

**EXPERT SYSTEM FOR VISUALLY**

**CHALLENGED USING AI**

by

Adarsh Verma (2147102) Amber Ujjwal Linda (2147104) A nkur Sharma(2147105)

Under the guidance of

**Dr. THIRUNAVUKKARASU V**

A Project report submitted in partial fulfilment of the requirements for the award of the degree of Master of Computer Applications of CHRIST (Deemed to be University)

Dec - 2022



CERTIFICATE

*This is to certify that the report titled* ***EXPERT SYSTEM FOR VISUALLY CHALLENGED USING AI*** *is a bona fide record of work done by*Adarsh Verma (2147102), Amber Ujjwal Linda (2147104) , A nkur Sharma(2147105) *of CHRIST (Deemed to be University), Bengaluru, in partial fulfillment of the requirements of V Trimester MCA during the year 2022.*

## Head of the Department Faculty In charge

Valued-by: Names : Adarsh Verma Amber Ujjwal Linda

Ankur Sharma

Register Number(s) : 2147102

2147104

2147105

Examination Centre : CHRIST (Deemed

to be University)

Date : 15/12/2022

# ACKNOWLEDGEMENTS

I would also like to express my gratitude to our Vice-Chancellor, Dr. Fr. Abraham Vettiyankal Mani, Pro-Vice Chancellor Dr.Fr. Joseph CC, our Head of Department(HOD), Dr. Ashok Immanuel V, Coordinator Dr. Shoney Sebastian and the faculty for providing all the required facilities to accomplish the project.

I would like to convey our gratitude to our project guide, Dr. Thirunavukkarasu V, for giving us a constant source of inspiration and help in preparing the project, personally correcting our work and providing encouragement throughout the project.

Finally, I would like to thank my parents and friends, without them this assignment would not have been completed.

Ankur Sharma - 2147105 Adarsh Verma - 2147102

Amber Ujjwal Linda - 2147104

# ABSTRACT

Image Translation is a vital Machine learning feature used in many ways in today's world to create images from scratch or enhance noisy images. The topic intends to help the police and forensics get accurate images of criminals with a minimal number of records, as it can help the police get images of how they might look in the present time using older data.

Image Aging is the process of de-aging or post-aging the object's age in the image. This project's object is supposed to be a human facial structure. When an image like a portrait photograph, selfie, or passport image is to be used as an input, the Output is supposed to be a post-aged image of the information.

This project was built using Cycle Generative Adversarial Network or CGAN is a deep convoluted neural network which is used for training models to perform image-to-image translation. The architecture involves training two generator models and two discriminator models simultaneously. The generator is used to create images for the first domain. The discriminator model uses generated images from Generator and genuine photos from Domain to determine if they are real or fake. Using an unpaired dataset, the network learns the mapping between the input and output images.

The project concludes with the facts and Output, which can be helpful not only for law agencies and other institutions but also for scientific research, clinical practice, weather forecasting, and filmmaking.

Image Aging using CycleGAN

Department of Computer Science, CHRIST (Deemed to be University)



|  |  |
| --- | --- |
| **TABLE OF CONTENT** |  |
| Acknowledgments | iii |
| Abstract | iv |
| List of Tables | vii |
| List of Figures | viii |
| 1. Introduction | 1 |
| 1.1 Project Description | 1 |
| 1.2 Existing System | 1 |
| 1.3 Objectives | 1 |
| 1.4 Purpose, Scope, and Applicability | 2 |
| 1.4.1 Purpose | 2 |
| 1.4.2 Scope | 2 |
| 1.4.3 Applicability | 2 |
| 1.5 Overview of the Report | 3 |
| 2. System Analysis and Requirements | 4 |
| 2.1 Problem Definition | 4 |
| 2.2 Requirements Specification | 5 |
| 2.3 Block Diagram | 6 |
| 2.4 System Requirements | 7 |
| 2.4.1 User Characteristics | 7 |
| 2.4.2 Software and Hardware Requirements | 7 |
| 2.4.3 Constraints | 10 |
| 2.5 Conceptual Models | 11 |
| 2.5.1 Data Flow Diagram | 11 |
| 3. System Design | 13 |
| 3.1 System Architecture | 13 |
| 3.2 Module Design | 16 |
| 3.3 Interface Design and Procedural Design | 18 |
| 3.3.1 User Interface Design | 18 |
| 4. Implementation | 19 |

Image Aging using CycleGAN

Department of Computer Science, CHRIST (Deemed to be University)



|  |  |  |
| --- | --- | --- |
|  | 4.1 Implementation | 19 |
| 4.2 Coding Standard | 19 |
| 4.3 Coding Details | 20 |
| 4.4 Screen Shots | 26 |
| 5. | Testing | 29 |
|  | 5.1 Test Cases | 29 |
|  | 5.2 Testing Approaches | 30 |
|  | 5.3 Test Reports | 31 |
| 6. | Conclusion | 32 |
|  | 6.1 Design and Implementation Issues | 32 |
|  | 6.2 Advantages and Limitations | 32 |
|  | 6.3 Future Scope of the Project | 33 |

References



# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Fig no.** | **Fig name** | **Page no.** |
| 1.1 | Block Diagram | 6 |
| 2.1 | DFD-0 | 11 |
| 2.2 | DFD-1 | 12 |
| 3.1 | Model Architecture | 15 |
| 3.2 | Generator | 16 |
| 3.3 | Discriminator | 17 |
| 3.4 | Interface | 18 |
| 4.1 | Main Dialog Box | 26 |
| 4.2 | Image selection | 27 |
| 4.3 | Output | 27 |
| 4.4 | About Us | 28 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table no.** | **Title** | **Page No.** |
| 1.1 | Dependencies | 7 |
| 2.1 | Test Cases | 29 |
| 2.2 | Test Report | 31 |

# INTRODUCTION



This chapter gives a brief description, objective and scope the system proposed. The chapter also talks about the existing systems and the limitations of that.

## PROJECT DESCRIPTION

The project aims to develop a system that gives a post-aged image of a person. The project takes an image of proper and particular containing a human face, transforming it into an aged version of itself and producing an image to represent the transformation [1]. The project uses the latest AI/ML frameworks & algorithms to achieve this. The process happening in the project can be termed 'Image-to-Image Translation,' i.e., the task of taking images from one domain and transforming them, so they have the style (or characteristics) of images from another domain. The algorithm for performing image-to-image translation is called GAN, specifically cGAN(cycle Generative Adversarial Network). The project was trained based on UTK and CACD datasets.

## EXISTING SYSTEM

A similar kind of system exists in Social Media sites and Apps in the form of filters and extensions.[2] Mostly android apps use this system as there is easy access to cameras and phone storage in real-time.

Some popular Examples: are Snapchat & Instagram.

* + 1. Limitations of The Existing System

The existing use of this is limited to the fun stuff and social media activities. And the Output is further engineered to produce an over-morphed visual that may be inaccurate.

The proposed system aims to remove any additional anomalies and provide Output with high-accuracy detailing and gimmicks.

## OBJECTIVE

The objective of the project is to tackle the problem statement mentioned with help of using AL/ML (CGAN), by performing image-to-image

upon the dataset. The objective is also to train a proper model from the dataset that can successfully perform translation. By fulfilling the above requirements, the user can input an image and can get an age transformed output.



The project should be able to achieve the following transforming upon not only a single input but multiple different inputs.

## PURPOSE, SCOPE, AND APPLICABILITY

* + 1. Purpose
       1. The project aims to help officials keep the digital pictorial records of people on the red and yellow list up to date.
       2. This project can be implemented in many places where the user needs to get an age-transformed image of a person. Example - To keep the image database of people up to date with the latest records wherever possible. (iii). Since there is no current system for this kind of use, this project will improve the current system records by giving an approximate image according to the current time or future.
    2. Scope

To get an accurate result from the trained model we assume that the

image of the person is clear, the face of the individual is not tampered i.e., he has not undergone any plastic surgery, his face is not damaged.

* + 1. Applicability

We define the applicability of this project more in-line with entities dealing with crime and investigations, mostly federal and police institutions. The model can help federal entities in investigation by providing them a robust application which they can rely upon to gain crucial information relating to individuals involved in the investigation. Especially if the investigative operations last for longer duration. In such cases the individual involved would have aged significantly leading to ambiguity in



the operation. Using the model the agents can keep in track the age

progression of the individual.

