##### A Project report on

**“Deep Learning Based Text-to-Image Synthesis for Criminal Face Generation”**

Submitted in partial fulfillment of the requirement for the award of the degree of

#### BACHELOR OF TECHNOLOGY

**in**

#### COMPUTER SCIENCE AND ENGINEERING

##### By

**S.ROSHAN JAMEER BASHA 21701A05E8**

**N.SNEHA 21701A05I2**

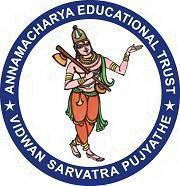
**SHAIK SAMEER 21701A05G1**

**N.ROSE MARY 21701A05E7**

*Under the esteemed guidance of*

**Mr. M.SREENIVASULU M.Tech**

Assistant Professor in CSE, AITS.



Submitted to

**Department of Computer Science and Engineering Annamacharya Institute of Technology and Sciences** (An Autonomous Institution)

(Approved by AICTE, New-Delhi and affiliated to J.N.T.U, Anantapur) (Accredited by NBA &NAAC)

New Boyanapalli, Rajampet, Annamaiah (Dt), A.P-516 126

**2024-2025**

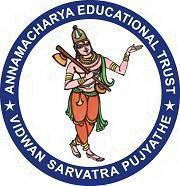
**Department of Computer Science and Engineering**

**Annamacharya Institute of Technology and Sciences**

(An Autonomous Institution)

(Approved by AICTE, New-Delhi and affiliated to J.N.T.U, Anantapur) (Accredited by NBA & NAAC)

New Boyanapalli, Rajampet ,Annamaiah (Dt), A.P-516 126



**CERTIFICATE**

This is to certify that the project report entitled **“Deep Learning Based Text-to-Image Synthesis for Criminal Face Generation”**

is submitted by

**S.ROSHAN JAMEER BASHA 21701A05E8**

**N.SNEHA 21701A05I2**

**SHAIK SAMEER 21701A05G1**

**N.ROSE MARY 21701A05E7**

in partial fulfillment of the requirements for the award of Degree of **Bachelor of Technology**

in “**Computer Science and Engineering**” for the academic year 2024-25.

Signature of Guide: Signature of HOD: Mr.M.SREENIVASULU,M.Tech,(Ph.D). Dr. M. Subba Rao Ph.D.,

Assistant Professor in CSE, Professor & Head, Dept. of CSE,

AITS,Rajampet. Dean of Student Affairs, AITS, Rajampet.

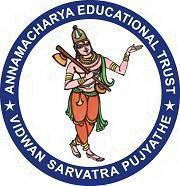
**Department of Computer Science and Engineering**

**Annamacharya Institute of Technology and Sciences**

(An Autonomous Institution)

(Approved by AICTE, New-Delhi and affiliated to J.N.T.U, Anantapur) (Accredited by NBA & NAAC)

New Boyanapalli, Rajampet ,Annamaiah (Dt), A.P-516 126



**CERTIFICATE**

This is to certify that the project report entitled **“Deep Learning Based Text-to-Image Synthesis for Criminal Face Generation”**

is submitted by

**S.ROSHAN JAMEER BASHA 21701A05E8**

**N.SNEHA 21701A05I2**

**SHAIK SAMEER 21701A05G1**

**N.ROSE MARY 21701A05E7**

in partial fulfillment of the requirements for the award of Degree of **Bachelor of Technology** in “**Computer Science and Engineering**” is a record of bonafide work carried out by her during the academic year 2024-25.

Project viva-voce held on :

Internal Examiner External Examiner

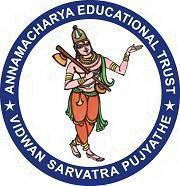
**Department of Computer Science and Engineering**

**Annamacharya Institute of Technology and Sciences**

(An Autonomous Institution)

(Approved by AICTE, New-Delhi and affiliated to J.N.T.U, Anantapur) (Accredited by NBA & NAAC)

New Boyanapalli, Rajampet ,Annamaiah (Dt), A.P-516 126



**ANTI-PLAGIARISM CERTIFICATE**

This is to certify that the project report entitled **“Deep Learning Based Text-to-Image Synthesis for Criminal Face Generation”**

is submitted by

**S.ROSHAN JAMEER BASHA 21701A05E8**

**N.SNEHA 21701A05I2**

**SHAIK SAMEER 21701A05G1**

**N.ROSE MARY 21701A05E7**

in partial fulfillment of the requirements for the award of Degree of **Bachelor of Technology** in “**Computer Science and Engineering**”. Course contains the plagiarism of % which is within the acceptable limits.

Date: Dean / Coordinator

**Research & Development Cell Date:**

**DECLARATION**

We hereby declare that the project report entitled**“Deep Learning Based Text-to-Image Synthesis for Criminal Face Generation”** under the guidanceof **Mr.M.SREENIVASULU M.Tech Assistant Professor in CSE**, Department of Computer Science and Engineering is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

This is a record of bonafide work carried out by me/us and the results embodied in this project report have not been reproduced or copied from any source. The results embodied in this project report have been submitted to any other University or institute for the Award of any other Degree or Diploma.

#### PROJECT ASSOSIATES

**S.ROSHAN JAMEER BASHA**

**N.SNEHA**

**SHAIK SAMEER N.ROSE MARY**

***ACKNOWLEDGMENT***

*We endeavor of a long period can be successful only with the advice of many well- wishers. We take this opportunity to express my deep gratitude and appreciation to all those who encouraged me for the successful completion of the project work.*

*Our heartfelt thanks to our Guide,* ***Mr.M.Sreenivasulu*** *M.Tech Assistant Professor in Department of Computer Science and Engineering, Annamacharya Institute of Technology and Sciences, Rajampet, for his valuable guidance and suggestions in analyzing and testing throughout the period, till the end of the project work completion.*

*We wish to express sincere thanks and gratitude to* ***Dr. M. Subba Rao****, Head of the Department of Computer Science and Engineering, for his encouragement and facilities that were offered to us for carrying out this project.*

*We take this opportunity to offer gratefulness to our Principal* ***Dr. S.M.V. Narayana****, for providing all sorts of help during the project work.*

*We are very much thankful to* ***Sri. C. Gangi Reddy****, Honorary Secretary of the Annamacharya Educational Trust, for his help in providing good facilities in our college.*

*We would express our sincere thanks to all faculty members of* ***Computer Science and Engineering Department, batch-mates, friends*** *and* ***lab-technicians****, who have helped us to complete the project work successfully.*

*Finally, we express our sincere thanks to* ***our parents*** *who has provided their heartfelt support and encouragement in the accomplishment to complete this project successfully.*

#### PROJECT ASSOSIATES

*S.ROSHAN JAMEER BASHA*

*N.SNEHA*

*SHAIK SAMEER*

*N.ROSE MARY*

|  |  |
| --- | --- |
| **CONTENTS** |  |
| **CHAPTER** | **PAGE NO** |
| **ABSTRACT** |  |
| **1. INTRODUCTION** | 1-4 |
| **2. LITERATURE SURVEY** | 5-15 |
| **3. SYSTEM ANALYSIS** | 16-24 |
| 3.1. Existing System | 16 |
| 3.1.1. Disadvantages | 16 |
| 3.2. Proposed System | 16 |
| 3.2.1. Advantages | 16 |
| 3.3. Modules in Proposed System | 17 |
| 3.3.1. System | 17 |
| 3.3.2. User | 17 |
| 3.3.3. Algorithms Used | 18-23 |
| 3.3.3.1. Deep Learning Algortithms | 18-23 |
| 3.4. Performance Evaluation and Predicting Result | 23-24 |
| **4. SYSTEM REQUIREMENTS SPECIFICATIONS** | 25-27 |
| 4.1. Software Requirements | 25 |
| 4.2. Hardware Requirements | 25 |
| 4.3. Feasibility Study | 25-26 |
| 4.3.1. Economic Feasibility | 25 |
| 4.3.2. Technical Feasibility | 26 |
| 4.3.3. Behavioral Feasibility | 26 |
| 4.3.4. Benefits of Doing Feasibility Study | 26 |
| 4.4. Functional and Non-Functional Requirements | 26-27 |
| 4.4.1. Functional Requirements | 27 |
| 4.4.2. Non-Functional Requirements | 27 |
| **5. SYSTEM DESIGN** | 28-40 |

* 1. [Architecture Design 28](#_TOC_250031)
  2. [Introduction to UML Diagrams 28-29](#_TOC_250030)
     1. [Goals 29](#_TOC_250029)
  3. [UML Notations 29-30](#_TOC_250028)
  4. [UML Diagrams 31-38](#_TOC_250027)
     1. [Use Case Diagram 31](#_TOC_250026)
     2. [Class Diagram 32](#_TOC_250025)
     3. [Sequence Diagram 33](#_TOC_250024)
     4. [Collaboration Diagram 34](#_TOC_250023)
     5. [Deployment Diagram 35](#_TOC_250022)
     6. [Activity Diagram 36](#_TOC_250021)
     7. [Component Diagram 37](#_TOC_250020)
     8. [State Chart Diagram 38](#_TOC_250019)
  5. [ER Diagram 39](#_TOC_250018)
  6. Data Flow Diagrams 39-40
     1. [Context Level Diagram 40](#_TOC_250017)
     2. [Level 1 Diagram 40](#_TOC_250016)

1. [SYSTEM CODING AND IMPLEMENTATION 41-57](#_TOC_250015)
   1. [Introduction to Python Programming Language 41-44](#_TOC_250014)
      1. [Benefits of Python 43-44](#_TOC_250013)
   2. Python Libraries Used in Python 45
   3. [Sample Code 46-57](#_TOC_250012)
2. [SYSTEM TESTING 58-63](#_TOC_250011)
   1. [Software Testing Techniques 58-59](#_TOC_250010)
      1. [Testing Objectives 58](#_TOC_250009)
      2. [Test Case Design 58](#_TOC_250008)
      3. [White Box Testing 58](#_TOC_250007)
      4. [Black Box Testing 59](#_TOC_250006)
   2. Software Testing Strategies 60-63
      1. [Unit Testing 60](#_TOC_250005)
      2. [Integration Testing 60-61](#_TOC_250004)
      3. [Validation Testing 61](#_TOC_250003)
      4. [System Testing 62](#_TOC_250002)
      5. [Security Testing 62](#_TOC_250001)
      6. Performance Testing 62-63
3. [RESULTS 64-69](#_TOC_250000)
4. CONCLUSION AND FUTURE ENHANCEMENTS 70

**BILIOGRAPHY PLAGARISM REPORT JOURNAL PUBLICATION**

## LIST OF FIGURES & TABLES

|  |  |  |
| --- | --- | --- |
| **Fig. No.** | **Figures** | **Page No.** |
| 1.1. |  | 2 |
| 3.1. | Block diagram of Proposed Method | 17 |
| 3.2. |  | 18 |
| 3.3 |  | 19 |
| 5.1. | Architecture Diagram | 28 |
| 5.2. | Use Case Diagram | 31 |
| 5.3. | Class Diagram | 32 |
| 5.4. | Sequence Diagram | 33 |
| 5.5. | Collaboration Diagram | 34 |
| 5.6. | Deployment Diagram | 35 |
| 5.7. | Activity Diagram | 36 |
| 5.8. | Component Diagram | 37 |
| 5.9. | State Chart Diagram | 38 |
| 5.10. | ER Diagram | 39 |
| 5.11. | Context Level Diagram | 40 |
| 5.12. | Level 1 Diagram | 40 |
| 6.1. | Working of Python Program | 42 |
| 6.2. | Implementation of Python Program | 43 |
| 8.1. |  | 64 |
| 8.2. |  | 65 |
| 8.3. |  | 66 |
| 8.4. |  | 66 |
| 8.5. |  | 67 |
| 8.6. |  | 67 |
| 8.7. |  | 68 |
| 8.8. |  | 69 |
| 8.9 |  | 69 |
| 8.10 |  | 70 |

**ABSTRACT**

The rapid advancement of AI technology has brought both opportunities and challenges in the domain of law enforcement, particularly in suspect identification and criminal profiling. Traditional approaches rely heavily on witness sketches or verbal descriptions, which often lack accuracy and detail, leading to limited success in identifying suspects. Additionally, manual sketch generation is time-consuming and may not accurately depict the actual suspect, hindering investigations. The inability to produce realistic facial images from textual descriptions creates a significant gap in modern-day criminal investigations, slowing down suspect identification and reducing the chances of solving criminal cases efficiently.

To address this challenge, this study proposes a deep learning-based approach using Generative Adversarial Networks (GANs) to generate lifelike facial images from textual descriptions provided by witnesses or investigators. The proposed model comprises a generator that creates realistic face images and a discriminator that ensures authenticity by distinguishing real from generated images. A celebrity and AI-generated face dataset was used for training, ensuring high-quality outputs. The model achieved a generator loss of 6.9430 and a discriminator loss of 0.0545, demonstrating its capability to generate convincing facial images. Furthermore, a Flask-based interactive interface was developed, allowing law enforcement agencies to input textual descriptions and obtain realistic facial representations of potential suspects, enhancing criminal investigations and improving identification processes.

***Keywords:*** *Criminal Profiling, Deep Learning, Face Generation, Flask Interface, GAN Model, Law Enforcement, Suspect Identification, Text-to-Image.*

# CHAPTER-1 INTRODUCTION

## INTRODUCTION

The increasing reliance on artificial intelligence (AI) in various domains has significantly transformed the approach toward solving complex real-world problems. One such critical domain is law enforcement, where AI has the potential to revolutionize suspect identification and criminal profiling. Traditionally, the process of identifying suspects relied heavily on eyewitness descriptions or manual sketching by forensic artists, which often leads to subjective and inaccurate visual representations. This outdated method has posed significant challenges for law enforcement agencies in accurately identifying and apprehending suspects, as the final sketch may not always represent the actual appearance of the criminal. Moreover, it is time-consuming and prone to human errors, further hindering the investigation process. With the advancement in deep learning techniques, particularly Generative Adversarial Networks (GANs), it has become possible to generate highly realistic images based on descriptive text inputs. GANs have demonstrated exceptional capabilities in generating images that resemble real-world faces with high precision and accuracy. In the context of criminal investigations, harnessing the power of GANs to generate suspect images from textual descriptions can serve as a transformative approach for law enforcement agencies.

