

College of Computing & Informatics (CCIT)

SENIOR PROJECT-I REPORT

Deepfake

Author(s):

S180165179

Mohammed Shaya Alsahli

S180314937

Abdulaziz Ibraheem AlsIman

S170094948

Rayed Muqbel Al-Enezi

Project Supervisor:

Dr Mohamed Habib

Deepfake

By:

Mohammed Shaya Alsahli

Abdulaziz Ibraheem Alslman,

Rayed Muqbel Al-Enezi

Thesis/Project submitted to:

College of Computing & Informatics, Saudi Electronic University, Riyadh, Saudi Arabia.

In partial fulfillment of the requirements for the degree of:

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

Project Supervisor

Project Committee Chair

ABSTRACT

Deepfake algorithms, as they are generally known, were developed in recent years these algorithms allow the user to accurately plant one person's face onto another.

Giving anyone the ability to make photographs or videos of anyone doing something. The same can be done for the voice and speech in a related branch, enabling the ability to make anyone say anything.

Therefore, my colleagues and I will make an application for this new technology and we will work on it from all different techniques, such as: Deepfakes Video, Audio and image.

The work on this technology will help professionally understand the techniques of machine learning, deep learning and artificial intelligence and these technologies will be applied to specialized tools and will be programmed in python language and requires advanced experience.

DEDICATION

We have completed a full study of the report and under the direct and weekly supervision of Dr. Mohammed Habib, so we thank him for his support and providing guidance and advice to us and to make it seem easy by giving us hope, and we also thank our university for empowering us and teaching us these modern technologies and their keenness to have our outputs in high quality.

PREFACE

This report contains a comprehensive definition of deep learning technology and machine learning and its concept in general and will take the subject of deepfake in particular in its full characteristics, techniques and ideals from reality and will address its working methods and requirements and problems that can occur and ways to avoid them.

In the end, this report was prepared to clarify all functional and non-functional requirements, and the analysis and design models that should be taken at the stage of implementing the deepfake application and operating its main functions.

This project will be presented as a final it projects in a bachelor's degree in the Faculty of Computing and Informatics at the Saudi Electronic University.

REVISION HISTORY

Name	Date	Reason For Changes	Version

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION	8
1.1 Project Background/Overview:	8
1.2 Problem Description:	9
1.3 Project Scope:	10
1.4 Project Objectives:	10
1.5 Project Structure/Plan:	10
Project Plan:	10
GANTT Chart:	11
WBS Diagram:	11
Activates Table:	11
ADM Diagram:	12
PDM Diagram:	12
CHAPTER 2: LITERATURE REVIEW	13
2.1 Machine Learning	13
2.1.1 Machine Learning Techniques	15
2.2 Deep Learning	16
2.2.1 Deep Learning definition	17
2.2.2 Deep Learning and Machine learning differences	17
2.2.3 Deep Learning Applications	18
2.2.4 Deep learning techniques	18
2.3 Deepfake	21
2.3.1 Generative Adversarial Networks (GANs)	23
2.3.2 Autoencoders	24
2.3.3 Face reenactment (face synthesis)	25
2.3.4 Face swapping	26
2.3.5 DEEPFAKE DETECTION	28
2.3.6 deepfake applications	29
CHAPTER 3: METHODOLOGY	32
CHAPTER 4: SYSTEM ANALYSIS	34
4.1 Product Features:	34
4.1.3 Face Generation	36
4.2 Functional Requirements:	37
Use-Case 1	37
Use-Case 2	38
Use-Case 3	39