## OVERVIEW OF THE REPORT

The report sticks to a linear path and contents are aligned accordingly. All the components and subcomponents are clear and well defined, and modularized in the proper way.

The **First** chapter is the **Project Introduction**, it contains submodules as Project Description, Project Description, Existing System, Objectives, Scope and Applicability which introduces the project aim and what it tends to achieve.

The **Second** chapter is **System Analysis and Requirements,** it contains **Problem Definition**, **Requirements Specification**, **Block Diagram**, **System Requirements, User Characteristics**, **Software and Hardware Requirements, Constraints** and **Conceptual Models.** This part defines the problem statement the project aims to tackle and all other decisions and plan of action taken in accordance with it. It also mentions a very important component: the System requirements which are very important with respect to any project.

The **Third** chapter is **System Design,** it encompasses all the important criteria related to making the model such as **System Architecture, Module Design** and **Interface and Procedural Design.**

The **Fourth** chapter is the **Implementation,** which consists of **Implementation Approaches**, **Coding Standard** and **Coding Details.** This part is the development part of the project which requires technical and programming knowledge. Here the training and testing of the model is done after coding. The logical base of the code will impact every aspect of the project, hence making it a very important module.



The **Fifth** chapter is the **Testing**, this part comes after the implementation or coding

phase of the project. This phase makes sure that the components designed and implemented are all working properly by writing test cases and performing tests intensively. The model consists of **Test Cases** and **Testing Approaches.**

The **Sixth** and the last is **Conclusion**. It comprises of **Design and Implementation Issues, Advantages, Limitation** and **Future Scope of the Project.**

# SYSTEM ANALYSIS AND REQUIREMENTS

System Analysis and Requirements is the second chapter, and it includes the following: Problem Definition, Requirements Specification, Block Diagram, System Requirements, User Characteristics, Software and Hardware Requirements, Constraints, and Conceptual Models. This section outlines the issue statement the project will attempt to solve as well as any further choices and plans of action made in response to it. It also makes note of a crucial element: the system requirements, which are crucial for each project.

## PROBLEM DEFINITION

The development goal is to produce a transformed image of the provided data set. As the problem with visual data formats is that they remain the same, an image taken at a particular period will remain the same throughout time. You cannot determine the current form of the entities inside.

The project aims to tackle this problem by creating a model using the latest AI/ML frameworks, which can convert a given image or set of images into age- transformed data. Different individuals can use this model in different domains to successfully transform several types of visual data into useful information. Organizations such as the FBI, CBI, Intelligence, and Police. Medical Sciences can use this to predict future results of certain cases. Individuals can also use it to get an idea about their future selves.

## REQUIREMENTS SPECIFICATION



* + 1. Functional Requirements

**Dataset:** The Dataset.py module contains important pre-processing parameters. This is the first segment of the entire operation. It is also responsible for the segregation of data. It accepts only particular data formats (JPG or PNG). The separated data are stored in respective folders.

**GAN Module:** The GAN module contains important data loading, configuration, optimization, and training classes. The training steps include no. of workers, epoch, and batch size. All of it is done using CycleGAN. The Output is stored in a separate .pth file(state\_dict.pth), which saves the trained model for future use.

**Model:** The model.py file contains the very important **Generator** and

**Discriminator** class. It is the backbone of the entire GAN framework. **Generator:** The generator part of a GAN learns to create fake data by incorporating feedback from the discriminator. It knows to make the discriminator classify its Output as accurate.

**Discriminator:** A GAN's discriminator is only a classifier. It makes an effort to discriminate between data that is accurate and data that was generated. It employs a network design suitable for the classification of the data.

The training data for the discriminator originates from two sources:

**Real Data:** instances of real data, photographs of real individuals. These events are used as encouraging examples by the discriminator during training.

**Fake Data:** examples of fake data that the generator has produced. These situations are used as negative examples by the discriminator during training.

* + 1. Non-Functional Requirements



## Constraints

The Only Constraints in the working system are regarding image type. The image should be in proper format (JPG or PNG), and the image should necessarily consist of a human face of 215 x 215 minimum. However, the Image resolution can be higher.

## Response time

The response time is limited to 1-2 sec. Depending on the inputs from the user, this may increase by a few mili-seconds but does not hamper the overall performance.

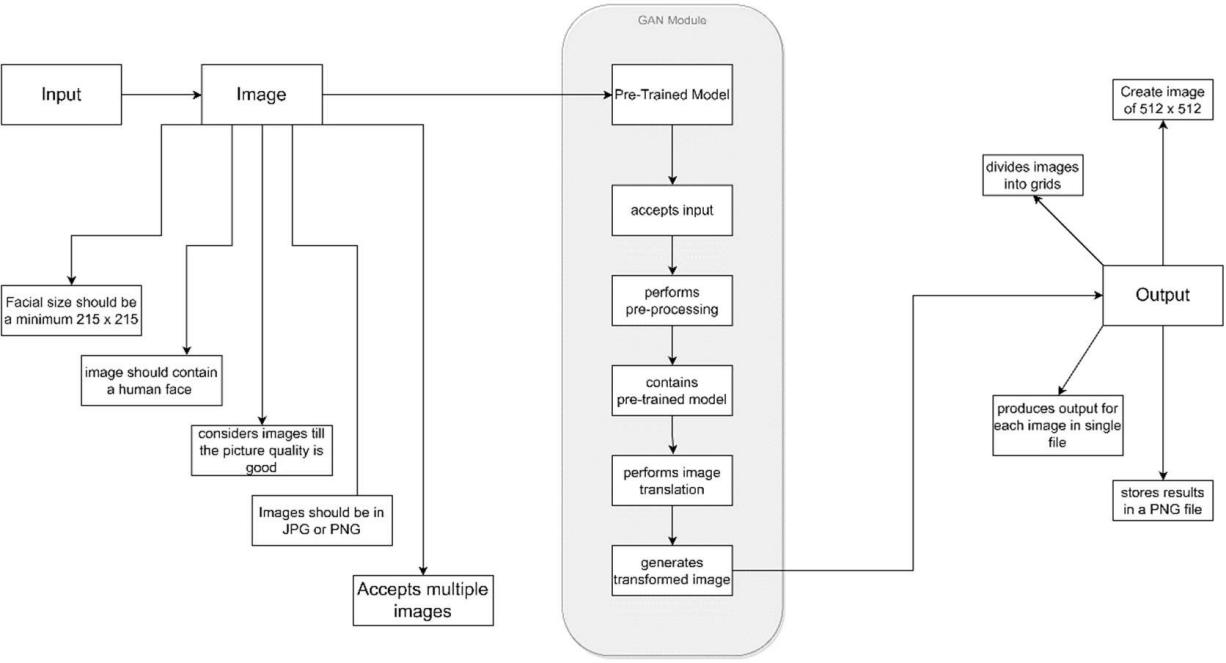
## Reusability

The trained model is robust; no matter the number of uses, the Output remains consistent throughout.

## Capacity:

No limitations are present currently in the working system. The user can input n- number of inputs and receive proper Output.

## 2.3 BLOCK DIAGRAM



**Fig 1.1 Block Diagram**

## SYSTEM REQUIREMENTS



* + 1. User Characteristics

Our model is supposed to cater to a range of users, but the underlying characteristics are the same for all the users, i.e the character should have a minimum configured system and an image data to work with the model. Apart from that the image data should have a proper facial identity that should be minimum of 215x215 sized and be in a proper format. Barring this criteria the system will reject the input from the user.

* + 1. Software and Hardware Requirements
       1. Software and Hardware Requirements for training a model The Dataset used is highly demanding in terms of system requirements. The minimum system requirements can take approximately 100 hrs.