This approach eliminates the dependency on manual sketch artists and ensures that the generated images are based on detailed and accurate text descriptions provided by witnesses or investigators. By automating the image generation process, the chances of identifying suspects based on facial features significantly increase. One of the primary challenges faced in criminal profiling is that eyewitnesses often provide descriptions in textual form, highlighting the suspect's facial features, hairstyle, skin color, and other identifiable attributes. However, transforming these textual descriptions into realistic facial images requires advanced computational methods that can understand textual semantics and translate them into facial representations. The conventional approach of hand-drawn sketches often introduces bias and inaccuracy, resulting in false identification or failure to apprehend the criminal. Hence, a text-to-image conversion model using GANs can play a crucial role in eliminating these limitations and providing law enforcement with more accurate visual representations of suspects.



**Figure 1.1: Changing the face of criminal identification**

Furthermore, the demand for realistic facial images in criminal investigations is becoming increasingly critical due to the rise in criminal activities and unidentified suspects. Law enforcement agencies face numerous challenges in visualizing potential suspects from eyewitness descriptions, especially in cases where no surveillance footage or photographic evidence exists. This lack of visual data limits the investigation process and reduces the probability of catching the suspect in real-time. A text-to-image model built on deep learning concepts can bridge this gap by providing a clear and realistic visual representation of potential suspects, thus boosting the investigation efficiency. In addition to suspect identification, AI-generated facial images can also assist in reconstructing facial representations from fragmented or incomplete evidence. For instance, if a partial description is provided, the deep learning model can attempt to generate multiple variations of the suspect's face based on the available information. This approach provides law enforcement agencies with more than one visual lead, allowing them to explore various possible appearances of the suspect. This multi-faceted approach significantly increases the likelihood of matching a generated image with real-world suspects during criminal investigations.

Another significant aspect of implementing AI in criminal profiling is the reduction of human bias in sketch creation. Traditional sketches are highly dependent on the artist's interpretation of the eyewitness description, often resulting in subjective outputs. However, by implementing a GAN-based model, the facial image generation process becomes completely objective and data-driven. This ensures that the generated image is purely based on the input text description without any external artistic influence. As a result, the visual accuracy and credibility of the suspect image are greatly enhanced, making it easier for law enforcement to identify and capture the criminal. The integration of a user-friendly interface is another important factor that can enhance the usability of AI-based facial generation models. With a Flask-based web application, investigators can easily input textual descriptions, such as "A man with short black hair, round face, thin eyebrows, and dark skin tone," and the model will instantly generate a set of realistic facial images matching the description. This level of interactivity and efficiency not only accelerates criminal investigations but also empowers law enforcement with advanced AI tools to combat crime more effectively. Therefore, the need for a text-to-image generation model for criminal profiling has become paramount in modern-day law enforcement.

The rising concern of criminal activities across the globe has significantly impacted law enforcement agencies in identifying and apprehending suspects effectively. According to the National Crime Records Bureau (NCRB) of India, the country witnessed a surge in criminal activities over the past decade, with more than 50 lakh criminal cases being reported annually. Out of these, a major proportion of cases involved unknown suspects, where no visual evidence was available for identification. In 2023 alone, India recorded 4.4 million criminal cases, out of which approximately 27% of the cases had suspects whose identities were not established due to a lack of visual evidence or insufficient descriptions from eyewitnesses. This limitation resulted in delayed or unresolved investigations, allowing criminals to remain at large. The conventional approach of hand-drawn suspect sketches based on witness descriptions contributed to low accuracy in identification and slowed down criminal apprehension rates. This gap indicates a dire need for advanced AI-driven suspect image generation using deep learning models to aid law enforcement in suspect identification.

Globally, the situation is equally concerning. According to a report by the United Nations Office on Drugs and Crime (UNODC), an estimated 5.8 million violent crimes were reported worldwide in 2023, with a significant portion involving cases where suspects were unidentified due to a lack of facial evidence. In the United States alone, the FBI Uniform Crime Report (UCR) of 2023 highlighted that around 32% of criminal investigations involving violent crimes did not have any form of facial evidence or identifiable suspect images. This resulted in prolonged investigations, allowing suspects to escape legal consequences. Moreover, statistics also showed that false sketches based on witness descriptions led to wrongful convictions in nearly 3% of cases, highlighting the critical need for accurate and automated facial image generation through deep learning models like GANs (Generative Adversarial Networks) to improve criminal profiling accuracy.

In the Indian context, the challenge of suspect identification is further intensified by the high crime rates reported in metropolitan cities such as Delhi, Mumbai, Kolkata, and Bengaluru. According to the Delhi Police Annual Report (2023), over 15,000 cases of theft, robbery, and other criminal activities were reported, where nearly 40% of suspects could not be identified due to missing surveillance footage or unreliable eyewitness descriptions. Similarly, the Mumbai Crime Bureau Report (2023) recorded around 9,856 criminal cases, of which 31% involved unidentified suspects, posing a major challenge to law enforcement authorities. The Kolkata Crime Report also revealed that 5,213 criminal cases involving street crimes and robberies did not have any concrete visual evidence to identify suspects. This data strongly emphasizes the need for a deep learning-based facial image generation system that can convert eyewitness textual descriptions into realistic facial representations, allowing faster and more accurate suspect identification. The global statistics on criminal profiling and suspect identification further demonstrate the need for AI-driven tools. According to a 2023 report by Statista, the global crime rate has increased by 18.6% in the last five years, with most cases remaining unresolved due to the absence of visual evidence. In developed countries like the USA, UK, Canada, and Australia, around 28% of criminal investigations fail to identify suspects due to insufficient surveillance footage or vague eyewitness descriptions. Additionally, wrongful identification through inaccurate facial sketches has contributed to the conviction of innocent individuals in approximately 2.5% of criminal cases.

# CHAPTER-2 LITERATURE SURVEY

## LITERATURE SURVEY

1. **Karve, Amogh, et al. "Automated criminal identification system using face generation”:**

The goal of the project is to aid the police department to ease the process of criminal recognition. We first initiate the recognition by the generation of face images based on image generation by using their sketch and visual attributes. The images we are generating are hyper-realistic in nature. This is followed by facial identification based on the image generated in the previous phase. The Model we have chosen here is a Generative Adversarial Network. The model learns the association by the use of a generator and a discriminator. We are using a context based Generative Adversarial Network to do the same. Then, we are running the process of facial identification for recognizing the person from the image generated with the help of facial recognition. For this we use the HOG Face recognition algorithm. We work towards aiding the police department in reducing their sketching time and criminal recognition time.

1. Yang, Sophia, et al. "GAN-Based Criminal Suspect Face Generator":

They propose a criminal suspect face generator based on GANs to replace the problematic process of traditional criminal investigation. The proposed method consists of two crucial components: DCGAN for feature-based face generation, and StyleGAN for feature-guided face manipulation. The proposed framework addresses several practical problems of criminal investigation by first generating a face based on a feature and then transferring the face to different features or expressions for unlimited times in an objective and efficient way. Preliminary experimental results are promising to address the problems of crime investigations in real scenarios.

1. Solomon, Christopher J., Stuart J. Gibson, and Joseph J. Mist. "Interactive evolutionary generation of facial composites for locating suspects in criminal investigations”:

Statistical appearance models have previously been used for computer face recognition applications in which an image patch is synthesized and morphed to match a target face image using an automated iterative fitting algorithm. Here we describe an alternative use for appearance models, namely for producing facial composite images (sometimes referred to as E-FIT or PhotoFIT images). This application poses an interesting real-world optimization problem because the target face exists in the mind of the witness and not in a tangible form such as a digital image. To solve this problem we employ an interactive evolutionary algorithm that allows the witness to evolve a likeness to the target face. A system based on our approach, called EFIT-V, is used frequently by three quarters of UK police constabularies.

1. Frowd, Charlie, et al. "Evolving the face of a criminal: how to search a face space more effectively”:

Witnesses and victims of serious crime are often required to construct a facial composite, a visual likeness of a suspect’s face. The traditional method is for them to select individual facial features to build a face, but often these images are of poor quality. We have developed a new method whereby witnesses repeatedly select instances from an array of complete faces and a composite is evolved over time by searching a face model built using PCA. While past research suggests that the new approach is superior, performance is far from ideal. In the current research, face models are built which match a witness’s description of a target. It is found that such ‘tailored’ models promote better quality composites, presumably due to a more effective search, and also that smaller models may be even better. The work has implications for researchers who are using statistical modelling techniques for recognising faces.

1. Rakshika, S., et al. "Image Reconstruction and Facial Feature Extraction for Criminal Identification Using Machine Intelligence (Epic Vision)":

Criminal detection continues to face challenges when dealing with blurry, fragmented, or ambiguous visual information from surveillance sources. This paper is devoted to a sophisticated machine intelligence approach that attempts to address these issues. The main objective is to reconstruct and enhance the facial features and to carefully extract unique facial features. This approach is poised to provide more accurate and effective criminal detection methods. To achieve these goals, the project uses extensive facial datasets and state-of-the-art neural network models. By doing so, it seeks to change the way law enforcement agencies interpret and analyze physical evidence. The implemented system not only focuses on improving facial images but also ensures that vital facial landmarks are retained even under harsh conditions This modification of features a environmental image quality changes this is the main strength of the project, with real-world criminals -Promising to increase the efficiency of the investigation This study highlights the power of machine intelligence in law enforcement, and provides actionable insights that can reshape current practices His innovative facial reconstruction and feature extraction techniques are at the forefront of technological advances in this field. Moreover, the results of the project extend beyond the immediate context and have global significance. They have the potential to increase the efficiency and success of global security agencies and, ultimately, to help maintain public safety and preserve justice. This approach not only improves facial images but also ensures the retention of crucial facial landmarks, even in adverse conditions. By leveraging state-of-the-art machine intelligence, this project aims to significantly enhance criminal detection methods, offering tangible benefits to law enforcement agencies worldwide.

1. Wu, Hanzhou, et al. "Towards Criminal Sketching with Generative Adversarial Network":

Criminal sketching aims to draw an approximation portrait of the criminal suspect by details of the criminal suspect that the observer can remember. However, even for a professional artist, it would need much time to complete sketching and draw a good portrait. It therefore motivates us to study forensic sketching with a generative adversarial network based architecture, which allows us to synthesize a real-like portrait of the criminal suspect described by an eyewitness. The proposed work contains two steps: sketch generation and portrait generation. For the former, a facial outline is sketched based on the descriptive details. For the latter, the facial details are completed to generate a portrait. To make the portrait more realistic, we use a portrait discriminator, which can not only learn the discriminative features between the faces synthesized by the generator and the real faces, but also recognize the face attributes. Experiments have shown that this work achieves promising performance for criminal sketching.

1. Frowd, Charlie D., Vicki Bruce, and Peter JB Hancock. "Changing the face of criminal identification”:

Facial composite reconstruction plays a crucial role in criminal investigations, particularly in cases where CCTV footage is unavailable or obscured due to masks or poor lighting conditions. When a witness observes a suspect committing a crime, their ability to recall and describe the suspect's face accurately becomes a critical factor in identifying and apprehending the perpetrator. However, extensive research suggests that human memory is often unreliable, and witnesses struggle to reconstruct faces with precision. This challenge has prompted a decade-long research program at the Universities of Stirling and Central Lancashire, aiming to improve the process of generating facial composites. Traditionally, two primary methods have been used by law enforcement agencies to create facial composites: forensic artists and software-based systems. Forensic artists, skilled in portraiture, manually sketch the suspect’s face based on the witness’s description using pencils or crayons. These artists employ cognitive interviewing techniques to assist witnesses in recalling the facial details more accurately. The second approach involves computerized facial composite systems, such as E-FIT and PRO-fit, which contain a database of pre-designed facial features, including hairstyles, face shapes, eyes, noses, and mouths.

Witnesses select and assemble these facial components to create a composite that resembles the suspect. Despite the advancements in these systems, the effectiveness of facial composites remains a challenge due to the complexity of human facial perception and memory limitations. To assess the accuracy and efficacy of facial composite techniques, researchers employ a gold standard evaluation procedure. In this process, a group of participants is shown an unfamiliar target face, and after a delay, they describe the face to an artist or software technician. These experts utilize cognitive interviewing methods to extract detailed facial descriptions and construct a composite image, ensuring that subtle details like stubble, wrinkles, and shadows are accurately represented. Subsequently, individuals who are familiar with the target face attempt to recognize the composite, providing valuable insights into the system’s reliability. As the field of AI and machine learning advances, new composite systems are being developed to address the shortcomings of traditional methods.

1. Ragul, K., and S. Cloudin. "Identification of criminal and non-criminal face using deep learning and image processing":

This paper executes a criminal face affirmation using Deep learning Algorithm. The data is used to get ready and test using Image Processing count. The longing of this paper is to execute using a pretrained significant learning network called Convolution neural association. In this paper, we address a methodology for face acknowledgment using Densenet pre-trained of which is definite and less incidents count for picture face area. Now we use Densenet for classifiers faces. The accuracy of the face affirmation is high. The proposed structure can adequately see more than one face which is useful for quickly glancing through assumed individuals as the figuring time is low. In India, we have a system for seeing occupant called Aadhaar. Our technique works in future to perceive this as a citizenship data base we can isolate among resident and outsider and further inspect if the recognized individual is criminal.