Use-Case 4.....	40
Use-Case 5.....	41
4.3 Nonfunctional Requirements.....	42
Performance Requirements	42
Safety Requirements.....	42
Security Requirements	42
Software Quality Attributes.....	42
4.4 Analysis Models	43
Use-Case Diagram.....	43
CHAPTER 5: SYSTEM DESIGN.....	44
component diagram	44
deployment diagram	45
CHAPTER 6: SYSTEM IMPLEMENTATION.....	46
6.1 Application Implementation	46
6.2 Deepfake applications.....	46
6.2.1 Face swap for pictures	46
6.2.2 Face swap in real time with webcam pc	49
6.2.3 Face Generation Using GAN's	51
CHAPTER 7: TESTING & EVALUATION.....	55
CHAPTER 8: RESULTS AND ANALYSIS	57
8.1 Face swap for pictures	57
8.1.1 create a new canvas for final image in exactly the same size of destination image.....	57
8.1.2 create the destination face mask	58
8.1.3 invert the face mask color	59
8.1.4 mask destination face	60
8.1.5 place new face into destination image.....	61
8.1.6 Do seamless clone to make the attachment blend with the surrounding pixels	62
8.2 Face swap in real time with the pc cam.....	65
8.2.1 Landmark points of source and destination	65
8.2.2 Detector faces in frame check	65
8.2.3 The completed destination canvas	66
8.2.4 the final destination face mask.....	67
8.2.5 the inverted final destination face mask.....	67
8.2.6 The destination face on webcam masked	68
8.2.7 Result before seamless cloning	69
8.2.8 Final result after doing seamless cloning	70
8.3 Face Generation Using GAN's.....	72
8.3.1 Training epochs and time	72
8.3.2 Plot generated images and final result	73
CHAPTER 9: CONCLUSION AND FUTURE WORK	77

7.2	Conclusion	Error! Bookmark not defined.
7.3	Future Work	Error! Bookmark not defined.
REFERENCES		81

CHAPTER 1: INTRODUCTION

1.1 Project Background/Overview:

Deepfake is content or material that has been synthesized or altered using artificial intelligence (AI) methods in order to be passed off as real. It can comprise audio, video, image, and text synthesis.

This report gives readers an overview of:

The basic requirements for understanding deepfake which are (machine learning and deep learning techniques), The many deepfake categories, how could they be made and detected, the most recent trends in this field, audio deepfakes, most popular deepfakes applications.

Deepfakes Video:

videos that are manipulated have had a great spread and interest in this new and controversial technology, especially after the deepfake technique that can manipulate images and videos also

using deeplearning tools, The deepfake algorithm can switch whole facets of targeted video with a face from another video.

We'd like to create a software tool that can combine different Deepfake approaches to create a video that is both visual and audio integrated and of high quality.

Our solution will function with images and audio recordings, and will be easily accessible through ordinary gadgets in practically every computer utilized, thanks to recent technological advancements.

1.2 Problem Description:

Briefly describe the problem and the need to solve it Deep counterfeiting algorithms, as they are often known, have been developed in recent years.

These algorithms allow the user to implant one person's face over another in a convincing manner, allowing anyone who does anything efficiently to generate photos or films.

The same can be done with sound and voice in a related branch, giving the capacity to make anyone say anything.

These algorithms, however, could be employed in the development of application to allow the user to insert himself in the video.

Our project may make use of high-quality facial image and vocal representation to build a highly customizable application at a reasonable cost.

1.3 Project Scope:

Describe the aims, benefits and the outcome of the solution developed

The deepfake application will do the following:

- **Face Swap** - is a popular technology used by many applications in which someone's faces are replaced by someone else's face with a video or photo.
- **Lip Syncing** - is the outcome of combining the latest research on generating faces using ML. and the astonishing progress in speech synthesis, which is less popular than generating faces and synthesizing speech.
- **Face Generation** - The task of producing (or interpolating) new faces from an existing dataset

1.4 Project Objectives:

The main purpose of the Deepfake project is

- Speech and voice synthesis
- Music & sound synthesis
- Image synthesis
- Video synthesis
- Mixed reality synthesis
- Natural-language generation
- mask the identity of people's voices and faces to protect their privacy
- Generation Digital avatars instead of living people can be used in customer service, commercial advertisements, or educational lessons.