## Software Requirements

* + - * + OS: Windows 10
        + Compiler: G++ 4.x.x
        + Python: Python 3.10
        + GCC: 12.2
        + Dependencies:

|  |
| --- |
| absl-py==1.0.0 |
| aiohttp==3.8.1 |
| aiosignal==1.2.0 |
| async-timeout==4.0.2 |
| asynctest==0.13.0 |
| attrs==21.4.0 |
| cachetools==5.0.0 |
| certifi==2021.10.8 |
| charset-normalizer==2.0.12 |
| colorama==0.4.4 |
| cycler==0.11.0 |
| fonttools==4.29.1 |

|  |
| --- |
| frozenlist==1.3.0 |
| fsspec==2022.2.0 |
| future==0.18.2 |
| google-auth==2.6.0 |
| google-auth-oauthlib==0.4.6 |
| grpcio==1.44.0 |
| idna==3.3 |
| importlib-metadata==4.11.2 |
| kiwisolver==1.3.2 |
| Markdown==3.3.6 |
| matplotlib==3.5.1 |
| multidict==6.0.2 |
| numpy==1.21.5 |
| oauthlib==3.2.0 |
| packaging==21.3 |
| Pillow==9.0.1 |
| protobuf==3.19.4 |
| pyasn1==0.4.8 |
| pyasn1-modules==0.2.8 |
| pyDeprecate==0.3.1 |
| pyparsing==3.0.7 |
| python-dateutil==2.8.2 |
| pytorch-lightning==1.5.10 |
| PyYAML==6.0 |
| requests==2.27.1 |
| requests-oauthlib==1.3.1 |
| rsa==4.8 |
| six==1.16.0 |
| tensorboard==2.10.1 |
| tensorboard-data-server==0.6.1 |
| tensorboard-plugin-wit==1.8.1 |
| torch==1.11.0 |
| torchmetrics==0.10.0 |
| torchvision==0.13.1 |
| tqdm==4.63.0 |

|  |
| --- |
| typing-extensions==4.1.1 |
| urllib3==1.26.8 |
| Werkzeug==2.0.3 |
| yarl==1.7.2 |
| zipp==3.7.0 |

**Table no**. 1.1

Here are the **Minimum System Requirements**:

* CPU: i5 10th Gen or Ryzen 5 3rd Gen (o r Above)
* CPU SPEED: 2.5GHz (or Above)
* RAM: 8 GB
* OS: Any
* VIDEO CARD: Dedicated
* GPU: GTX 1650 (or Above)
* STORAGE: 120 GB (SSD Recommended)

## Recommended System Requirements:

* CPU: i7 10th Gen or Ryzen 7 5th Gen (or Above)
* CPU SPEED: 3.7GHz (or Above)
* RAM: 16 GB
* OS: Any
* VIDEO CARD: Dedicated
* GPU: RTX 3rd Gen (or Above)
* STORAGE: 512 GB (SSD Recommended)
  + - 1. Software and Hardware Requirements for the trained model

## Hardware Requirements

Here are the **Minimum System Requirements**:

* CPU: Dual Core
* CPU SPEED: 1 GHz
* RAM: 2 GB
* OS: Any
* VIDEO CARD: Integrated
* STORAGE: 8 GB

## Software Requirements

* + OS: Windows XP
  + Compiler: GCC 6.5
  + Python: Python 3.6.8
    1. Constraints

Hardware Limitations: Requires high-end computing infrastructure, including both high-configuration CPU & GPU with the latest updated drivers. Not fulfilling, which will lead to high training time. Language Requirements: Requires support of python and all AI/ML libraries and packages. Not fulfilling, which will lead to the non- functioning of the model.

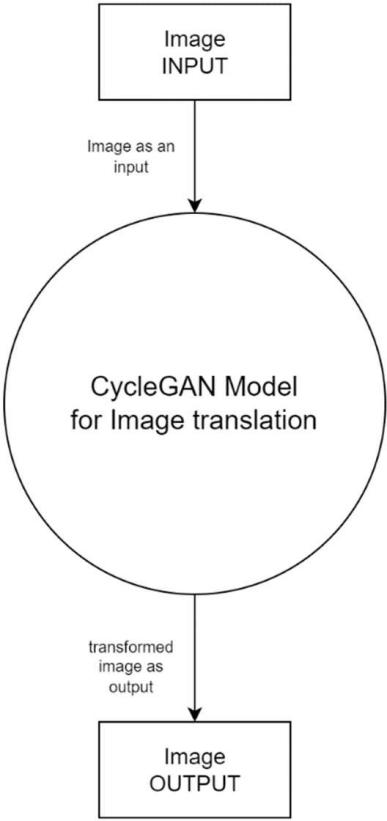
Reliability Requirements: Since the project deals with images of people, the privacy issue must be considered while creating a reliable environment.

Now that you have understood the problem domain and the requirements, produce a model of the system, reflecting the operations/functions that can be performed on/by the system and the allowable sequence of these operations. Conceptual Models can include Context Diagram, Data Flow Diagrams (At least 0 and 1 Level), Entity Relationship (ER) Diagrams, Class Diagrams, System Flow Charts, etc.

## CONCEPTUAL MODELS



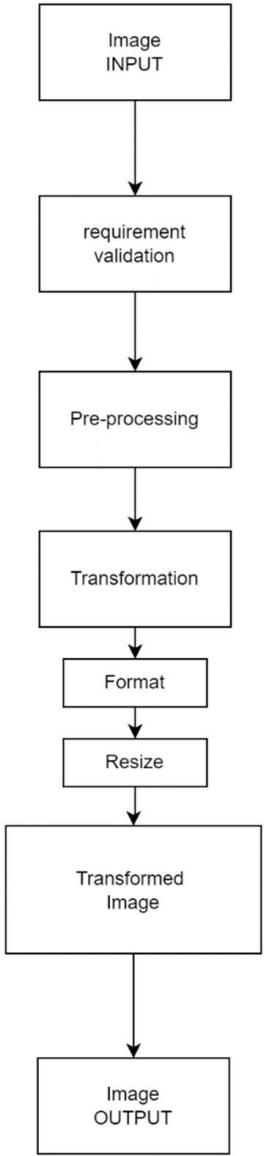
* + 1. Data Flow Diagram
* The base model can be conceptualized by using subsystems of CGAN, consisting of a Generator and Discriminator with specific purposes.
* In the first step image as an input is provided.
* The next step contains requirement validation, which checks If the image satisfies the required criteria or not
* Next comes the pre-processing stage, where the image undergoes certain pre-processing parameter.
* Then the image undergoes the transformation phase, under which it gets transformed and resized into a particular resolution, and the image format is also changed into a particular format.
* In the last step, an output is provided to the user in the form of a transformed image.



**Fig 1.2** DFD level-0



The Data Flow Diagram of level 0 represents data flow within the model. As level-0 represents an abstract view hence the entire transformation module under a single structure. The only data flow is input as an image and output as a transformed image.



**Fig 1.3** DFD level-1

The Data Flow Diagram of level 1 represents the proper data flow within the entire structure. The transformation part is subdivided into its constituent modules.

# SYSTEM DESIGN



This chapter contains the various designs related to the system that encompass all the sub-modules and subsystems of the main system. It also covers the system architecture, data flow, and other design aspects.

## SYSTEM ARCHITECTURE

The system architecture can be divided into two parts: the training and the trained. The upper half of the system architecture is the system's training component, which contains several modules.

**Image Dataset** :The UTKFace dataset is a large-scale face dataset, with a long age span with annotations for age, gender, and ethnicity that make up the dataset.

**Name**: UTKFace **Type**: Image Dataset **Format**: JPG

**Number of Images**: 27000 **Dimensions**: 200x200 **Color space**: RGB

**Span**: 0-116 years old

**Pre-processing** is an important module with several pre-processing parameters. It further ensures the images follow the proper parameters before reaching the main algorithm module.

**GAN:** Then comes the most important part of the system: the GAN module.

We have two collections of photographs, and they are unpaired, meaning they are photos of different faces at different ages; we don't have the same facial structure in older ages and younger ages.

* **Collection 1**: Photos of younger ages.
* **Collection 2**: Photos of older ages.

We will develop an architecture of two GANs, each with a discriminator and a generator model, meaning there are four models in total in the architecture.

The first GAN will generate photos of winter given photos of summer, and the second GAN will generate photos of younger ages given photos of older ages.

* **GAN 1**: Translates photos of younger ages (collection 1) to older ages (collection 2).



* **GAN 2**: Translates photos of older ages (collection 2) to younger ages (collection 1).

Each GAN has a conditional generator model that will synthesize an image given an input image. And each GAN has a discriminator model to predict how likely the generated image is to have come from the target image collection. A GAN's discriminator and generator models are trained under normal adversarial loss like a standard GAN model.

We can summarize the generator and discriminator models from GAN 1 as follows:

## Generator Model 1:

* + **Input**: Takes photos of younger ages (collection 1).
  + **Output**: Generates photos of older ages (collection 2).