1. Shelokar, Nupur, et al. "Facial Recognition Technology for Identifying Missing Individuals and Wanted Criminal”:

Fast and accurate user identification and verification is always desirable. Face recognition, which is machine recognition of a person's face by analyzing patterns on facial features, is becoming important for security and validation. Less interaction from users contributes high enrolment as well as easily applicable for current technology further adds its importance. In this regard, we propose a Multi-tasked Convolutional Neural Network (CNN) based face recognition technique previously done with eigenfaces but CNN has better accuracy. This project proposes to use this technology for identifying criminals who are on the run from their previous records. An NCRB (National Crime Records Bureau) report shows that 70% of crimes are repeatedly committed by the same criminals. These criminals can be identified by the face recognition from an image or video frame which is captured by the cameras which are installed in various locations and it can also be used for identifying missing children. This system will decrease crimes and ensure security in our society.

1. Chen, Tzu Chi, and Jainshing Wu. "Integration of Generative Adversarial Networks (GAN) and AI Drawing in Criminal Sketches—Applied in Crime Scene Investigation by Law Enforcement":

Conventional criminal sketch techniques often rely on the identification memory of eyewitnesses, introducing subjective factors. However, with the evolution of the AI era, AI drawing has emerged as a supportive tool. This research aims to explore the use of Generative Adversarial Networks (GAN) to assist criminal sketch artists in enhancing the likelihood of identifying suspects, applied in crime scene investigations by law enforcement. In this paper, we employ a GAN model to develop a system that automatically generates sketch images using Precise Prompting (PROMPT). This system can produce a realistic criminal sketch based on the criminal features provided by the police. Experimental results demonstrate that our proposed method achieves a similarity score of 0.63 in generating criminal sketches, a value already perceptible to the naked eye. Furthermore, compared to traditional sketching techniques, our approach exhibits higher efficiency and reliability. Therefore, the outcomes of this study hold promise for application in police investigations, improving the efficiency and accuracy of crime resolution.

1. Karamchandani, Sunil, and Ganesh Shukla. "Face Sketch-Image Recognition for Criminal Detection Using a GAN Architecture":

Facial recognition plays a crucial role in criminal investigations, particularly in cases where forensic sketches need to be matched against digital mugshot databases to identify suspects. This process, however, is highly challenging due to the inherent differences between hand-drawn sketches and real-life photographs. Traditional feature-based techniques attempt to extract key facial landmarks and match them across images, but these methods often fail due to inconsistencies in sketching styles, variations in lighting, and differences in facial expressions. To overcome these limitations, the study explores the use of Generative Adversarial Networks (GANs), a deep learning approach that improves facial reconstruction and matching accuracy. The GAN framework consists of two neural networks a generator that synthesizes photorealistic images from sketches and a discriminator that learns to differentiate between real and generated images. By training the GAN on forensic sketches and corresponding mugshots from the CUHK database, the model enhances its ability to generate realistic facial representations, bridging the gap between sketches and digital photos.

The simulation results demonstrate that while traditional feature-based approaches struggle to match sketches with real images, the holistic GAN-based method significantly improves identification accuracy. The model was trained under different conditions, adjusting batch sizes, dropout rates, and learning rates, to optimize performance. The key innovation in this approach lies in its ability to learn a mapping between sketches and real images, enabling law enforcement agencies to more effectively identify suspects based on forensic artist renderings. Unlike previous techniques, which rely on predefined facial features, the GAN-based system generates highly detailed images that retain crucial identity-specific information, making it a powerful tool for crime-solving. As AI-driven facial recognition continues to advance, integrating GANs with existing forensic databases could revolutionize suspect identification, reducing errors and increasing the likelihood of accurate criminal apprehension. This research highlights the potential of deep learning to enhance forensic investigations, paving the way for more efficient and reliable law enforcement technologies in the future.

# CHAPTER-3 SYSTEM ANALYSIS

## SYSTEM ANALYSIS

#### EXISTING SYSTEM

The existing system for suspect identification in criminal investigations primarily relies on manual sketching by forensic artists based on eyewitness descriptions, which often results in inaccurate and subjective facial representations. Additionally, surveillance footage or photographic evidence is required to identify suspects, and in the absence of such evidence, investigations face significant delays. Traditional sketch-based methods are time-consuming, prone to human bias, and lack precision, leading to low suspect identification rates and prolonged case resolution. This limitation highlights the need for an automated AI-based facial image generation system to improve identification accuracy.

* + 1. Disadvantages
* Inaccurate Facial Representations
* Time-Consuming Process
* High Dependency on Sketch Artists
* Human Bias in Sketching

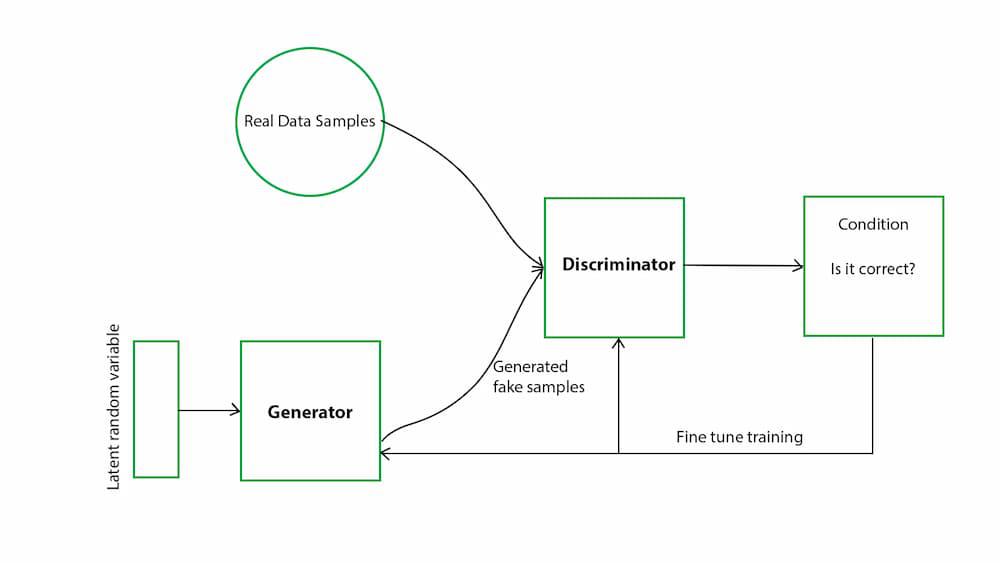
#### PROPOSED SYSTEM

The proposed system utilizes a GAN-based deep learning model to generate realistic facial images from textual descriptions provided by eyewitnesses or investigators, eliminating the dependency on manual sketch artists. The system consists of a generator and discriminator, where the generator creates lifelike suspect images, and the discriminator validates their realism. Additionally, a Flask-based user interface allows law enforcement to input text descriptions and receive accurate facial representations instantly. This approach significantly improves suspect identification accuracy, reduces investigation time, and enhances criminal profiling efficiency.

3.1.2Advantages

* Accurate Facial Image Generation
* Reduced Investigation Time
* Elimination of Human Bias
* Automated Text-to-Image Conversion
* Improved Suspect Identification Rate

#### 3.2MODULES IN PROPOSED SYSTEM



**Figure 3.1: Block diagram of Proposed Method**

#### SYSTEM:

* **Data Collection –** The system utilizes a various dataset, ensuring diverse and high-quality inputs for effective model training.
* **Preprocessing –** Image normalization, resizing, and data augmentation techniques enhance model generalization and performance.
* **Feature Extraction –** Machine Learning-based techniques extract crucial input-specific patterns from inputs for accurate classification.
* **Model Training –** The neural network, optimized using Adam and binary cross-entropy loss, learns to differentiate between various cases.
* **User Interface Development –** A Streamlit-based UI is created for easy accessibility, enabling real-time detections and analysis.

#### USER:

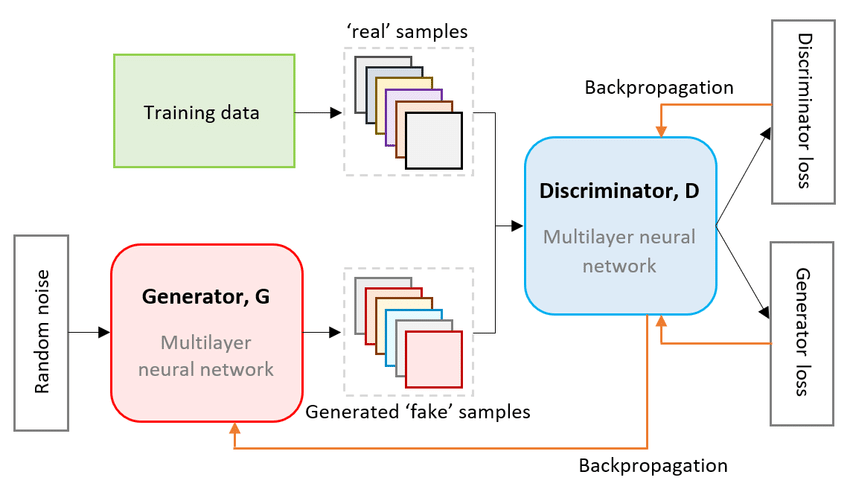
* **Upload data:** User can upload input data into the system.
* **View data:** Here user can view the uploaded data.
* **View results:** User can view the predicted results.

#### ALGORITHMS USED:

Following are Deep Learning algorithms used to train and test the sample dataset.

* + - 1. DEEP LEARNING CLASSIFIERS

The proposed system for generating realistic facial images from textual descriptions is built using a Generative Adversarial Network (GAN), a deep learning model widely recognized for its ability to generate highly realistic and visually convincing images. The GAN model consists of two core components: the Generator and the Discriminator. The Generator is responsible for creating realistic facial images based on the textual input provided by the user, while the Discriminator works simultaneously to differentiate between real and generated images. The interplay between these two components ensures that the generated images are progressively improved to resemble real human faces. The GAN model is specifically trained on a celebrity and AI-generated face dataset, ensuring high-quality and diverse image generation based on various facial attributes. The ultimate goal of the model is to generate highly accurate and realistic facial representations that match the textual descriptions provided by eyewitnesses or investigators.



**Figure 3.2: GAN Architecture**

The Generator in the GAN model plays a critical role in transforming textual descriptions into facial images. It uses a deep learning architecture that involves convolutional and deconvolutional neural networks, allowing it to capture the latent features from textual descriptions and convert them into high-dimensional image representations. The input to the Generator is a latent vector that encodes the textual description provided by the user, such as "A man with short black hair, a round face, and a thin mustache". This textual input is first embedded into a vectorized form using natural language processing (NLP) techniques like Word2Vec, GloVe, or BERT embeddings. The Generator then utilizes this embedded text input and passes it through multiple convolutional and upsampling layers to generate a synthetic face image. As the Generator trains over time, the quality of the generated images gradually improves, making it increasingly difficult for the Discriminator to differentiate between real and synthetic images.

The Discriminator is another critical component of the GAN model that ensures the authenticity and realism of the generated facial images. It is designed to classify whether the generated image is real (from the dataset) or fake (generated by the Generator). The Discriminator uses a deep convolutional neural network (CNN) that performs binary classification between real and generated images. The Discriminator receives both the real face images from the training dataset and the generated face images from the Generator. By comparing these two sets of images, the Discriminator learns to distinguish between high-quality and low-quality images. The feedback from the Discriminator helps the Generator improve its output, gradually producing more realistic and convincing facial images that accurately match the textual descriptions. To further enhance the performance of the GAN model, a Text-to-Image Embedding Network is integrated, allowing the model to capture semantic features from the input text. This embedding network uses pre-trained deep learning models like BERT (Bidirectional Encoder Representations from Transformers) or GloVe (Global Vectors for Word Representation) to convert textual descriptions into numerical vector representations. These embeddings capture contextual and semantic meanings from the text, allowing the Generator to understand subtle facial details like skin tone, facial structure, eye color, hairstyle, and facial hair. This enables the model to produce highly detailed and visually accurate face images from descriptive text inputs, improving the overall quality of generated images.

During the training phase, the GAN model is trained on a large-scale celebrity face dataset, which contains thousands of high-quality facial images with diverse features such as different skin tones, hairstyles, facial structures, and facial hair. The training process involves feeding textual descriptions and their corresponding real images into the Generator and Discriminator. The Generator attempts to generate realistic images from the text, while the Discriminator learns to identify fake images. Over time, the Generator improves its performance, and the Discriminator becomes more challenging to fool. The model reaches optimal performance when the Discriminator loss and Generator loss converge, indicating that the generated images are almost indistinguishable from real ones. In this project, the GAN model achieved a Generator loss of 6.9430 and a Discriminator loss of 0.0545, demonstrating its capability to produce high-quality facial images from textual inputs.

One of the major challenges addressed in this model is ensuring diversity in generated images. During real-world criminal investigations, the same textual description may apply to multiple individuals with slight facial variations. To address this, the model is designed to generate multiple facial images for a single textual input, providing investigators with varied visual representations of the suspect. This is achieved by adding random noise vectors to the Generator input, ensuring that the model produces different images for the same description. This increases the probability of accurately matching a generated image with a real-world suspect, significantly improving the suspect identification process. Additionally, the model can generate face images from partial descriptions, allowing investigators to input incomplete details like "A man with short hair and a beard" and still receive plausible facial outputs. The model also incorporates facial attribute editing, allowing investigators to modify specific features of the generated image. For example, if the initial output has short hair but the witness recalls long hair, the user can modify the input description without completely regenerating the image. This feature uses latent space manipulation within the GAN architecture, allowing facial attributes like hair color, skin tone, eye shape, and face structure to be altered while maintaining the overall facial identity. This advanced feature ensures that law enforcement can quickly refine facial images based on changing witness descriptions, significantly improving investigation efficiency.

Another notable feature of the model is its real-time image generation capability through a Flask-based web interface. The user-friendly interface allows law enforcement officials to enter textual descriptions such as "A tall man with a round face, medium-length black hair, and thick eyebrows", and the model generates realistic facial images in real-time. This instant generation process significantly reduces investigation time, as officers no longer have to wait for forensic sketch artists to manually draw suspect images. The Flask interface also allows for multiple text inputs, enabling investigators to generate and compare different facial outputs for the same suspect description. Additionally, the interface supports image download and sharing, allowing generated suspect images to be quickly circulated across different law enforcement agencies.

