares ridge endings and bifurcations between real and generated images.
Cosine Similarity: Measures the feature vector similarity between original and generated fingerprints.

Validation Process

Both real and generated fingerprints are passed through the feature extractor.
Similarity metrics are calculated.
Results are visualized as a score or percentage showing the degree of match between real and AI-generated images.

6.6 REAL-TIME EXECUTION AND RESULTS

The Flask server enables real-time fingerprint generation and comparison through a web interface:
The user uploads a real fingerprint.
The backend immediately processes it and generates a synthetic version.
The system displays both images side-by-side with similarity results.
The process is quick and efficient due to GPU acceleration using **CUDA**.

6.7 BENEFITS OF THE PROPOSED SYSTEM

Privacy Preservation:
Eliminates the need to use real biometric data for testing or training, reducing privacy risks.
Dataset Augmentation:
Helps create large-scale fingerprint datasets for research without manual collection.
Improved Model Training:
Provides diverse synthetic data for building more robust fingerprint recognition systems.
Realistic Generation:
Produces fingerprints that closely mimic the structure and ridge flow of real samples.
Research and Testing Support:
Facilitates testing of biometric algorithms in a controlled and ethical manner.
Cost Efficiency:
Reduces data collection costs and resource dependency for dataset acquisition.

VII. CONCLUSION

The *"Fingerprint Image Generation using GAN"* successfully demonstrates the potential of Generative Adversarial Networks (GANs) in producing realistic synthetic fingerprint images. By training the GAN model with real fingerprint datasets, the system learns complex texture patterns, ridge structures, and minutiae features unique to human fingerprints. The generated images closely resemble real ones, providing a valuable resource for research, biometric testing, and privacy-preserving data generation.

The bridges the gap between data scarcity and the need for secure biometric systems by enabling artificial fingerprint synthesis without compromising real user data. The GAN-based model enhances the diversity and quality of fingerprint datasets, improving the robustness of biometric recognition systems. The results show that the generated fingerprints exhibit high similarity (measured using SSIM and other comparison metrics) with authentic prints, validating the efficiency of the proposed approach.

In summary, the highlights how deep learning and GAN architectures can revolutionize biometric image generation, ensuring both accuracy and privacy. It also demonstrates how AI-based synthetic data generation can reduce dependency on sensitive personal data while maintaining system performance in training and testing environments.

APPENDIXES

SOURCE CODE:

```
importos
discriminator.save_weights(disc_path)
print("Saved weights", gen_path)
# sample
sample_and_save(generator, epoch+1)
defsample_and_save(generator, epoch, n=9,
latent_dim=LATENT_DIM):
noise = tf.random.normal([n, latent_dim])
imgs = generator(noise, training=False).numpy()
imgs = (imgs * 255).astype('uint8')
fig, axes = plt.subplots(1, n, figsize=(n * 2, 2))
fori, ax in enumerate(axes):
ax.imshow(imgs[i].squeeze(), cmap='gray')
ax.axis('off')
plt.suptitle(f'Samples @ epoch {epoch}')
out = SAMPLES_DIR / f'sample_epoch_{epoch}.png'
plt.savefig(out)
plt.close(fig)
print('Saved sample to', out)
# ---------------------- Similarity & simple minutiae-like check -
---------------------
defcompute_ssim(a, b):
# a, b are float images in [0,1], shape HWC
a2 = (a.squeeze() * 255).astype('uint8')
b2 = (b.squeeze() * 255).astype('uint8')
score = ssim(a2, b2)
return float(score)
defsimple_skeleton_minutiae_score(img):
# INPUT: img float [0,1], grayscale
```

```
# Quick pipeline: threshold ->skeletonize -> count branchpoints& endpoints
arr = util.img_as_ubyte(img.squeeze())
thresh = filters.threshold_otsu(arr)
bw = arr> thresh
skel = morphology.skeletonize(bw)
# hit-or-miss for endpoints and branchpoints (approx)
# convolution to count neighbors
kernel = np.ones((3, 3), dtype=int)
neigh = cv2.filter2D(skel.astype('uint8'), -1, kernel)
endpoints = np.sum((skel == 1) & (neigh == 2)) # endpoint has self + 1 neighbor -> 2
branchpoints = np.sum((skel == 1) & (neigh >= 4))
return {'endpoints': int(endpoints), 'branchpoints': int(branchpoints)}
defcompare_minutiae_like(a, b):
sa = simple_skeleton_minutiae_score(a)
sb = simple_skeleton_minutiae_score(b)
# score is inverse relative difference
ep_score = 1.0 - abs(sa['endpoints'] - sb['endpoints']) / max(1, (sa['endpoints'] + sb['endpoints'])/2)
bp_score = 1.0 - abs(sa['branchpoints'] - sb['branchpoints']) / max(1, (sa['branchpoints'] + sb['branchpoints'])/2)
ep_score = max(0.0, ep_score)
bp_score = max(0.0, bp_score)
return float((ep_score + bp_score) / 2.0)
```