## Discriminator Model 1:

* + **Input**: Takes photos of older ages from collection two and Output from Generator Model 1.
  + **Output**: The likelihood of the image is from collection 2. Similarly, we can summarize the generator and discriminator models from GAN 2 as follows:

## Generator Model 2:

* + **Input**: Takes photos of older ages (collection 2).
  + **Output**: Generates photos of younger ages (collection 1).

## Discriminator Model 2:

* + **Input**: Takes photos of younger ages from collection one and Output from Generator Model 2.
  + **Output**: The likelihood of the image is from collection 1.

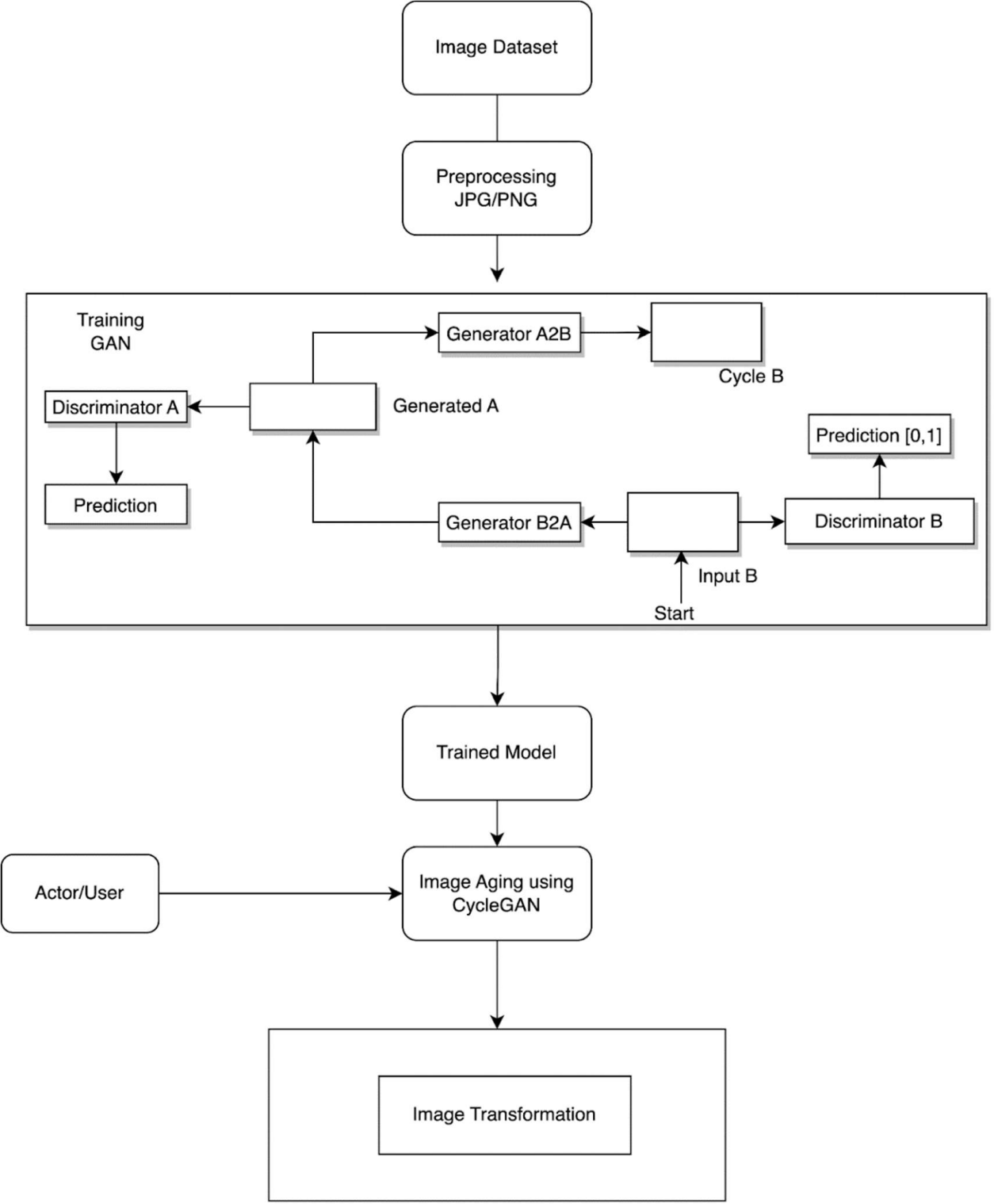
After the GAN module is finished performing training, we are left with a **trained model.**

After this, we can move to the later half of the architecture, where we deal with the user's involvement. Since the model is trained, the user can use this trained

model to provide it with input in the form of images. The pretrained model will work upon the input from the user side and perform transformation operations upon it.



In the end, the user is provided with an output, a transformed version of the input image that the user can access.



**Fig 3.1 Model Architecture**

## MODULE DESIGN



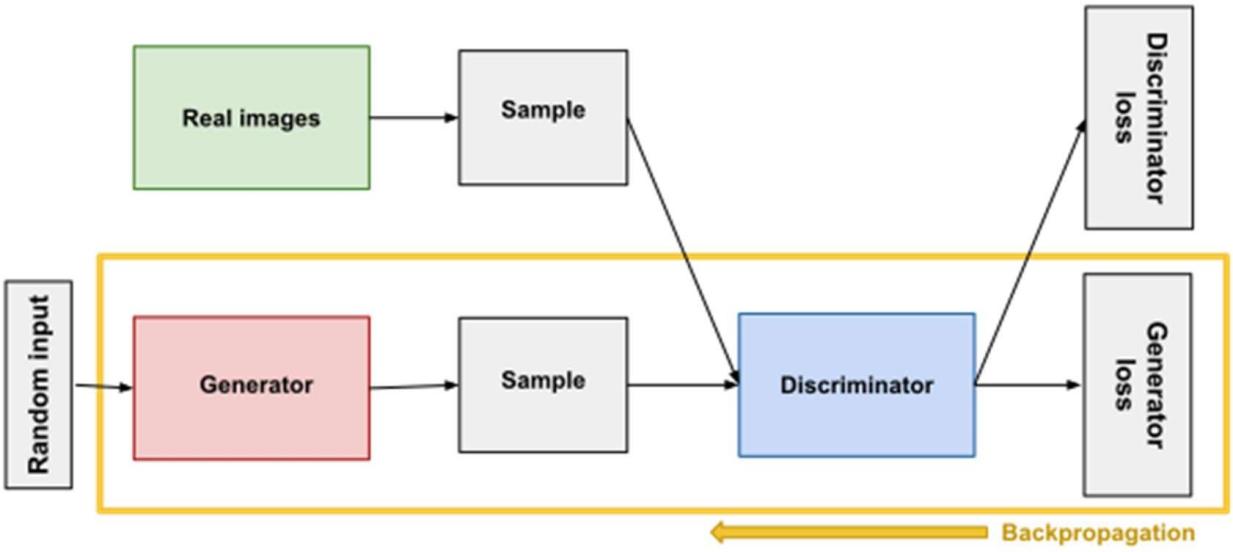
The main algorithm used here, i.e., CycleGAN, comprises two major subsystems which formulate two separate modules. Namely Generator and Discriminator, both of which have their role and function.

## Generator Model

By incorporating feedback from the discriminator, the generator component of a GAN learns to produce fictitious data. It gains the ability to get the discriminator to label its Output as real. Compared to discriminator training, generator training necessitates a tighter integration between the generator and the discriminator.

The GAN's generator training section consists of the following:

* + - 1. various input
      2. generating network that turns the input into a data instance from randomness
      3. network discriminator, which categorizes the generated data
      4. discriminator output
      5. generator loss, a punishment for the generator's inability to trick the discriminator



## Fig 3.2 Generator

* + 1. **Discriminator Model**



A GAN's discriminator is only a classifier. It tries to discern between actual data and data generated by the generator. Any network design suitable for the classification of the data could be used.