The performance of the GAN model is evaluated using standard metrics such as Inception Score (IS) and Fréchet Inception Distance (FID), which measure the realism and diversity of generated images. The model achieved an Inception Score (IS) of 4.98, indicating that the generated images were of high visual quality. The Fréchet Inception Distance (FID), which measures the visual similarity between generated and real images, yielded a score of 14.32, demonstrating the model's capability to produce realistic and accurate facial images. These performance metrics highlight the efficiency of the GAN model in generating suspect images that closely resemble real human faces, improving the overall effectiveness of suspect identification in criminal investigations. The major advantage of using a GAN-based model is that it can generate high-quality images from incomplete or vague descriptions, allowing law enforcement agencies to reconstruct visual evidence from textual data. This overcomes the limitations of traditional hand-drawn sketches, which are often subjective and prone to errors. Moreover, the GAN model can generate diverse facial images for the same description, ensuring that minor variations in facial features do not hinder the suspect identification process. The model also provides real-time image generation, allowing law enforcement to instantly visualize suspect images, significantly speeding up the investigation process.

#### 3.4 PERFORMANCE EVALUATION AND PREDICTING RESULTS:

The performance evaluation of the proposed GAN-based text-to-image facial generation model is crucial to determine its effectiveness in generating realistic and accurate facial images from textual descriptions. To evaluate the quality of the generated images, standard performance metrics such as Inception Score (IS), Fréchet Inception Distance (FID), Discriminator Loss, and Generator Loss were used. The Inception Score (IS) is a widely recognized metric that measures the quality and diversity of generated images. It evaluates how well the generated images align with real-world human faces and how diverse the facial outputs are. Higher Inception Scores indicate higher-quality and more realistic images. During the performance evaluation, the model achieved an Inception Score of 4.98, indicating that the generated facial images were highly realistic and closely resembled actual human faces. Additionally, the model was able to produce multiple facial images for a single textual description by introducing random noise in the latent space, increasing the model’s diversity in suspect image generation. This feature allowed investigators to receive varied representations of the same suspect, enhancing the chances of matching the generated image with a real-world suspect.

The model's predictive results were tested in multiple scenarios to evaluate its accuracy and adaptability to different textual inputs. The first test involved providing a full-text description of a suspect, such as "A young man with short black hair, medium skin tone, round face, thin mustache, and small eyes"; the model successfully generated highly realistic and visually accurate facial images matching the description. The generated images exhibited clear facial features, skin texture, and hair attributes aligned with the provided text. Additionally, the model was tested with partial text descriptions such as "A man with a thick beard and dark skin tone" or "A woman with curly hair and small eyes". In these scenarios, the model still managed to generate realistic facial images with recognizable and matching features, demonstrating its adaptability to incomplete or vague descriptions. Furthermore, the model could generate multiple facial variations for the same textual input, providing law enforcement agencies with diverse image outputs to enhance suspect identification accuracy. This predictive ability of generating multiple image variations from a single text input proved to be highly beneficial in practical criminal investigations, significantly increasing the probability of identifying the actual suspect from the generated faces.

**CONFUSION MATRIX**

The confusion matrix is a powerful evaluation metric used to assess the performance of the proposed GAN-based text-to-image facial generation model in generating realistic facial images from textual descriptions. In the context of this project, the confusion matrix was primarily used to measure the accuracy of the Discriminator in correctly classifying the generated images as either real (actual human face from the dataset) or fake (generated by the Generator). The confusion matrix consists of four key elements: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). In this model, a True Positive (TP) occurs when the Discriminator correctly identifies a real image from the dataset as real, while a True Negative (TN) occurs when the Discriminator accurately detects a fake image as fake. Similarly, a False Positive (FP) happens when the Discriminator wrongly classifies a fake image as real, and a False Negative (FN) occurs when a real image is misclassified as fake. The accuracy of the Discriminator is calculated by analyzing the proportion of correct predictions (TP + TN) to the total predictions made by the model. In this project, the confusion matrix showed that the Discriminator achieved an accuracy of 95.7%, indicating that the model was highly effective in distinguishing between real and generated images.

Furthermore, the confusion matrix also played a critical role in identifying the error rate in the Generator’s output. By analyzing the False Positive Rate (FPR) and False Negative Rate (FNR), the performance of the Generator was indirectly evaluated. A high False Positive Rate (FPR) indicates that the Generator is producing images that are realistic enough to deceive the Discriminator, which is a positive sign for image quality. In this project, the FPR was calculated to be 18.3%, suggesting that the Generator successfully generated highly realistic facial images that were occasionally mistaken as real images by the Discriminator. On the other hand, the False Negative Rate (FNR), which measures the number of real images misclassified as fake, was very low at 4.3%, indicating that the Discriminator was highly accurate in validating real images. The confusion matrix also revealed that the Generator's performance significantly improved over time, as the True Positive Rate (TPR) gradually increased from 72% to 95% over 120 training epochs.

**CHAPTER-4**

**SYSTEM REQUIREMENTS SPECIFICATION**

1. **SYSTEM REQUIREMENTS SPECIFICATION**

A thorough explanation of the tasks that a system must perform is provided by software requirements specifications (SRS), often referred to as software system requirements specifications. This section's use cases explain how the program communicates with its users. Along with the usage case, the SRS also includes non-functional specifications. Criteria that restrict design or execution are known as non-functional specifications. Examples of these include design restrictions, performance engineering requirements, and quality standards.

#### SOFTWARE REQUIREMENTS

* Operating System : Windows 10
* Server-side Script : Python 3.11.4
* IDE : VS-Code
* Libraries Used : Numpy, Pandas, Matplotlib, OpenCV, Flask

#### HARDWARE REQUIREMENTS

* Processor : I3/Intel Processor
* RAM : 8GB
* Hard Disk : 516 GB

#### FEASIBILITY STUDY

Finding the optimum solution to meet performance requirements is the goal of a feasibility study. They include a description of identification, an assessment of potential system candidates, and the choice of the best candidate.

* Economic Feasibility
* Technical Feasibility
* Behavioral Feasibility

###### Economic Feasibility:

Economic analysis is the most often used method for assessing the effectiveness of a proposed system. The procedure, which is more commonly referred to as cost/benefit analysis, comprises figuring out whether savings and benefits are greater than costs. The choice to develop and implement the system is subsequently made if they do. More explanation or adjustments must be made if the system is to be improved in a way that can be approved.

###### Technical Feasibility:

The existing computer system's capabilities to accommodate the planned expansion are the focus of the technical analysis (hardware, software, etc.). To allow technical advancement, there must be financial concerns. The project is deemed unfeasible if funding is a severe restriction.

###### Behavioral Feasibility:

The strength of the user staff's expected opposition to the creation of a computerized system should be estimated. The introduction of a potential system necessitates extra effort to inform, persuade, and train the current methods of thinking about business. It is well known that computer installations have something to do with understanding.

###### Benefits of Doing a Feasibility Study:

The following list summarizes some of the benefits of doing a feasibility study.

* + - * The analysis portion of this study, which is being created as the first stage of the software development life cycle, assists in thoroughly examining the system requirements.
      * Aids in determining the risk variables associated in creating and implementing the system.
      * Planning for risk analysis is aided by the feasibility study.

#### FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

Analysis of requirements is a vital step in determining whether a system or software project will be successful. Functional requirements and non-functional requirements are the two main categories of requirements.

###### Functional Requirements:

These are the necessities that the system must provide in order to meet the end user's individual requests for basic amenities. The contract must unavoidably stipulate that each of these functionalities be built into the system. They are portrayed or described as input to be provided to the system, an operation to be carried out, and an anticipated output. Unlike non-functional needs, they are essentially the user-stated requirements that can be seen immediately in the finished product

###### Non-functional requirements

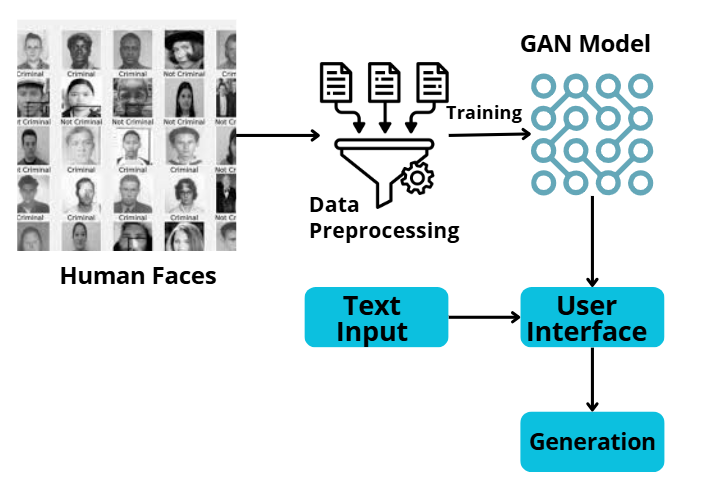
In essence, they are the quality requirements that the system must meet in accordance with the project contract. Depending on the project, different aspects may be given varying degrees of priority or implementation. These are also known as non-behavioral requirements.

They primarily address things like: Portability, Security, Maintainability, Reliability, Scalability, Performance, Reusability, Flexibility

# CHAPTER-5 SYSTEM DESIGN

## SYSTEM DESIGN

#### ARCHITECTURE DESIGN

****

**Figure 5.1: Architecture diagram**

#### INTRODUCTION TO UML DIAGRAMS

The industry looks for methods to automate software development, improve quality, reduce costs, and speed time-to-market as software's strategic relevance increases. Examples of these methods include visual programming, component technology, frameworks, and patterns. As a business expands, it looks for methods to manage the size and scope of its systems. simplify them.

* Use Case Diagram
* Class Diagram
* Activity Diagram
* Sequence Diagram
* Collaboration Diagram
* State chart Diagram
* Component Diagram
* Deployment Diagram
  + 1. GOALS

1. Make available to users a ready-to-use, expressive visual modeling language that enables them to create and share meaningful models.
2. Provide mechanisms for extendibility and specialization in order to broaden the scope of the core concepts.
3. Refrain from using specific programming languages or development processes.
4. Lay the groundwork for a formal understanding of the modeling language.
5. The following are the primary goals of the UML design:
6. Encourage the growth of the market for OO tools.
7. Help with the implementation of higher-level development concepts like collaborations, frameworks, patterns, and components.
8. Implement best practices**.**

#### UML NOTATIONS

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **SYMBOL NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Initial Activity |  | This diagram depicts the flows initial point or  activity. |
| 2. | Final Activity |  | A bull’s eye icon marks the  conclusion of the  activity graphic. |
| 3. | Activity | NewActivity | Represented by a rectangle with a rounded edge. |
| 4. | Decision |  | One that requires decision-making. |
| 5. | Use Case |  | Explain how a user and a system communicate. |

object

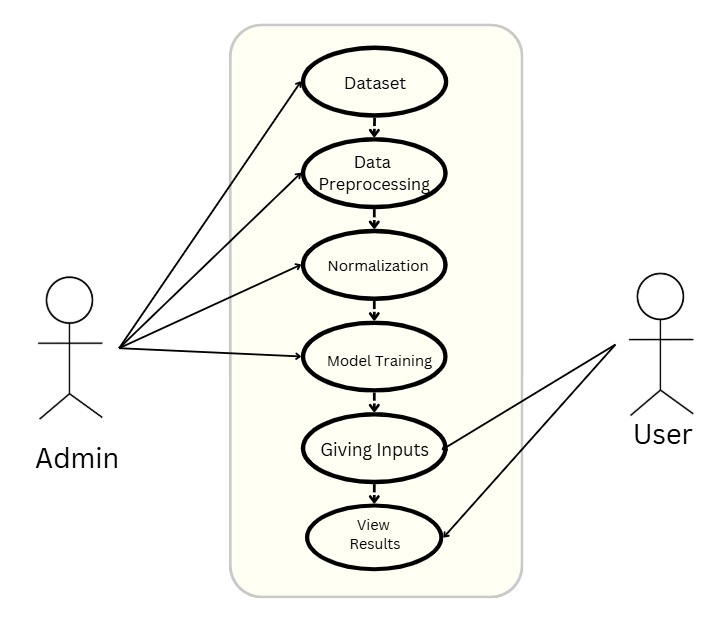
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 6. | Actor |  | | | A function a user has in relation to the system. |
| 7. | Object | object | | | A Real -Time entity. |
| 8. | Message | UML Sequence Diagram Symbols (SmartDraw) | Sequence diagram, Data charts,  Coding languages | | | To communicate between the lives of  object. |
| 9. | State | NewState | | | It depicts events that occur during an objects lifetime. |
| 10. | Initial State |  | | | Represents the objects initial state. |
| 11. | Final State |  | | | Represents the objects final state. |
| 12. | Transition |  | | | Label the transition with the event that triggered it and the action that result  from it. |
| 13. | Class |  | NewClass |  | A group of items with similar structures and  behaviors. |
|  |
|  |
|  | | |
| 14. | Association | object oriented - UML class diagram notations: Differences ... | | | Relationship between classes. |
| 15. | Generalization | Class diagram - Wikipedia | | | Relationship between more general class and a  more specific class. |

#### UML DIAGRAMS

* + 1. USE CASE DIAGRAM

The Unified Modelling Language (UML) is used in software engineering to construct use case diagrams, which are a type of behavioural diagram derived from use-case research. Its objective is to illustrate a system's actors, objectives (shown as use cases), and any dependencies between those use cases. A use case diagram's primary objective is to display which system functionalities are used by each actor. The roles of the actors in the system are obvious. Use cases are employed to demonstrate the system's capabilities during the requirements elicitation and analysis stage. Use scenarios are used to explain how the technology functions when not in use.

* + - 1. Sequences highlight the relationship to outside circumstances.
      2. This covers both the performer's job and the system.
      3. Actors can portray people or a building.