1.5 Project Structure/Plan:

Project Plan:

Activity name	Start	Finish	Duration
1- Planning	29/1/2022	19/3/2022	50d
2- Analysis	20/3/2022	13/4/2022	25d
3- Design	14/4/2022	6/5/2022	23d
4- Implantation	6/8/2022	24/9/2022	50d

5- Testing	25/9/2022	14/10/2022	20d
------------	-----------	------------	-----

Table 1: Project Plan

GANTT Chart:

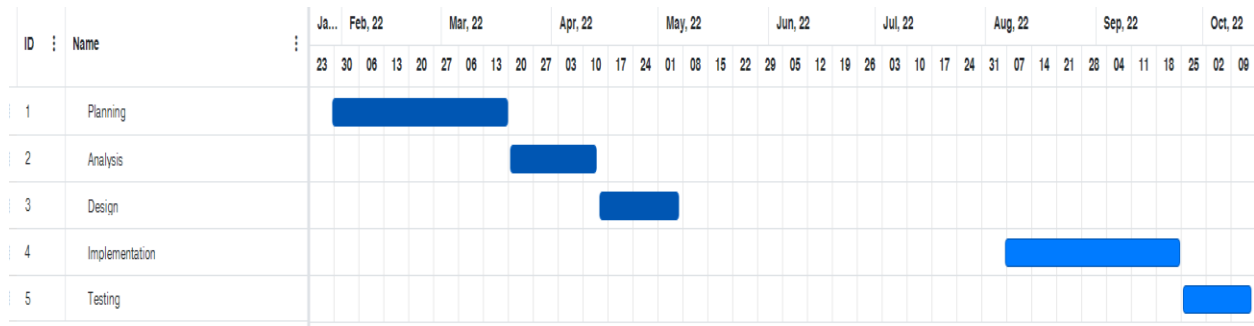


Figure 1: GANTT Chart

WBS Diagram:

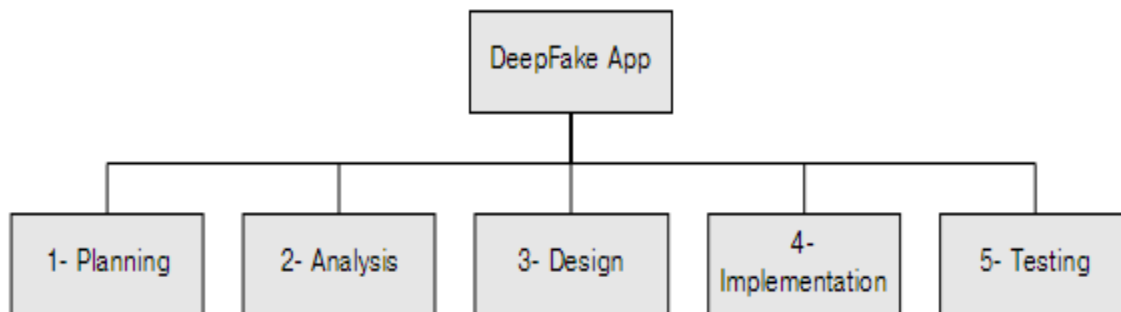


Figure 2: WBS Diagram

Activates Table:

Activity name	Predecessor	Duration
1- A (Planning)	None	50d
2- B (Analysis)	A	25d
3- C (Design)	B	23d
4- D (Implantation)	C	50d
5- E (Testing)	D	20d

Table 2: Activities

ADM Diagram:

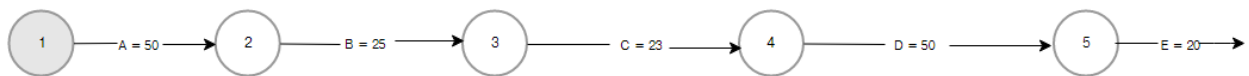


Figure 3: ADM Diagram

PDM Diagram:

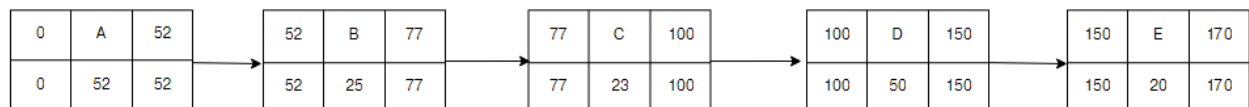


Figure 4: PDM Diagram

CHAPTER 2: LITERATURE REVIEW

2.1 Machine Learning

Machine learning is a data science profession that uses statistical approaches to enhance performance based on prior experience or find new patterns in large amounts of data. (Theobald, n.d.)