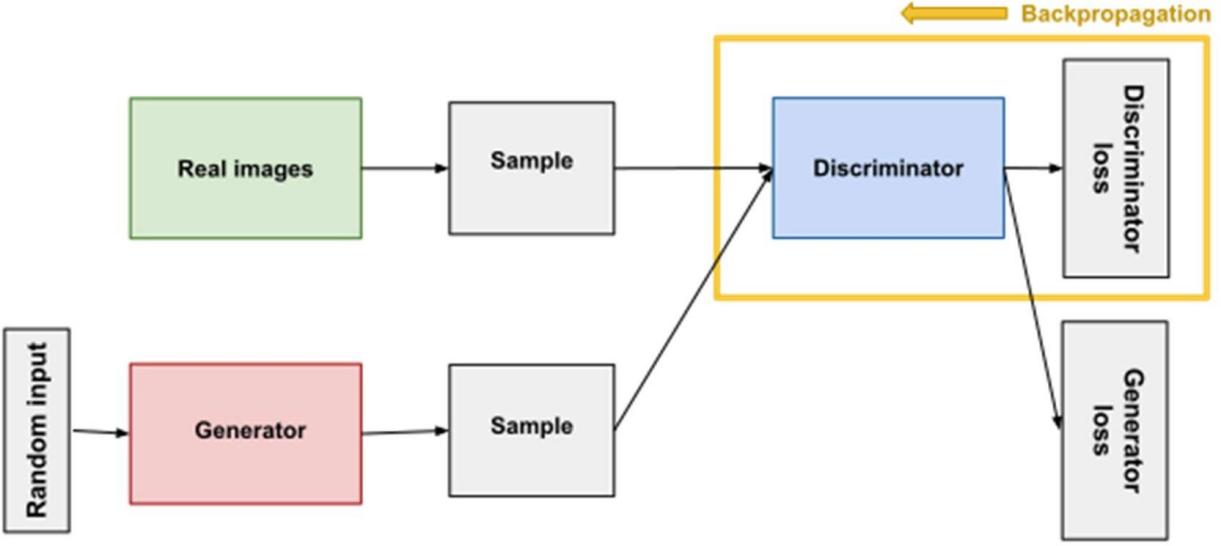
The training data for the discriminator originates from two sources:

**Real Data:** instances of genuine data, such as actual photographs of people. These events are used as encouraging examples by the discriminator during training.

**Fake Data:** examples of fake data that the generator has produced. These situations are used as negative examples by the discriminator during training.

When developing discriminators:

* + - 1. The discriminator separates the generator's actual data from its phony data.
      2. The discriminator who incorrectly labels a real instance as fake or a fake instance as real is penalized by the discriminator loss.
      3. The discriminator updates its weights through backpropagation from the discriminator loss through the discriminator network.



## Fig 3.3 Discriminator

* 1. **INTERFACE DESIGN AND PROCEDURAL DESIGN**

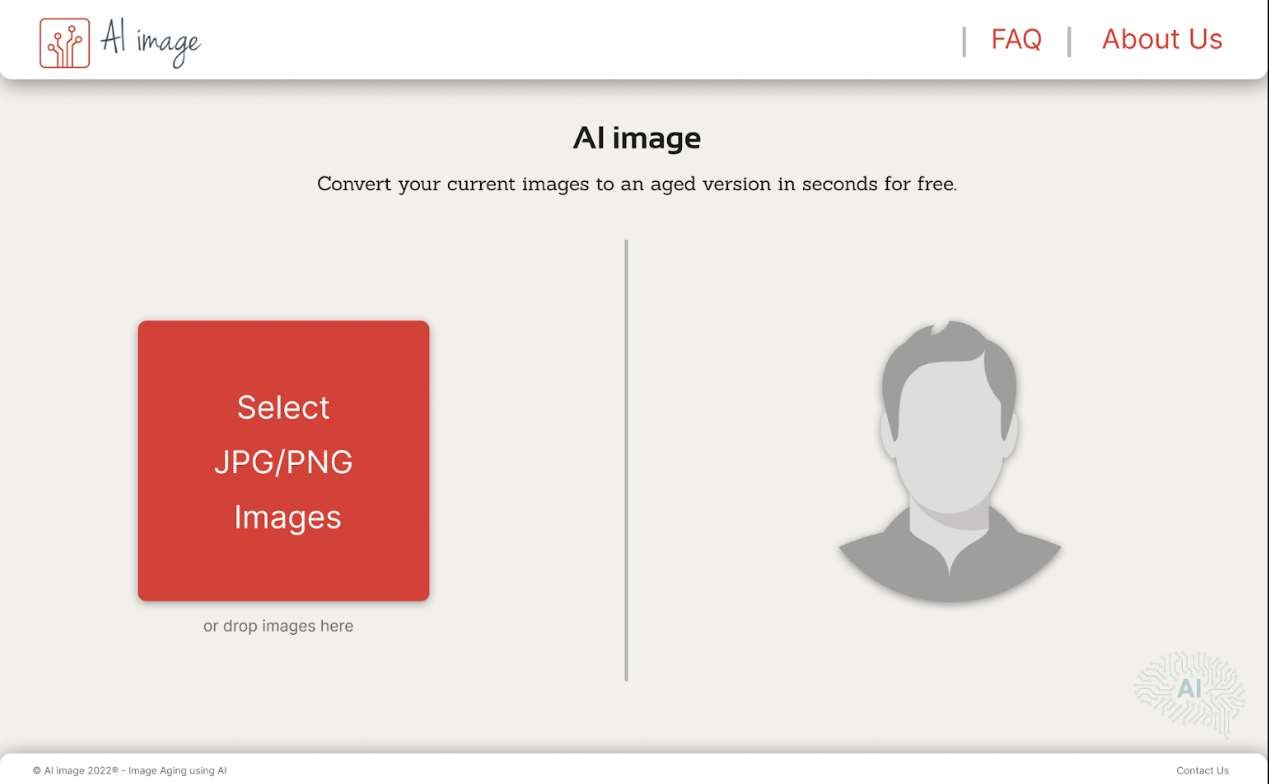


The UI incorporated here was made by keeping in mind the user perspective; the Interface's theme, design, and components have a minimalist tone. While creating the UI, the latest interfaces in similar domains were reviewed for reference.

## USER INTERFACE DESIGN:

The Interface follows very simple and user-friendly operations. It divides the page into two sections upload and download. Users can upload a file from their system or drag and drop it. The download section will be active after the image is successfully transformed. Once active, the user can download the images right into their system.

The Interface also contains common standard options like

* FAQ: for more user-friendliness. (Top Right)
* About us: to know more about this project. (Top Right)
* Contact us: In case the user wants to establish contact. (Bottom Right)
  1. **Interface**

# IMPLEMENTATION



Implementation, which is divided into Implementation Approaches, Coding Standards, and Coding Details, is covered in this chapter. This part is the development part of the project which requires technical and programming knowledge. After coding, the model is trained and tested in this case. The logical foundation of the code will have an impact on all areas of the project, making it a crucial module.

## IMPLEMENTATIONS APPROACHES

The implementation approach opted towards converting the objectives into an actual working model is the Agile approach. An agile approach is a value-centered approach to project management; the agile project management process allows for processing projects in discrete stages or cycles. The approach is adaptable, and projects with dynamic characteristics would benefit from it. Project managers working in this environment would treat milestones as "sprints" to continuously adapt to sudden changes brought on by client feedback. Small software projects with highly collaborative teams or requiring frequent iterations are best suited for it.

## CODING STANDARD

* + - Proper coding standards were followed during the coding and implementation stage.
    - Since the project is AI/ML oriented, the entirety of the code is written using python3.
    - Comments were added at every juncture within the code.
    - The code was properly modularized by grouping the tasks into proper well- defined functions with adequate spacing for each module to improve the readability of the code.
    - The program was properly structured to improve follow-ups.
    - Naming conventions were implemented to identify each component easily.
    - Error and Exception Handling at every critical section of code for error handling.