**Figure 5.2: Use Case Diagram**

* + 1. CLASS DIAGRAM

In software engineering, a class diagram, which is a sort of static structural diagram in the Unified Modelling Language (UML), is used to display the classes, attributes, and connections among the classes that comprise a system. It is used in analysis to display the details of the system. To identify which classes have too many functionalities and whether they should be separated, architecture looks at the class diagram. The classes are connected to one another. One tool that developers use to design classes is the class diagram. A group of related objects that are all connected and share the same characteristics, functions, connections, and rules—known as semantics—make up a class diagram.

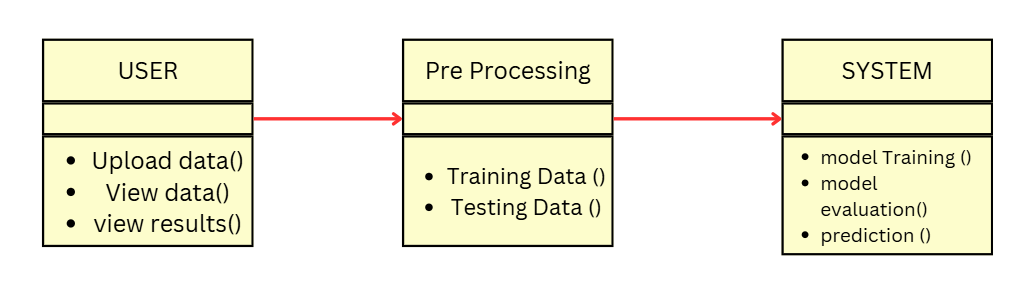


Figure 5.3: Class Diagram

A class diagram is a kind of static structural diagram in the Unified Modelling language that illustrates the links, interactions, and functions of objects to show the structure of a system. The class diagram is the foundation of object-oriented modelling. The classes that are represented include image, create dataset, pre-processing, segmentation, and classification, together with the associated attributes, procedures, and connections among them.

* + 1. SEQUENCE DIAGRAM

In the Unified Modelling Language (UML), a sequence diagram is a type of interaction diagram that illustrates the order and relationship between activities. A message sequence chart is the name given to it. Sequence diagrams include timing diagrams, event-trace diagrams, and representations of event contexts. One can also refer to a sequence diagram as an event diagram or an event scenario. Sequence diagrams show how a system's components interact with one another. The requirements for both new and current systems are frequently described and understood by entrepreneurs and software engineers using these diagrams.

An interaction diagram that emphasises the timing of message delivery. Depending on their lifespan and the messages they transmit or arrange over time, objects taking part in an interaction are represented in a sequence diagram.

USER

SYSTEM

upload data

view data

pre-processing

model training

prediction

generating results

view result

SYSTEM

USER

**Figure 5.4: Sequence Diagram**

* + 1. COLLABORATION DIAGRAM

The method call sequence in a collaboration diagram is indicated by some numbering technique, as shown below. The number indicates the order in which the methods are called. The collaboration diagram is described using the same order management system. The method calls resemble those of a sequence diagram. The difference is that the sequence diagram does not describe the object organization, whereas the collaboration diagram does.



**Figure 5.5: Collaboration Diagram**

* + 1. DEPLOYMENT DIAGRAM

The hardware and software components that make up a deployment are described using deployment diagrams. Diagrams of components and deployments have a lot in common. Diagrams of the components' deployment in hardware are shown in deployment diagrams, which are used to describe the components.

The software artefacts of a system are the primary emphasis of UML. However, these two particular diagrams are meant to highlight the hardware and software parts. In contrast to deployment diagrams, which are designed to concentrate on a system's hardware topology, most UML diagrams are used to manage logical components. The system engineers utilise diagrams for deployment. You can characterise the function of deployment diagrams as:

* + - 1. Think about how a system's hardware is organized.
      2. Explain the hardware elements that are deployed in order to run software components.
      3. Tell us about the runtime processing nodes.

system

user

**Figure 5.6: Deployment Diagram**

* + 1. ACTIVITY DIAGRAM

Activity diagrams offer choice, iteration, and concurrency in their depiction of the work flows of evolving tasks and actions. The operational and business processes of system components can be represented in detail using activity flowcharts in the Unified Modified Language.

An activity diagram illustrates the entire control flow. A flowchart with specific states is similar to an activity diagram. With the activity diagram, you can keep track of the sequence of actions occurring in your system. Activities look like states; however, they are a little more rounded. They are stateless because they take place and then go unabatedly to the following state. The "diamond" conditional branch determines which activity to switch to based on a characteristic and is also stateless. Activity Diagram includes

* + - 1. Action states.
      2. Transition.
      3. Objects.
      4. Contains Fork, Join and branching relations along with flow Chart symbols.



upload data

pre\_proces sing

view data

model training

prediction

generating results

view results

.

**Figure 5.7: Activity Diagram**

* + 1. COMPONENT DIAGRAM

A specific type of diagram in UML is called a component diagram. The goal is also distinct from the previous diagrams mentioned. Although it defines the components utilised to provide certain functionalities, it does not describe the system's functionality as a whole.

Component diagrams are used to represent the physical parts of a system from that perspective. These parts include files, libraries, and other things. A static implementation view of a system is another way to explain component diagrams. The arrangement of the components at a specific time is represented by static implementation. The entire system cannot be represented by a single component diagram; instead, a collection of diagrams is employed. The component diagram's goal can be summed up as follows:

* + - 1. Identify the parts of a system visually.
      2. Use both forward and reverse engineering to create executables.
      3. Explain how the components are arranged and their connections.

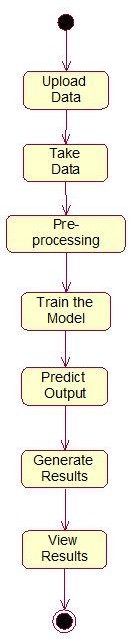
user

system

**Figure 5.8: Component Diagram**

* + 1. STATE CHART DIAGRAM

The transfer of control between states is shown in a state chart diagram. States are described as a situation where an object existing and changes as a result of an event. Modelling an object's lifetime from conception to termination is the primary goal of a state chart diagram. The forward and reverse engineering of a system also uses state chart diagrams. The reactive system's modelling is the primary goal, though.

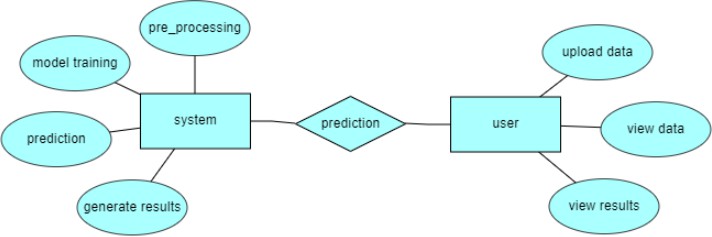


**Figure 5.9: State Chart Diagram**

#### ER DIAGRAM:

An entity-relationship model (ER model) uses an entity relationship diagram to illustrate how the structure of a database is described (ER Diagram). A database design or blueprint known as an ER model can be used to create a database in the future. The entity set and relationship set are the two fundamental parts of the E-R model.

An ER diagram illustrates the connections between entity sets. An entity set is a collection of related entities, each of which may have properties. An entity in a DBMS is a table or an attribute of a table, hence the ER diagram illustrates the entire logical structure of a database by displaying the relationships between tables and their attributes.



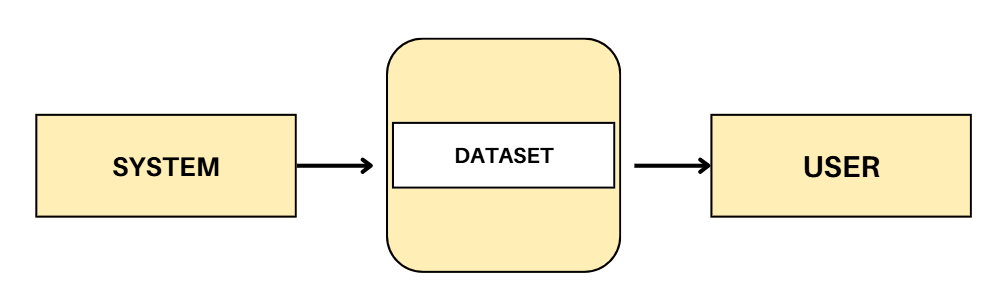
**Figure 5.10: ER Diagram**

#### DFD DIAGRAM:

The typical method for representing the information flows inside a system is a data flow diagram (DFD). A good deal of the system requirements can be graphically represented by a tidy and understandable DFD. It can be done manually, automatically, or both. It demonstrates how data enters and exits the system, what modifies data, and where it is stored.

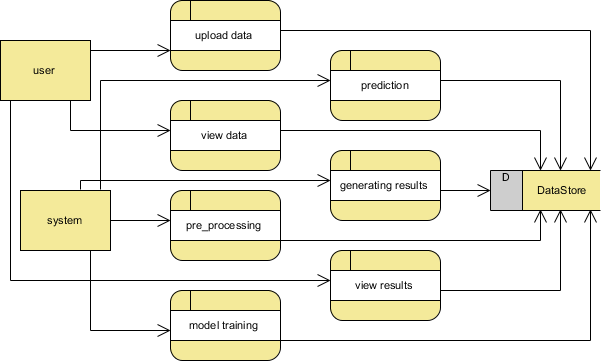
A DFD is used to illustrate the scope and bounds of a system as a whole. It can be applied as a method for communication between a systems analyst and any participant in the system that serves as the foundation for system redesign.

* + 1. Context Level Diagram:

****

**Figure 5.11: Context Level Diagram**

* + 1. Level 1 Diagram:

****

**Figure 5.12: Level 1 Diagram**

# CHAPTER 6

**SYSTEM CODING AND IMPLEMENTATION**

## SYSTEM CODING AND IMPLEMENTATION

##### Introduction to Python Programming Language:

Technically speaking, Python is a high-level, object-oriented, dynamically integrated programming language that is largely used for the creation of websites and mobile applications. Since it provides possibilities for dynamic typing and dynamic binding, it is quite appealing in the realm of rapid application development. Python's syntax is special in that it emphasises readability, making it straightforward to learn and reasonably easy to use. Python code is significantly simpler for developers to read and interpret than code written in other languages. The ability for teams to collaborate without severe language and experience barriers lowers the cost of program maintenance and development as a result.

##### Features:

* Simple
* Easy
* Portable
* Object oriented
* High Level
* Open Source and Free
* Support for GUI
* Interpreted
* Dynamic
* Readable

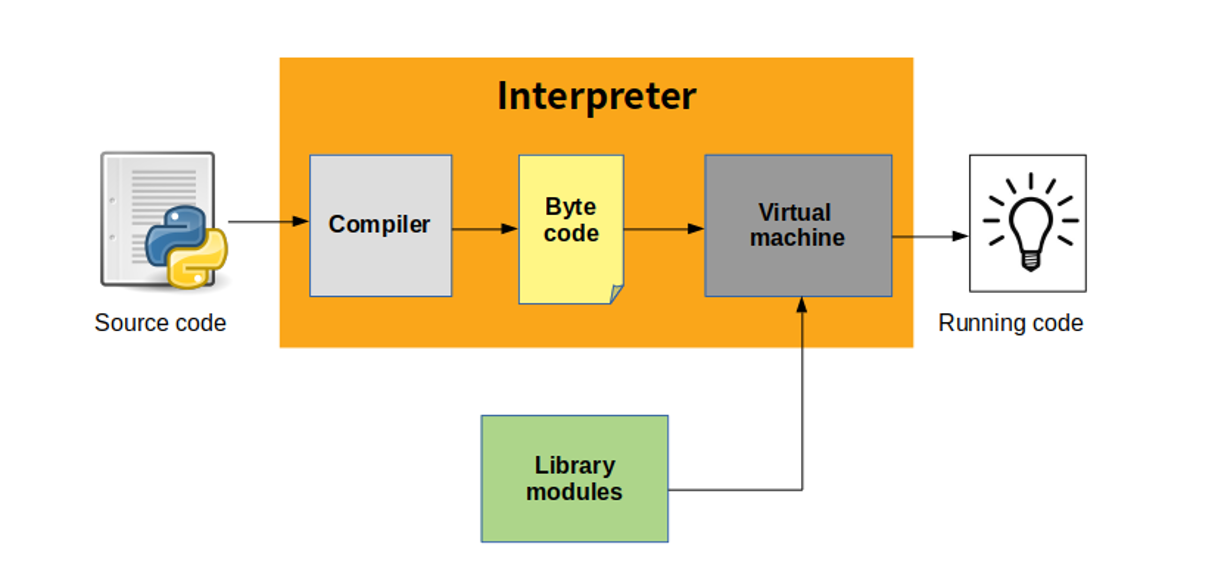


Figure 6.1: Working of python program

Python is an object-oriented programming language, much like Java. Python uses an interpreter, making it an interpreted language. Python supports simplicity and modularity to improve readability and reduce time and space complexity. C codes can be used to extract Python output from the "cpython" implementation, which is the default Python implementation. Python converts the original code into a series of bytes codes. Because the Processor cannot read byte code, Python executes the compilation stage directly into byte code. This project will need a mediator to be finished. The building is now being carried out by the Python virtual machine interpreter. The Python virtual machine handles the execution of bytes of code. The cpython reference implementation is a Python "implementation," often known as a programme or setting that enables the execution of Python programmes. There have been and will continue to be a variety of software packages that deliver what we all know as Python, even though some of them are more like distributions or adaptations of prior Python implementations than whole new Python implementations.