VIII. ACKNOWLEDGEMENT

First and foremost, we thank the almighty for giving me talents and opportunity to complete my project and my family for unwavering support.

I would like to express my sincere heartfelt thanks to our college chief Secretary, Er. SURIYA SUBRAMANIAM, for providing facility to getting more ideas related to this project work.

I would like to express my sincere gratitude to our college chief chairperson, Dr. SUJATHA SUBRAMANIAM, Ph.D., for providing large facilities in the institution for the completion of the project work.

I would like to express my sincere thanks and gratitude to our college principal, Dr. D.LOGANATHAN, M.Sc (Tech)., Ph.D., for his encouragement to do this innovative project.

I wish to express heartfelt thanks and sincere gratitude to our Head of the Department Mr.KALAIYARASAN.K, M.Tech.,(Ph.D)for her valuable advice to complete this work successfully and also for her enthusiastic encouragement to make my effort worthwhile and fruitful.

I am extremely indebted to my Project SupervisorMr. NAVEEN RAJ J, M.E.,Assistant Professor, Department of Computer Science and Engineering, who extended his complete guidance and valuable advice which helped me to take further step into depth of my project.

I also like to thank all the Faculty Members of Department of Computer Science and Engineering for their support and encouragement. I wish to thank all the non-teaching staff members of our Department, all our friends and well-wishers who are all behind the success of my project.

REFERENCES

[1] Yap, W. S., & Wong, Y. D. (2015). Intelligent transportation systems: a review. IET Intelligent Transport Systems, 9(2), 182-191.

[2] Amini, A., Amini, A., &Ghaemi, Z. (2018). A review on the application of geographic information systems (GIS) in disaster management. Geoenvironmental Disasters, 5(1), 11.

[3] Viti, F., Colajanni, M., & Romano, S. P. (2013). An ITS architecture for emergency management. Computer Communications, 36(4), 433-444.

[4] Zeadally, S., Hunt, R., Chen, Y. S., & Irwin, A. (2012). Vehicular ad hoc networks (VANETs): status, results, and challenges. Telecommunication Systems, 50(4), 217- 241.

[5] Khattak, A. M., Jabbar, S., Naeem, M., & Khan, A. A. (2016). Real-time data communication in vehicular adhoc networks: a review. Wireless Networks, 22(2), 395- 411.

[6] Talukder, J., &Hakak, S. (2018). A human-centric design approach for disaster management information systems.

[7] N. Lukas, Y. Zhang, and F. Kerschbaum, Deep neural network fingerprinting by conferrable adversarial examples, in *Proc. 9th Int. Conf. LearningRepresentations*,VirtualEvent, https://openreview.net/forum?id=VqzVhqxkjH1, 2021.

[8] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 770–778.

[9] C. Xiao, B. Li, J. Y. Zhu, W. He, M. Liu, and D. Song, Generating adversarial examples with adversarial networks, in *Proc. 27th Int. Joint Conf. Artificial Intelligence*, Stockholm, Sweden, 2018, pp. 3905–3911.

[10] Cappelli, R., A. Erol, D. Maio, and D. Maltoni, Synthetic fingerprint-image generation, *Proc. 15th International Conference on Pattern Recognition (ICPR2000)*, Barcelona, Vol. 3, pp. 475–478, Sept. 2000.

[11] Hill, C.J., Risk of masquerade arising from the storage of biometrics, B.Sc. thesis, Department of Computer Science, Australian National University, Nov. 2001..