## CODING DETAILS

def training\_step(self, batch, batch\_idx, optimizer\_idx): real\_A, real\_B = batch

if optimizer\_idx == 0: # Identity loss

# G\_A2B(B) should equal B if real B is fed same\_B = self.genA2B(real\_B)

loss\_identity\_B = F.l1\_loss(same\_B, real\_B) \* self.hparams['identity\_weight']

# G\_B2A(A) should equal A if real A is fed same\_A = self.genB2A(real\_A)

loss\_identity\_A = F.l1\_loss(same\_A, real\_A) \* self.hparams['identity\_weight']

# GAN loss

fake\_B = self.genA2B(real\_A) pred\_fake = self.disGB(fake\_B) loss\_GAN\_A2B = F.mse\_loss(pred\_fake,

torch.ones(pred\_fake.shape).type\_as(pred\_fake)) \* self.hparams[ 'adv\_weight']

fake\_A = self.genB2A(real\_B) pred\_fake = self.disGA(fake\_A) loss\_GAN\_B2A = F.mse\_loss(pred\_fake,

torch.ones(pred\_fake.shape).type\_as(pred\_fake)) \* self.hparams[ 'adv\_weight']

# Cycle loss

recovered\_A = self.genB2A(fake\_B)

loss\_cycle\_ABA = F.l1\_loss(recovered\_A, real\_A) \* self.hparams['cycle\_weight']

recovered\_B = self.genA2B(fake\_A)

loss\_cycle\_BAB = F.l1\_loss(recovered\_B, real\_B) \* self.hparams['cycle\_weight']

# Total loss

g\_loss = loss\_identity\_A + loss\_identity\_B + loss\_GAN\_A2B + loss\_GAN\_B2A + loss\_cycle\_ABA + loss\_cycle\_BAB

output = {



'loss': g\_loss,

'log': {'Loss/Generator': g\_loss}

}

self.generated\_B = fake\_B self.generated\_A = fake\_A

self.real\_B = real\_B self.real\_A = real\_A

# Log to tb

if batch\_idx % 500 == 0: self.logger.experiment.add\_image('Real/A',

make\_grid(self.real\_A, normalize=True, scale\_each=True),

self.current\_epoch) self.logger.experiment.add\_image('Real/B',

make\_grid(self.real\_B, normalize=True, scale\_each=True),

self.current\_epoch) self.logger.experiment.add\_image('Generated/A',

make\_grid(self.generate

d\_A, normalize=True, scale\_each=True),

self.current\_epoch) self.logger.experiment.add\_image('Generated/B',

make\_grid(self.generate

d\_B, normalize=True, scale\_each=True),

self.current\_epoch)

return output

if optimizer\_idx == 1: # Real loss

pred\_real = self.disGA(real\_A) loss\_D\_real = F.mse\_loss(pred\_real,

torch.ones(pred\_real.shape).type\_as(pred\_real))

# Fake loss

fake\_A = self.generated\_A

pred\_fake = self.disGA(fake\_A.detach()) loss\_D\_fake = F.mse\_loss(pred\_fake,

torch.zeros(pred\_fake.shape).type\_as(pred\_fake))

# Total loss

loss\_D\_A = (loss\_D\_real + loss\_D\_fake) \* 0.5

# Real loss

pred\_real = self.disGB(real\_B)

loss\_D\_real = F.mse\_loss(pred\_real, torch.ones(pred\_real.shape).type\_as(pred\_real))

# Fake loss

fake\_B = self.generated\_B

pred\_fake = self.disGB(fake\_B.detach()) loss\_D\_fake = F.mse\_loss(pred\_fake,

torch.zeros(pred\_fake.shape).type\_as(pred\_fake))

# Total loss

loss\_D\_B = (loss\_D\_real + loss\_D\_fake) \* 0.5 d\_loss = loss\_D\_A + loss\_D\_B

output = {

'loss': d\_loss,

'log': {'Loss/Discriminator': d\_loss}

}

return output

def configure\_optimizers(self): g\_optim =

torch.optim.Adam(itertools.chain(self.genA2B.parameters(), self.genB2A.parameters()),

0.999),

ay'])

d\_optim =

lr=self.hparams['lr'], betas=(0.5, weight\_decay=self.hparams['weight\_dec

torch.optim.Adam(itertools.chain(self.disGA.parameters(),

self.disGB.parameters

()),

ay'])

lr=self.hparams['lr'], betas=(0.5, 0.999),

weight\_decay=self.hparams['weight\_dec return [g\_optim, d\_optim], []

def train\_dataloader(self): train\_transform = transforms.Compose([



transforms.RandomHorizontalFlip(), transforms.Resize((self.hparams['img\_size'] + 30,

self.hparams['img\_size'] + 30)),

transforms.RandomCrop(self.hparams['img\_size']), transforms.ToTensor(),

transforms.Normalize(mean=(0.5, 0.5, 0.5), std=(0.5, 0.5,

0.5))



])

dataset = ImagetoImageDataset(self.hparams['domainA\_dir'], self.hparams['domainB\_dir'], train\_transform)

return DataLoader(dataset,

batch\_size=self.hparams['batch\_size'], num\_workers=self.hparams['num\_workers'], shuffle=True)

class Generator(nn.Module):

def init (self, ngf, n\_residual\_blocks=9): super(Generator, self). init ()

# Initial convolution block model = [nn.ReflectionPad2d(3),

nn.Conv2d(3, ngf, 7), nn.BatchNorm2d(ngf), nn.ReLU()]

# Downsampling in\_features = ngf

out\_features = in\_features \* 2 for \_ in range(2):

model += [nn.Conv2d(in\_features, out\_features, 3, stride=2,

padding=1),

nn.BatchNorm2d(out\_features), nn.ReLU()]

in\_features = out\_features out\_features = in\_features \* 2

# Residual blocks

for \_ in range(n\_residual\_blocks):

model += [ResidualBlock(in\_features)]

# Upsampling

out\_features = in\_features // 2 for \_ in range(2):

model += [nn.ConvTranspose2d(in\_features, out\_features, 3, stride=2, padding=1, output\_padding=1),

nn.BatchNorm2d(out\_features), nn.ReLU()]

in\_features = out\_features out\_features = in\_features // 2



# Output layer

model += [nn.ReflectionPad2d(3),

nn.Conv2d(ngf, 3, 7), nn.Tanh()]

self.model = nn.Sequential(\*model)

def forward(self, x): return self.model(x)

class Discriminator(nn.Module): def init (self, ndf):

super(Discriminator, self). init ()

# A bunch of convolutions one after another

model = [nn.Conv2d(3, ndf, 4, stride=2, padding=1), nn.LeakyReLU(0.2, inplace=True)]

model += [nn.Conv2d(ndf, ndf \* 2, 4, stride=2, padding=1), nn.BatchNorm2d(ndf \* 2),

nn.LeakyReLU(0.2, inplace=True)]

model += [nn.Conv2d(ndf \* 2, ndf \* 4, 4, stride=2, padding=1), nn.InstanceNorm2d(ndf \* 4),

nn.LeakyReLU(0.2, inplace=True)]

model += [nn.Conv2d(ndf \* 4, ndf \* 8, 4, padding=1), nn.InstanceNorm2d(ndf \* 8), nn.LeakyReLU(0.2, inplace=True)]

# FCN classification layer

model += [nn.Conv2d(ndf \* 8, 1, 4, padding=1)]

self.model = nn.Sequential(\*model)

def forward(self, x): x = self.model(x)

# Average pooling and flatten

return F.avg\_pool2d(x, x.size()[2:]).view(x.size()[0], -1)

We have two collections of photographs, and they are unpaired, meaning they are photos of different faces at different ages; we don't have the same facial structure in older ages and younger ages.



* **Collection 1**: Photos of younger ages.
* **Collection 2**: Photos of older ages.

We will develop an architecture of two GANs, each with a discriminator and a generator model, meaning there are four models in total in the architecture.

The first GAN will generate photos of winter given photos of summer, and the second GAN will generate photos of younger ages given photos of older ages.

* **GAN 1**: Translates photos of younger ages (collection 1) to older ages (collection 2).
* **GAN 2**: Translates photos of older ages (collection 2) to younger ages (collection 1).

Each GAN has a conditional generator model that will synthesize an image given an input image. And each GAN has a discriminator model to predict how likely the generated image is to have come from the target image collection. A GAN's discriminator and generator models are trained under normal adversarial loss like a standard GAN model.

We can summarize the generator and discriminator models from GAN 1 as follows:

## Generator Model 1:

* + **Input**: Takes photos of younger ages (collection 1).
  + **Output**: Generates photos of older ages (collection 2).

## Discriminator Model 1:

* + **Input**: Takes photos of older ages from collection two and Output from Generator Model 1.
  + **Output**: The likelihood of the image is from collection 2. Similarly, we can summarize the generator and discriminator models from GAN 2 as follows:

## Generator Model 2:

* + **Input**: Takes photos of older ages (collection 2).
  + **Output**: Generates photos of younger ages (collection 1).