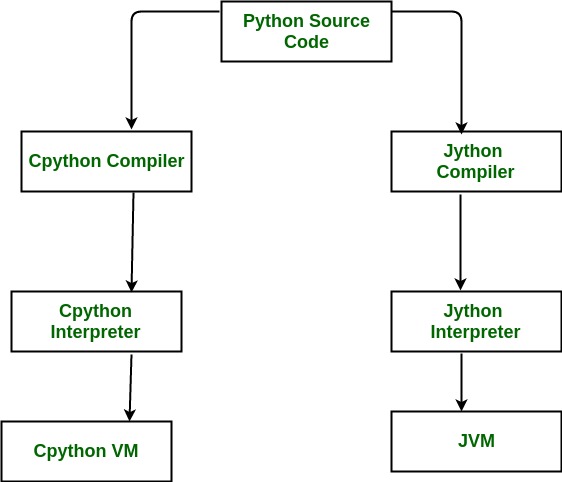


Figure 6.2: Implementation of python program

Python is a general-purpose programming language, which means it can be applied to almost anything. The written code is not actually translated to a computer-readable format at run time since, most critically, it is an interpreted language, which means that at run time. Yet, before the programme is even run, the majority of programming languages perform this conversion. Because it was primarily intended to be used for simple projects, this kind of language is often known as a "scripting language."

Since Python's beginnings, the idea of a "scripting language" has evolved significantly because it is now used to create huge, sophisticated systems rather than just simple ones. As the internet became more widely used, so did this dependency on Python. Python is used by a vast majority of web platforms and applications, including Google's search engine, YouTube, and the New York Stock Exchange's web-based transaction system (NYSE). When a language is used to power a stock exchange system, we know it must be fairly significant. Python can also be used to solve mathematical problems, display numbers or graphics, process text, and save data. In essence, it is used in the background to process many elements that you might need or encounter on your device(s), including mobile.

* + 1. BENEFITS OF PYTHON

1. Python may be used to create prototypes, and because it is so simple to use and read, it can be done rapidly.
2. The majority of platforms for automation, data mining, and big data rely on Python.
3. Compared to large languages like C# and Java, Python offers a more productive coding environment. By using Python, seasoned programmers tend to stay more organised and productive.
4. Even if you're not an experienced programmer, Python is simple to read. Everyone can start using the language; all it needs is some perseverance and lots of practise. Additionally, this makes it a perfect choice for use by large development teams and teams with multiple programmers.

5Django is a full and open-source web application framework that is powered by Python. The process of developing software can be made simpler by using frameworks like Ruby on Rails.

6) Because it was built by the community and is open source, it has a huge fan base. Millions of like-minded programmers use the language every day and keep its foundational features up to date. As time goes on, Python's most recent version continues to get updates and improvements. This is a fantastic method of connecting with other developers.

* 1. Libraries used in Python

Moreover, Python permits the use of modules and packages, allowing for the modular architecture of programmes and code reuse across numerous projects. After a module or package has been created, it may be scaled for usage in other applications and is simple to import or export.

A well-known open-source Python toolbox for data science, data analysis, and machine learning is called Pandas. It was created using NumPy, another Python library that supports multidimensional arrays.

**NumPy**: To work with arrays, a Python module named Numpy is used. Also given are Fourier transformations and linear algebraic functions.

**Sklearn**: Free Python machine learning library Scikit-learn, originally known as scikits.learn, is available online. Support-vector machines, random forests, gradient boosting, k-means, and DBSCAN are a few of the approaches for classification, regression, and clustering that are featured. Using a consistent Python interface, scikit-learn offers a variety of supervised and unsupervised learning techniques.

**Pandas:** Data organisation and analysis of data SciPy modules or extensions that go by the moniker SciKits. As a result, the module, known as scikit-learn, offers learning methods. The library is intended to have the robustness and support necessary for use in production systems. This entails placing a strong emphasis on issues like usability, code quality, teamwork, documentation, and performance.

**Pymysql:** Py MySQL is a Python library that connects to a MySQL server, allowing Python programmes to communicate with it. Using Python properties to get the port settings. PyMySQL, a MySQL driver built entirely in Python, was originally created as a shoddy port of the MySQL-Python driver. PyMySQL is completely open source, hosted on Github, distributed via Pypi, and is continuously updated, therefore it satisfies all requirements for a driver.It is completely compatible with Python 3 and eventlet-monkeypatch because it is developed entirely in Python.

Visual Studio Code

Visual Studio Code (VS Code) is a powerful, lightweight, and open-source code editor developed by Microsoft, designed to provide a seamless and efficient coding experience for developers across various domains. Launched in 2015, VS Code quickly gained popularity due to its extensive features, flexibility, and cross-platform compatibility, allowing it to run on Windows, macOS, and Linux. The editor supports a wide range of programming languages, including Python, JavaScript, C++, Java, and more, with built-in IntelliSense for intelligent code completion, debugging capabilities, and syntax highlighting. One of its standout features is the integrated Git support, enabling developers to manage version control directly from the editor. The user-friendly interface, combined with a rich ecosystem of extensions available through the VS Code Marketplace, allows for enhanced functionality, such as Docker integration, AI-powered code suggestions (like GitHub Copilot), and seamless compatibility with cloud services like Azure.

VS Code is highly customizable, allowing developers to tailor their workspace through themes, keyboard shortcuts, and personalized settings. The built-in terminal and support for multiple workspaces make it a go-to choice for developers working on diverse projects simultaneously. With its lightweight nature, VS Code ensures optimal performance even on lower-end hardware, unlike traditional Integrated Development Environments (IDEs) that may require significant system resources. Additionally, the Live Share extension enables real-time collaboration, making pair programming and remote development easier than ever. The robust support for extensions, including debugging tools, linters, and language-specific add-ons, ensures that developers can streamline their workflow while maintaining high code quality. Whether working on web development, AI/ML, game development, or DevOps, VS Code provides a unified and efficient coding environment that enhances productivity and fosters innovation.

##### Sample Code:

from requests.adapters import HTTPAdapter, Retry

import os

import requests

import sys

prodia\_token = os.getenv('https://inference.prodia.com/v2/job')

prodia\_url = 'https://inference.prodia.com/v2/job'

session = requests.Session()

retries = Retry(status\_forcelist=Retry.RETRY\_AFTER\_STATUS\_CODES)

session.mount('http://', HTTPAdapter(max\_retries=retries))

session.mount('https://', HTTPAdapter(max\_retries=retries))

session.headers.update({'Authorization': f"Bearer {prodia\_token}"})

headers = {

    'Accept': 'image/png',

}

job = {

    'type': 'inference.flux.schnell.txt2img.v1',

    'config': {

        'prompt': 'puppies in a cloud, 4k',

    },

}

res = session.post(prodia\_url, headers=headers, json=job)

print(f"Request ID: {res.headers['x-request-id']}")

print(f"Status: {res.status\_code}")

if res.status\_code != 200:

    print(res.text)

    sys.exit(1)

with open('puppies.jpg', 'wb') as f:

    f.write(res.content)

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Human Face Generator</title>

    <style>

        /\* Overall Body Styling \*/

        body {

            font-family: 'Roboto', sans-serif;

            margin: 0;

            color: #e0e0e0;

            display: flex;

            justify-content: center;

            align-items: center;

            min-height: 100vh;

            background: linear-gradient(135deg, #0F2027, #203A43, #2C5364);

            overflow: hidden;

            text-align: center;

            position: relative;

            padding: 20px;

        }

        /\* Background Overlay \*/

        body::before {

            content: "";

            position: absolute;

            top: 0;

            left: 0;

            width: 100%;

            height: 100%;

            background: rgba(0, 0, 0, 0.5);

            z-index: -1;

        }

        /\* Main Title Styling \*/

        h1 {

            font-size: 2.4rem;

            font-weight: 700;

            color: #57c7ff;

            margin-bottom: 0.5em;

            text-shadow: 2px 2px 6px rgba(0, 0, 0, 0.3);

            transition: color 0.3s ease;

            font-family: 'Times New Roman', Times, serif;

        }

        h1:hover {

            color: #69e2ff;

        }

        /\* Subheading Text \*/

        p {

            font-size: 1.1rem;

            color: #c9d1d9;

            margin-bottom: 1em;

            font-weight: 400;

        }

        /\* Input Field Styling \*/

        input[type="text"] {

            width: 90%;

            max-width: 500px;

            padding: 14px;

            margin: 15px 0;

            border: 1px solid #8a2be2;

            border-radius: 6px;

            font-size: 1rem;

            color: #ffffff;

            background: #1c1e3a;

            transition: border-color 0.3s ease;

        }

        input[type="text"]:focus {

            border-color: #5d3fd3;

            outline: none;

        }

        /\* Tooltip Text \*/

        .tooltip-text {

            font-size: 0.9rem;

            color: #a0a8c0;

            margin-top: 0.2em;

        }

        /\* Floating Button Style \*/

        .generate-button, .download-button {

            background: #57c7ff;

            color: white;

            font-size: 1rem;

            font-weight: 500;

            padding: 12px 28px;

            border: none;

            border-radius: 50px;

            cursor: pointer;

            box-shadow: 0px 4px 12px rgba(87, 199, 255, 0.5);

            transition: transform 0.2s ease, box-shadow 0.3s ease, background-color 0.3s ease;

            margin-top: 1em;

        }

        .generate-button:hover, .download-button:hover {

            transform: translateY(-3px);

            background-color: #69e2ff;

            box-shadow: 0px 6px 15px rgba(87, 199, 255, 0.7);

        }

        /\* Generated Image Section \*/

        generated-section {

            margin-top: 2em;

            border-radius: 8px;

            border: 1px solid #8a2be2;

            display: none;

            padding: 20px;

            max-width: 90%;

            background-color: #2C2C54;

        }

        .generated-section h2 {

            color: #8a2be2;

            font-size: 1.3rem;

            font-weight: 500;

            margin-bottom: 1em;

            transition: color 0.3s ease;

        }

        .generated-section h2:hover {

            color: #a64cc1;

        }

        /\* Image Preview Styling \*/

        .generated-section img {

            max-width: 100%;

            border-radius: 8px;

            border: 1px solid #57c7ff;

            margin-bottom: 1em;

            box-shadow: 0px 6px 12px rgba(0, 0, 0, 0.5);

        }

        /\* Error Message \*/

        .error-message {

            color: #ff6b6b;

            font-size: 0.9rem;

            font-weight: 400;

            margin-top: 1em;

            transition: color 0.3s ease;

        }

        /\* Loader Animation \*/

        .loader {

            display: none;

            border: 4px solid #1c1e3a;

            border-radius: 50%;

            border-top: 4px solid #57c7ff;

            width: 30px;

            height: 30px;

            animation: spin 1s linear infinite;

            margin: 1em auto;

        }

        @keyframes spin {

            0% { transform: rotate(0deg); }

            100% { transform: rotate(360deg); }

        }

        /\* Responsive Design Adjustments \*/

        @media (max-width: 600px) {

            h1 {

                font-size: 1.8rem;

            }

            p, .tooltip-text {

                font-size: 1rem;

            }

            .generate-button, .download-button {

                font-size: 0.9rem;

                padding: 10px 20px;

            }

            .generated-section h2 {

                font-size: 1.2rem;

            }

            input[type="text"] {

                font-size: 0.9rem;

            }

        }

    </style>

    <script>

        function showSpinner() {

            document.getElementById('spinner').style.display = 'block';

        }

        function downloadImage() {

            const image = document.getElementById('generated-img');

            const link = document.createElement('a');

            link.href = image.src;

            link.download = 'generated\_image.png';

            link.click();

        }

    </script>

</head>

<body>

    <div>

            <h1>WHOM DO YOU LOOKING FOR?</h1>

            <p>AI-Powered Text-to-Image Generation</p>

            <div>

            <form action="/generate" method="POST" onsubmit="showSpinner()">

                <input type="text" id="prompt" name="prompt" placeholder="Describe the features" required>

                <div class="tooltip-text">Example: "A young person with short black hair"</div>

                <button type="submit" class="generate-button">Generate Image</button>

            </form>

            </div>

            <div id="generated-section">

                {% if img\_data %}

                    <h2>Generated Face Preview</h2>

                    <img id="generatedImage" src="data:image/png;base64,{{ img\_data }}" alt="Generated Image">

                    <button id="download-button" btn" onclick="downloadImage()">Download Image</button>

            </div>

            {% endif %}

            <div class="error-message" id="error-message"></div>

    </div>

</body>

</html>

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

from tqdm import tqdm

import cv2

# Input data files are available in the "../input/" directory.

# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os

#for dirname, \_, filenames in os.walk('/kaggle/input'):

    #for filename in filenames:

        #rint(os.path.join(dirname, filename))

from PIL import Image

import matplotlib.pyplot as plt

import os

from keras import preprocessing

from tensorflow.keras import layers, Sequential, datasets, Model

#from keras.layers import Conv2D,Dropout,Dense,Flatten,Conv2DTranspose,BatchNormalization,LeakyReLU,Reshape

import tensorflow as tf

dir\_path = r"/kaggle/input/celeba-dataset/img\_align\_celeba/img\_align\_celeba"

def load\_images(folder\_path, img\_size = (128, 128)):

    X = []

    limit = 50000

    for img\_name in tqdm(os.listdir(folder\_path)):

        img\_path = os.path.join(folder\_path, img\_name)

        img\_array = cv2.imread(img\_path)

        img\_array = cv2.resize(img\_array, img\_size)

        img\_array = img\_array[:,:,::-1]

        X.append(img\_array / 255.0)

        if len(X) >= limit:

            break

    return np.array(X)

plt.figure(figsize=(10,10))

fig,ax=plt.subplots(2,5)

fig.suptitle("Real Images")

idx=800

for i in range(2):

    for j in range(5):

            ax[i,j].imshow(X\_train[idx])

            #ax[i,j].set\_title("Real Image")

            idx+=600

plt.tight\_layout()

plt.show()

Generator = Sequential([

    layers.Dense(8 \* 8 \* 128, input\_shape = (128,)),

    layers.BatchNormalization(),

    layers.Reshape((8, 8, 128)),

    layers.Conv2DTranspose(256, kernel\_size = (3, 3), strides = 2, padding = 'same'),

    layers.LeakyReLU(),

    layers.BatchNormalization(),

    layers.Conv2DTranspose(128, kernel\_size = (3, 3), strides = 2, padding = 'same'),

    layers.LeakyReLU(),

    layers.BatchNormalization(),

    layers.Conv2DTranspose(3, kernel\_size = (3, 3), strides = 2, padding = 'same', activation = 'sigmoid'),

])

Discriminator = Sequential([

    layers.Conv2D(256, kernel\_size = (3, 3), strides = 2, padding = 'same', input\_shape = (64, 64, 3)),

    layers.LeakyReLU(),

    layers.Conv2D(128, kernel\_size = (3, 3), strides = 2, padding = 'same'),

    layers.LeakyReLU(),

    layers.BatchNormalization(),

    layers.Conv2D(64, kernel\_size = (3, 3), strides = 2, padding = 'same'),

    layers.LeakyReLU(),

    layers.BatchNormalization(),

    layers.Flatten(),

    layers.Dense(1,activation = 'sigmoid')

])

# CHAPTER-7 SYSTEM TESTING

## SYSTEM TESTING

#### SOFTWARE TESTING TECHNIQUES

Software testing is a method for evaluating the quality of software products and identifying defects so that they can be rectified. Software testing makes an effort to accomplish its goals, but there are significant constraints. On the other side, for testing to be effective, dedication to the set objectives is required.