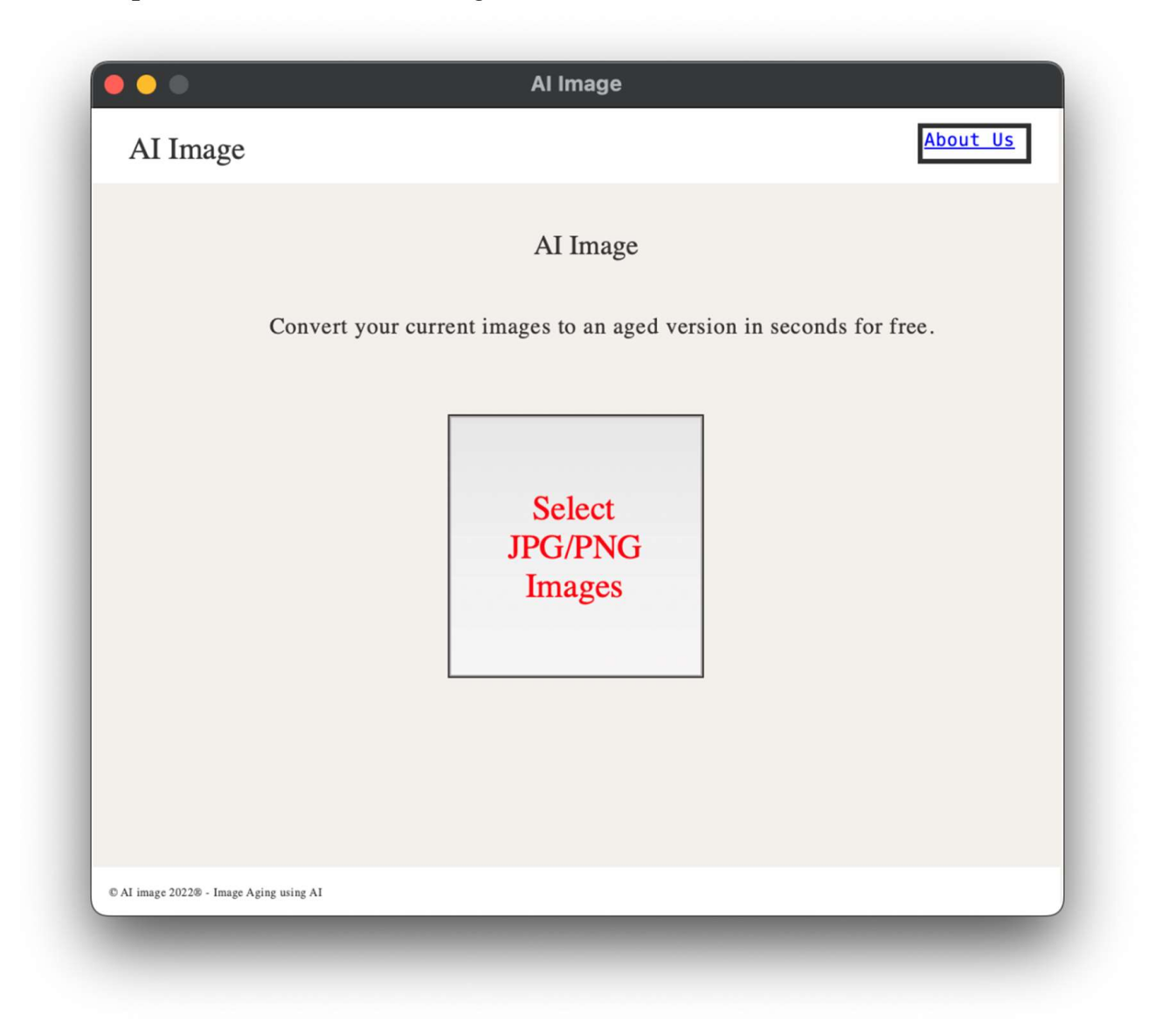
## Discriminator Model 2:

* + **Input**: Takes photos of younger ages from collection one and Output from Generator Model 2.
  + **Output**: The likelihood of the image is from collection 1.

## Screenshots

* **Main Dialog Box:**

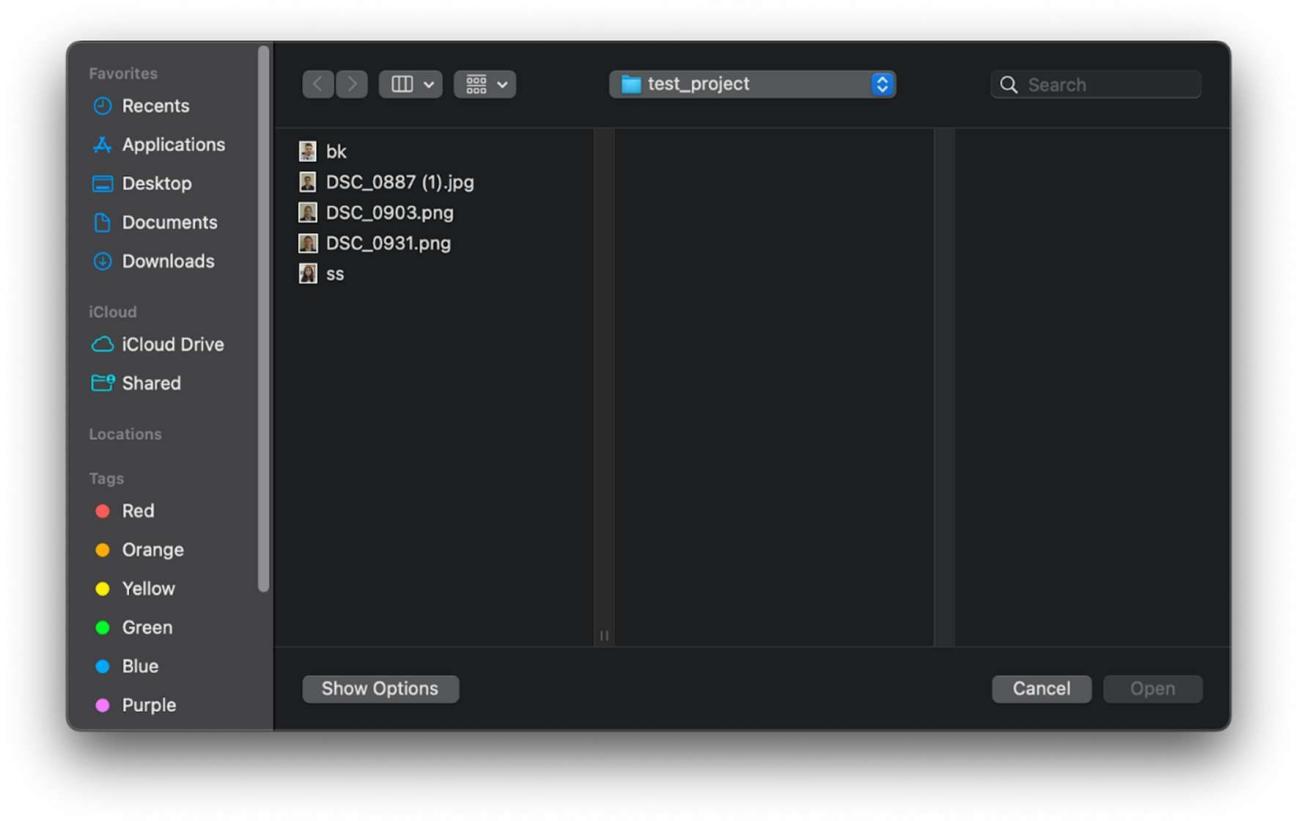
The main UI dialog box contains the Header, About Us, Project name, Mini- Description, Button to select image & Footer.



## Image Selection

**Fig no. 4.1 Main Dialog Box**

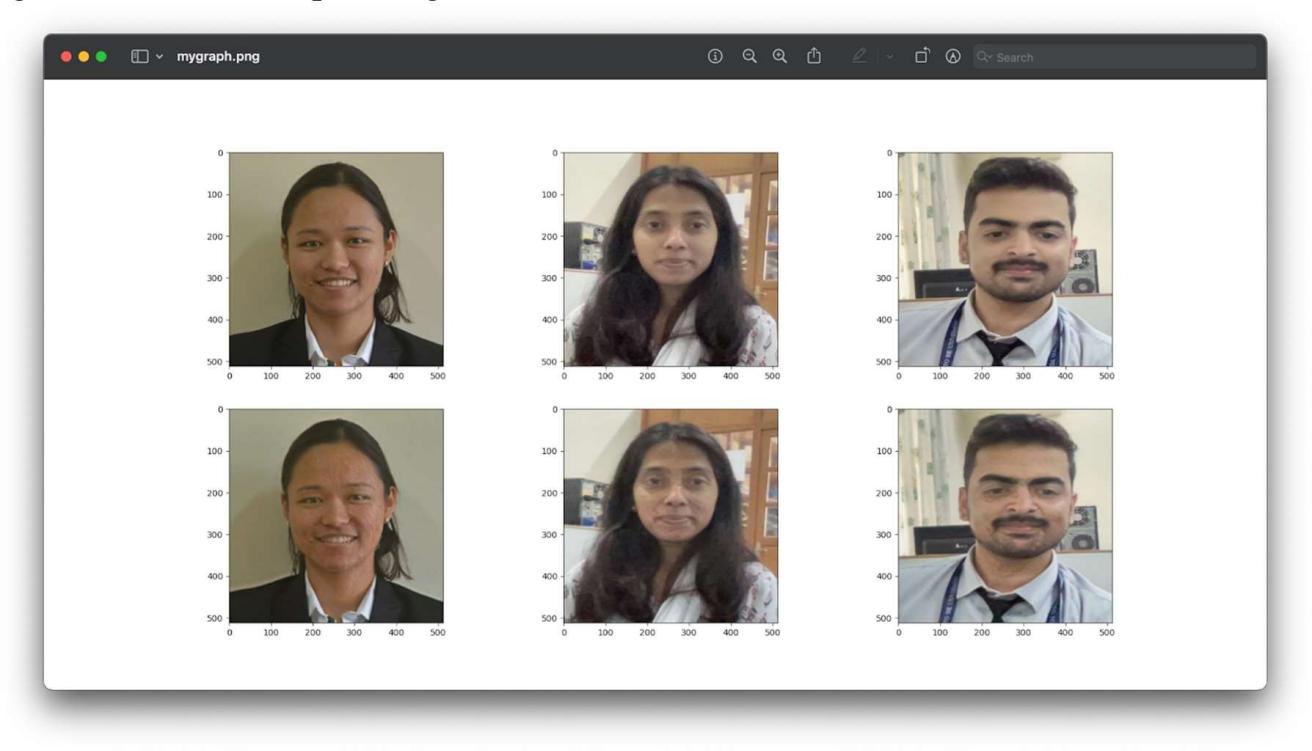
File Select Dialog Box appears upon clicking the Select JPG/PNG Button from the main page. This Dialog Box lets users choose an image from anywhere on the PC. The dialog box only lets images of either JPG or PNG format to get selected.



**Fig no. 4.2 Image Selection**

## Output

An image file of JPG format appears after processing, containing the input image and aged version of the input image.



**Fig no. 4.3 Output**

* **About Us:** (Links to Github Repo)



**Fig no. 4.4 About Us**

# TESTING

After the project's implementation or coding phase, the fifth chapter is the testing section. Writing test cases and doing extensive testing during this step ensures that the designed and implemented components are all functioning as intended. Test Cases and Testing Approaches make up the model.

## TEST CASES

* + 1. **Test case table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test ID** | **Test Name** | **Input Specification** | **Expected O/P** | **Actual Input** | **Actual output** |
| 1.1 | Input | 1. Select Image 2. Select Video 3. Select Document | 1. Accept 2. Error 3. Error | 1. flower.jpg 2. movie.mp4 3. aadhar.pdf | 1. Accepted 2. Error 3. Error |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2.1 | Requireme nt Validation | 1. Select JPG Image 2. Select PNG image 3. Select JPEG image | 1. Accept 2. Accept 3. Error | 1. flower.jpg 2. flower.png 3. flower.jpeg | 1. Accept 2. Accept 3. Error |

## TESTING APPROACH:

Unit Testing was chosen as the testing approach and was performed on the model. The model was divided into smaller units to affiliate with the testing standards. Test cases were designed by keeping the units in consideration.

**Unit testing:** A bit of code is checked to see if it functions as intended as part of this software testing. The name comes from the fact that tests are carried out one unit at a time. By identifying bugs or errors in the code level early in the development lifecycle, unit testing aims to lower the cost of bug fixes. Unit testing is a crucial component and a money saver for the company because the cost of fixing bugs increases exponentially as the Software Development Lifecycle moves into its later phases.

## Test Report



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test ID** | **Test Name** | **Input Specificatio n** | **Expecte d O/P** | **Actual Input** | **Actual output** | **Output** |
| 1.1 | Input | 1. Select | 1. | 1. | 1. | PAS |
|  |  | Image | Accept | flower.jpg | Accepte | S |
|  |  |  |  |  | d |  |
|  |  | 2. Select |  |  |  |  |
|  |  | Video | 2. | 2. |  |  |
|  |  |  | Error | movie.mp4 | 2. Error |  |
|  |  | 3. Select |  |  |  |  |
|  |  | Document |  |  |  |  |
|  |  |  | 3.Error | 3. | 3. Error |  |
|  |  |  |  | aadhar.pdf |  |  |
| 2.1 | Requiremen t Validation | 1. Select JPG Image | 1.  Accept | 1.  flower.jpg | 1.  Accept | PAS S |
|  |  | 2. Select PNG image | 2.  Accept | 2.  flower.png | 2.  Accept |  |
|  |  | 3. Select JPEG image | 3.  Error | 3.  flower.jpe g | 3. Error |  |

**Table no. 2.2 Test Report**



# CONCLUSION

## DESIGN AND IMPLEMENTATION ISSUES

The training phase of the model is the most demanding aspect of the entire project. This can easily hamper the design and implementation and create issues such as delays and time mismanagement.

Since the training time is high, the coding part has to be faultless, as changes in the code will lead to further training and more time delays.

Attributes such as scaling and performance will directly depend on the type of system used to train the model. Without a high-end configuration system, the project's scalability aspect will severely hamper other related attributes.

For implementation purposes, the user side should have a minimum system requirement fulfilled system to successfully and easily run the pre-trained model.

## ADVANTAGES AND LIMITATIONS Advantages:

* + - Since the existing system results are gimmicky and unrealistic, the project stays grounded with its approach and provides a realistic output type. It does not over-saturate the product in terms of getting an output.
    - The system also accepts multiple inputs from the user side and works with them. It performs transformation successfully over the number of inputs and produces proper Output.
    - The model successfully fulfils its objective of successfully creating a proper image-to-image translation while following all standards and parameters.

## Limitations:

* + - Does not work well in situations where the entity in the input image is minor. In such cases, the results are not accurate.
    - Sometimes accepts inputs with no proper facial attribute, thus affecting the Output
    - Does not work accurately to provide results on dysfunctional faces or faces with any anomaly.



## Future Scope of the Project

* + 1. Medical Sciences can use this model extensively with the dermatology department in collaboration with new technology to improve their service towards their patients.
    2. Film making, apart from technical domain the entertainment industry can also use this model by tweaking the algorithm to suit their needs, by age enhancing the film characters. Similar kind of options are available by Graphics designing & VFX, but this could be a cheaper option in- comparison to others.
    3. Record maintenance, any domain dealing with record keeping for a longer duration can implement it for better customer service and identification.

Image Aging using Cycle GAN

34

# REFERENCES



1. Face Aging With GAN, Siyao Sun, Alex Wang {siyaosun, cwang16}@stanford.edu https://github.com/siyaofd/GANSTER
2. GANs for Face Aging problems — What does your face look like in the next few years? Neurons AI Feb 9, 2021 https://neurondai.medium.com/
3. A Gentle Introduction to CycleGAN for Image Translation Jason Brownlee, August 5, 2019, https://machinelearningmastery.com/what-is-cyclegan/
4. A. Lanitis, C. J. Taylor, and C. J. Taylor, T. F. Cootes. 2002. Toward Automatic Simulation of Aging Effects on Face Images. IEEE Trans. Pattern Anal. Mach. Intell. 24, 4 (April 2002), 442-455. Jinli Suo, Song-Chun Zhu, Shiguang Shan, and Xilin Chen. 2010. A Compositional and Dynamic Model for Face Aging. IEEE Trans. Pattern Anal. Mach. Intell. 32, 3 (March 2010), 385-401.
5. Burt, D. M. and David I. Perrett. “Perception of age in adult Caucasian male faces: computer graphic manipulation of shape and colour information.” Proceedings. Biological sciences 259 1355 (1995): 137-43 .
6. Bernard Tiddeman, Michael Burt, and David Perrett. 2001. Prototyping and Transforming Facial Textures for Perception Research. IEEE Comput. Graph. Appl. 21, 5 (September 2001), 42-50.