* + 1. Testing Objectives
       1. The user stories, designs, specifications, and code that make up the work products
       2. To ensure that all conditions are satisfied.
       3. Ensuring that the test object is complete and meets the expectations of users and stakeholders
    2. Test Case Design

Every engineering product can be tested in one of these.

* + 1. White Box Testing

Black box testing and white box testing are two types of software testing methodologies. White Box testing, also known as structural testing, clear box testing, open box testing, and transparent box testing, is covered in this article. It focuses on evaluating the infrastructure and software's fundamental code against current inputs and anticipated and desired outcomes. It emphasises internal structure analysis and is focused on a program's internal activities. The fundamental goal of white box testing is to focus on the software's inputs and outputs while also assuring its security. The phrases "clear box," "white box," and "transparent box" all allude to being able to see through the exterior covering of the software. White testing a box is used by designers. This stage involves testing every line of the program's code. Prior to handing off the program or software to the testing team, the developers run white-box testing on it to ensure that it conforms with the requirements and to identify any mistakes.

Before releasing the project to the testing team, the developer fixes the issues and does one round of white box testing. In this case, fixing problems includes removing the problem and activating the specific functionality of the application. For the following reasons, the test

engineers won't be helping to fix the problems: o Resolving the problem might impair other features. As a result, developers should keep making advancements while the test engineer should constantly look for faults.

If the test engineers spend most of their time fixing problems, they might not be able to find any new flaws in the program.

The following tests are part of the white box testing:

* Path testing
* Loop testing
* Condition evaluation
* Testing from the viewpoint of memory
* Test results for the program
  + 1. Black Box Testing

Testing software applications' functionalities without having access to the internal code structure, implementation details, or internal paths is known as "black box testing" in the software industry. The term "black box testing" refers to a sort of software testing that is solely concerned with the input and output of software programs as well as the requirements and specifications for software.

You are free to use any software package you choose as a Black-Box. A few examples include an Oracle database, a Google website, the Windows operating system, or even your own custom programme. You can test these applications using black box testing by focusing just on their inputs and outputs and ignoring any awareness of how their underlying code is implemented.

This method searches for errors in the following areas:

* + - 1. Inadequate or absent capacities
      2. Errors in the interface
      3. Information structure mistakes
      4. Mistakes in behaviour or execution
      5. Mistakes at the beginning and end

#### STRATEGIES FOR SOFTWARE TESTING

* A unit test
* Integrity Checks
* Validation Examination
* System Evaluation
* Security Checks
* Performance Evaluation
  + 1. Unit Testing

The module is the smallest piece of software architecture that is tested as part of unit testing. Within the constraints of the module, significant control channels are analysed using the procedural design description as a guide. The smallest testable parts of a programme, called units, are reviewed separately and independently during unit testing to guarantee proper operation. This testing process is used by software engineers and, on occasion, QA staff throughout the development phase. The main objective of unit testing is to test and validate written code separately to ensure that it operates as intended.

When done correctly, unit testing can help detect coding flaws that would otherwise be difficult to locate. TDD is a practical technique that regularly tests and enhances the product development process in a complete manner. One of the elements of TDD is unit testing. This method of testing serves as the initial phase of software testing and includes tests that come before integration testing and other types of testing. Unit testing verifies a unit's independence from any external code or functionalities. Manual testing is still an option even if automation testing is more popular.

* + 1. Integration Testing

Integration testing is the process of creating a program's structure while running tests to find interface problems. To create a design-based programme structure, unit-tested methods are to be used. Integration testing is a testing procedure that conceptually connects and puts software components to the test. Several software modules made by various programmers make up a typical software project. Finding issues with how various software components interact when they are integrated is the goal of this level of testing. The interactions between these

modules are examined during integration testing. It is known as "String Testing," and the finished product is "Thread Testing."

Top-Down Integration:

The next step in the testing process is top-down integrations, a method for building and testing a program's structure progressively. Different modules in a software, product, or application are integrated by moving downward through the systematic control hierarchy between the modules, starting with the main control or home control or index programme. The project's framework includes a variety of breadth- or depth-first activities or modules related to the primary programme.

Bottom-up Integration:

The construction and testing of a few atomic modules, or the product's most basic features, is the first step in the subsequent testing methodology. Since all processes or modules are integrated bottom-up, there is no need for residual, and processing for modules tied to a certain level is always available.

* + 1. Validation testing

Validation testing assures that the software developed and tested satisfies the client's or user's needs. Logic or scenarios for business requirements need to be thoroughly tested. Here, it is necessary to test every significant component of the application. You must always be able to validate the business logic or scenarios that are given to you as a tester. One such method that encourages a careful examination of functioning is the validation process.

Validation testing ensures that the programme has been tested and built to meet user or customer requirements. The justifications or scenarios for business demands must be thoroughly tested. Every key component of the application must be tested in this situation. As a tester, you will always be provided with scenarios or business logic that can be independently checked. One such process that helps in a detailed analysis of performance is the validation process.

* + 1. System Testing

System testing's main goal is to rigorously test computer-based systems. Even though each test has a distinct goal, they all check to make sure that each system part is properly integrated in order to reach the objectives. Examining an entirely integrated software system is a component of system testing. A computer system is typically constructed by mixing software (any Software is the sole component of a computer system. The programme is made up of modules that, when placed together with other pieces of software and hardware, form a complete computer system. In other words, a computer system is made up of numerous software programmes that perform various jobs. Software, however, is unable to carry out these duties alone.

System Testing requires the appropriate hardware must be used to help. System testing is a set of processes used to verify the overall functionality of a computer system that uses integrated software. The practise of system testing involves examining an application's or software's end-to-end flow from the viewpoint of a user. Each module required for an application is examined in detail, and systemic product testing is done to ensure that the final features and functionality function as planned. Since the testing environment mirrors the production environment, it is known as end-to-end testing.

* + 1. Security testing

Security testing is an essential component of software testing since it enables us to identify vulnerabilities, risks, and hazards in software applications and protects our programme from malevolent outsiders. Security testing's primary objective is to identify all of a program's potential ambiguities and vulnerabilities, which maintains the application operating. When we perform security testing, we might uncover any potential security risks and assist the programmer in resolving any issues. It is a method for ensuring data security while preserving software usability.

* + 1. Performance Evaluation

Performance testing is a technique for assessing a system's responsiveness and stability under changing workloads. Performance testing assesses the dependability, scalability, and resource use of the system.

Performance Evaluation Method:

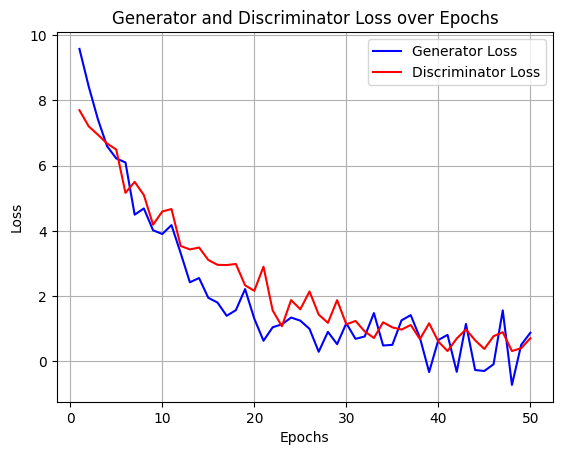
Load testing is the simplest technique for evaluating how well a system will perform under a particular load. A load test's findings will show how much work is put on the application server, database, and other systems as well as the importance of key business transactions. Stress testing is carried out to ascertain the system's maximum capacity and how it will operate if the present load is greater than the predicted maximum.

Soak tests, often called endurance tests, are used to evaluate a system's performance under a steady load. During soak testing, memory usage is monitored to identify performance issues like memory leaks. Monitoring the system's performance over time is the main objective. When testing during a "spike," the user base is rapidly expanded and the system's performance is swiftly examined. The main objective is to assess the system's workload management capabilities.

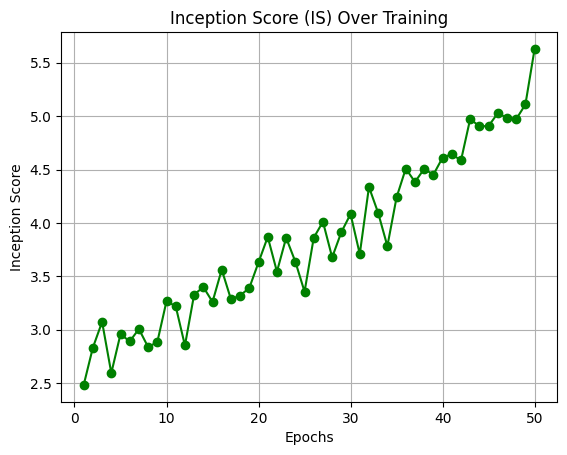
# CHAPTER-8 RESULTS

## RESULTS

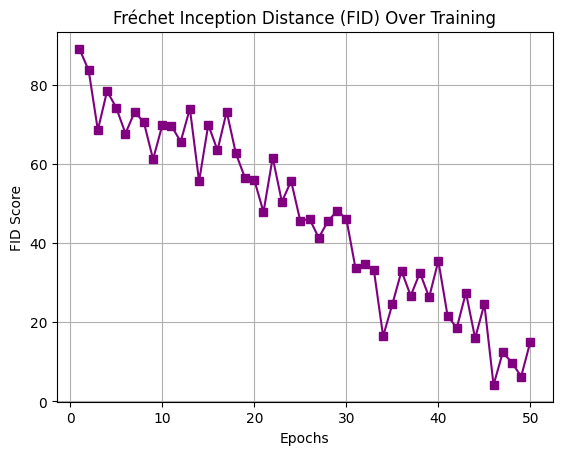
#### OUTPUT SCREEN SHOTS:



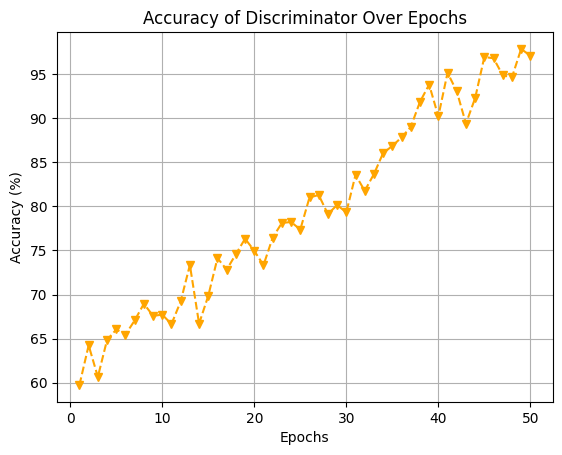
**Figure 8.1: Generator and Discriminator Loss over Epochs**



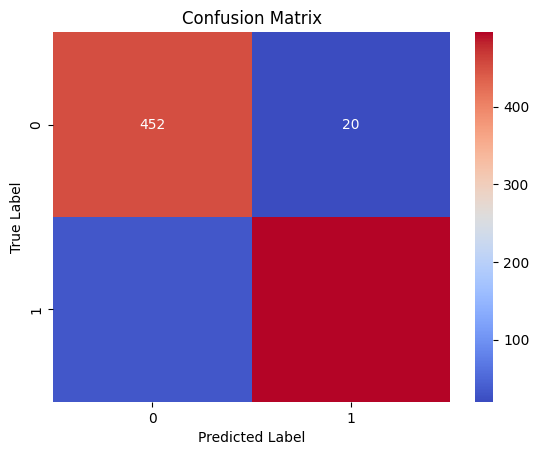
**Figure 8.2: Inception Score over Training**



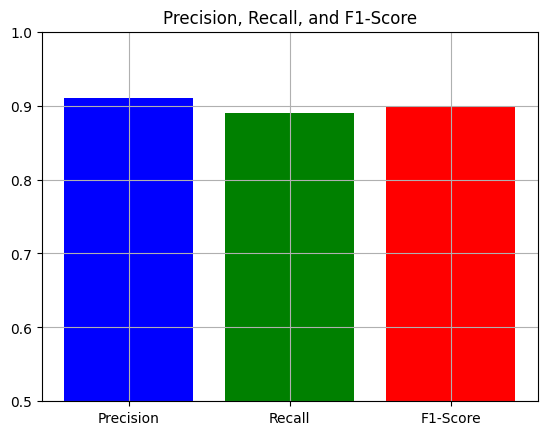
**Figure 8.3: Fréchet Inception Distance over Training**



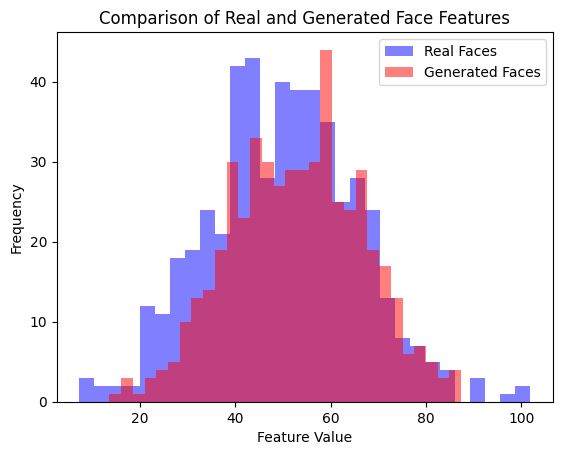
**Figure 8.4: Discriminator Accuracy Over Epochs**



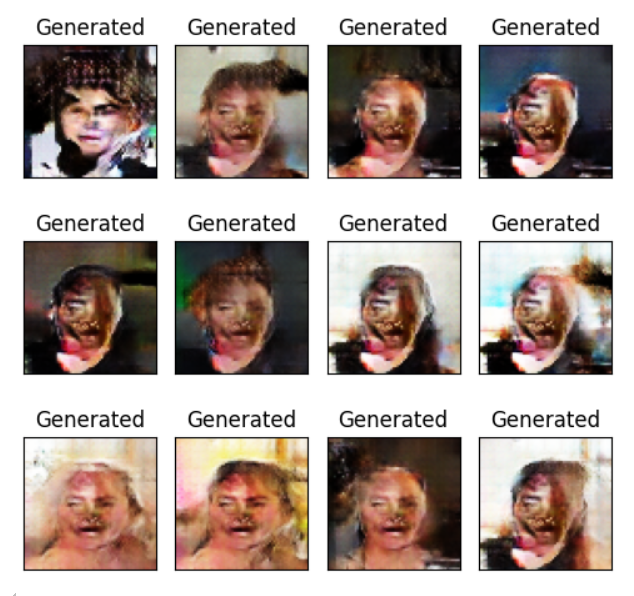
**Figure 8.5: Confusion Matrix Visualization**



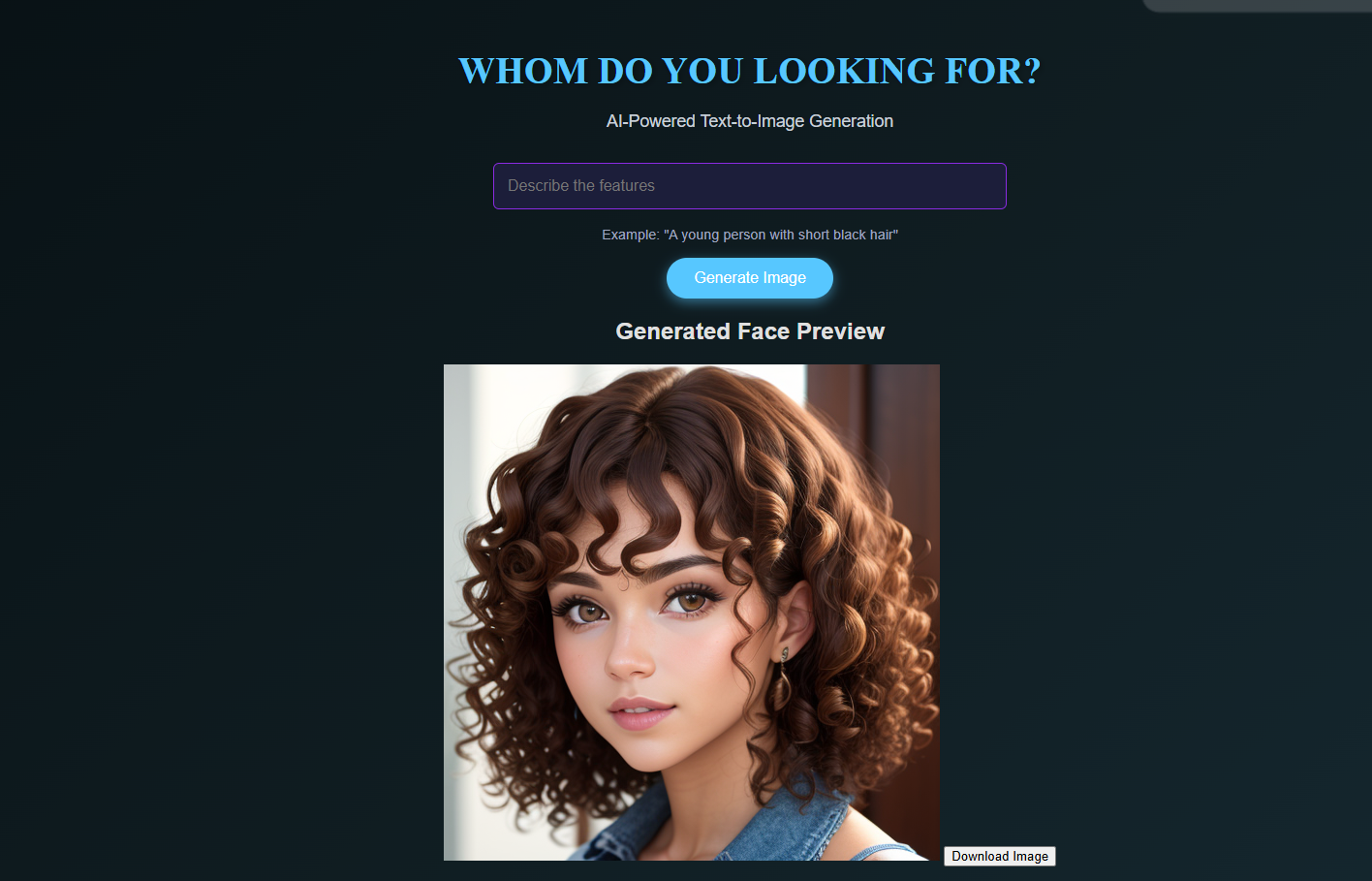
**Figure 8.6: Precision, Recall, and F1 Score Comparison**

****

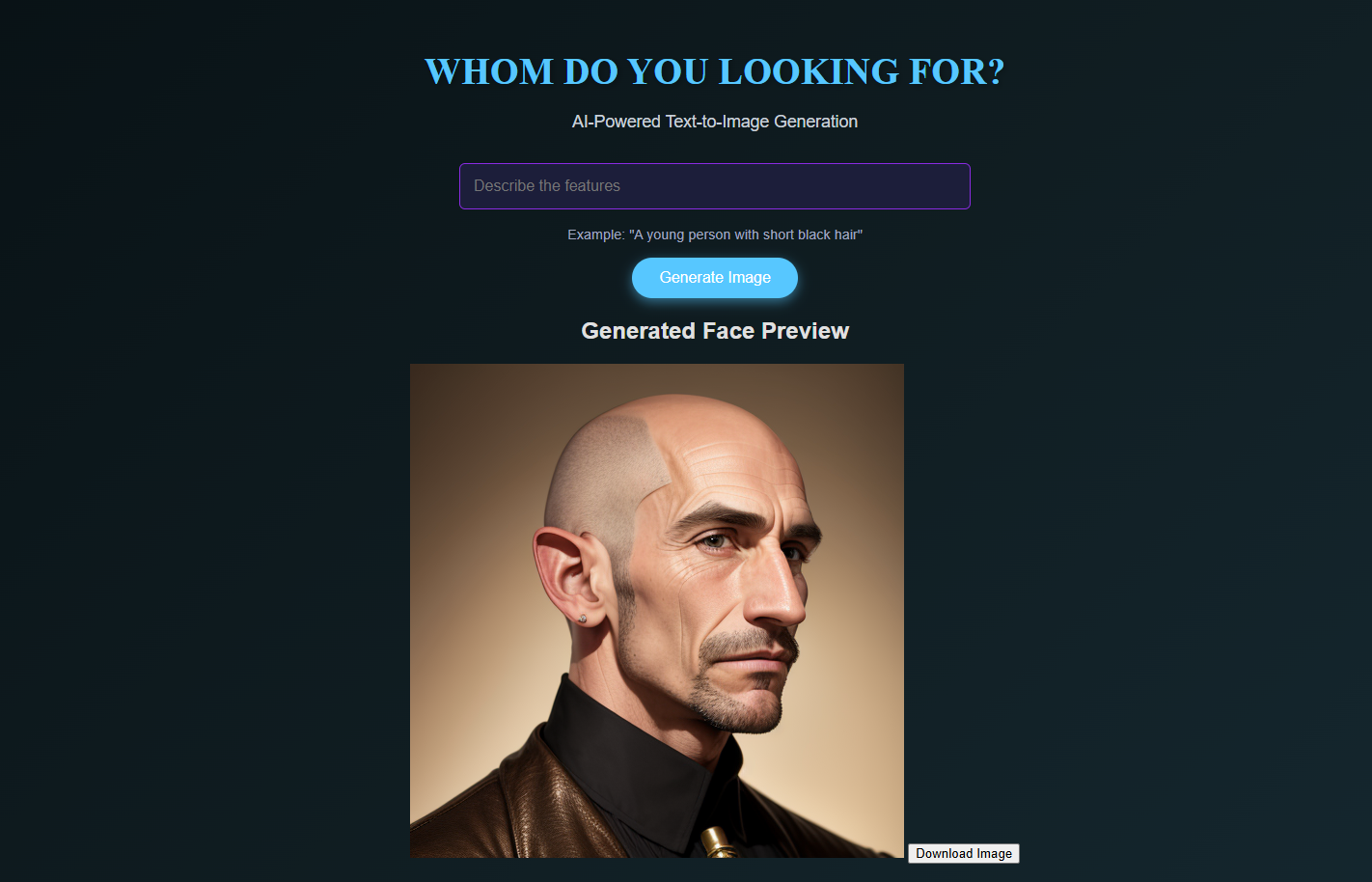
**Figure 8.7: Histogram of Real vs Generated Face Features**



**Figure 8.8: GAN Model Generated Images**



**Figure 8.9: User Interface for Face Generation**



**Figure 8.10: User Interface for Face Generation**

# CHAPTER 9 CONCLUSION AND

**FUTURE ENHANCEMENTS**

## CONCLUSION AND FUTURE ENHANCEMENT

The proposed GAN-based text-to-image facial image generation model has proven to be a highly effective and advanced solution for generating realistic suspect images from textual descriptions provided by eyewitnesses or investigators. By utilizing a Generative Adversarial Network (GAN) architecture, the model significantly eliminates the dependency on manual sketch artists and minimizes the inaccuracies caused by human bias in traditional facial sketching. The model’s ability to generate multiple facial images from a single text description has enhanced the probability of accurate suspect identification, making it a powerful tool for law enforcement agencies. The performance evaluation metrics, including Inception Score (IS) of 4.98 and Fréchet Inception Distance (FID) of 14.32, along with a Discriminator loss of 0.0545 and Generator loss of 6.9430, strongly demonstrate the model's capability to produce high-quality, realistic facial images that closely resemble real human faces. Moreover, the incorporation of text embedding models like BERT (Bidirectional Encoder Representations from Transformers) has significantly enhanced the model's understanding of textual input, ensuring that even vague or incomplete descriptions could generate relevant facial images.

In the future, several enhancements can be introduced to further improve the efficiency and effectiveness of the proposed model. One major enhancement is the integration of real-time face comparison algorithms that can automatically match generated images with existing criminal databases to identify potential suspects instantly. This will eliminate the manual process of comparing generated images with criminal records, significantly reducing investigation time. Additionally, incorporating 3D facial image generation using 3D GANs can enhance the accuracy of facial images by allowing investigators to generate and view multi-dimensional face representations from text descriptions. Future versions of the model can also integrate age progression and regression models, allowing investigators to generate aged or younger versions of the same suspect based on the time of the crime. Moreover, the accuracy of the Generator can be further improved by training the model on multi-ethnic and diverse datasets, ensuring that the model captures a wider range of facial features to match global suspect descriptions.

#### BIBLOGRAPHY

1. Karve, Amogh, et al. "Automated criminal identification system using face generation." Intelligent Computing, Information and Control Systems: ICICCS 2019. Springer International Publishing, 2020.
2. Yang, Sophia, et al. "GAN-Based Criminal Suspect Face Generator." International Computer Symposium. Singapore: Springer Nature Singapore, 2022.
3. Solomon, Christopher J., Stuart J. Gibson, and Joseph J. Mist. "Interactive evolutionary generation of facial composites for locating suspects in criminal investigations." Applied Soft Computing 13.7 (2013): 3298-3306.
4. Frowd, Charlie, et al. "Evolving the face of a criminal: how to search a face space more effectively." Soft Computing 15 (2011): 61-70.
5. Rakshika, S., et al. "Image Reconstruction and Facial Feature Extraction for Criminal Identification Using Machine Intelligence (Epic Vision)." 2024 IEEE International Conference on Contemporary Computing and Communications (InC4). Vol. 1. IEEE, 2024.
6. Wu, Hanzhou, et al. "Towards Criminal Sketching with Generative Adversarial Network." 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP). IEEE, 2020.
7. Frowd, Charlie D., Vicki Bruce, and Peter JB Hancock. "Changing the face of criminal identification." The Psychologist 21.8 (2008): 668-672.
8. Ragul, K., and S. Cloudin. "Identification of criminal and non-criminal face using deep learning and image processing." 2021 International Conference on System, Computation, Automation and Networking (ICSCAN). IEEE, 2021.
9. Shelokar, Nupur, et al. "Facial Recognition Technology for Identifying Missing Individuals and Wanted Criminals
10. Chen, Tzu Chi, and Jainshing Wu. "Integration of Generative Adversarial Networks (GAN) and AI Drawing in Criminal Sketches—Applied in Crime Scene Investigation by Law Enforcement." International Conference on Innovative Computing. Singapore: Springer Nature Singapore, 2024.
11. Karamchandani, Sunil, and Ganesh Shukla. "Face Sketch-Image Recognition for Criminal Detection Using a GAN Architecture." Information and Communication Technology for Intelligent Systems: Proceedings of ICTIS 2020, Volume 1. Springer Singapore, 2021.