The use of self-improving algorithms is a crucial part of machine learning. Self-improving algorithms, like humans, learn from previous experience and trial and error to make decisions. Machine learning is not only capable of thinking and learning like humans, but it is also more efficient. Humans are simply not predisposed to be as dependable and skilled at repetitive activities as computers are at data handling. Furthermore, the scale, complexity, and speed with which big data may be generated far outstrips our human skills. (Theobald, 2017) (Kubat, n.d.)

It's quite easy for us to spot the pattern here as humans. We can safely estimate that the unknown number in brackets on row five will be '24' because the second number in each row is twice as large as the number to its left in the brackets. We don't need a machine to forecast the unknown number in this circumstance. What if each row was made up of substantially larger integers with decimal points in the double digits and a less obvious relationship between each value? It would be incredibly difficult, if not impossible, for anyone to analyze and forecast in real time as a result of this. This task however is not daunting to a machine. Machines can perform the tedious tasks of

gathering, storing, and displaying data, as well as attempting multiple options to isolate vast chunks of data in order to address the problem at hand.

As a result, machine learning frees up our time to focus on increasing results or other business concerns.(Kubat, n.d.)

But how can we train a computer to calculate something that we don't even know how to do?This is a crucial feature of machine learning. Machine learning algorithms, when properly set, are capable of learning and recognizing new patterns in a matter of minutes However, machine learning does not begin on its own. There must be a human to program and supervise the automated process, just as there must be a human to program and supervise any machine or automated production line. This is where data scientists and data specialists come into play.(Kubat, n.d.) (Mohan et al., 2019)

Data scientists are responsible for configuring equipment

(such as servers, operating systems, and databases)

and architecture (how the equipment interacts), as well as developing algorithms employing various mathematical operations.Programming a computer is like training a guide dog. The dog is taught how to respond in various situations through specific training. The dog may be taught to heel at a red light or to securely take its master past obstacles,for example.If the dog has been adequately taught, the trainer is no longer necessary, and the dog can use his or her training in a variety of unattended scenarios.(Theobald, 2017)

This example is based on a hypothetical circumstance, but what if you wanted to program a computer to perform more difficult tasks like image recognition? How do you educate a computer to distinguish between different animals based on their physical characteristics? This, too, necessitates a significant amount of human involvement.(Kubat, n.d.)

The data scientist will need to approach this strategy differently than programming the computer to respond to a set option, such as navigating a path barrier or responding to a redlight.Because of the high failure rate, the data scientist cannot design the computer to recognize animals based on a human description (i.e. four legs, long tail, and long neck). This is due to the fact that many species, such as wallabies and kangaroos, have comparable features. Computers and traditional computer science programming have long been limited in their ability to solve such difficult jobs.

Instead, the data scientist must teach the machine how to identify animals by socializing examples, similar to how a toddler is taught.

A young youngster cannot correctly identify a 'goat' based on a description of its major characteristics. Of course, an animal with four legs, white fur, and a small neck may be mistaken for a variety of other creatures.Rather than playing a guessing game with a youngster, it's more helpful to show the child toy goats, photographs of goats, or even real-life goats in a paddock to demonstrate what a goat looks like.Machine learning is similar to image recognition, except that instruction is done using images and a programming language.For example, we can show the computer a variety of photos labeled with the topic matter, such as 'goat.' The system then uses these samples to recognize the subject's individual features, just like a child might.

I even read about a Chinese company that used machine learning algorithms to detect illegal video content and pornography at work. Now, for the solution to your question... Yes, in order to acquire such advanced video recognition capabilities, the computers would have been fed a large amount of pornographic content! Whether it's recognizing animals, human faces, or illegal adult material, the machine may use examples to create its own software to recognize and identify subjects. This reduces the need for humans to explain each subject's features in great detail, lowering the risk of

failure considerably. Machine learning can begin after the architecture and algorithms have been successfully configured. The computer can then start applying algorithms and models to data in order to classify, predict, and cluster it in order to gain new insights.

2.1.1 Machine Learning Techniques

Machine learning algorithms can be split into different classes of algorithms, including supervised and unsupervised.

2.1.1.1 Supervised

Learning guided by human observations and feedback with known outcomes is referred to as supervised algorithms.

Assume you want the machine to sort your email into spam and non-spam categories. (Mishra et al., 2019)

You already have information that you can feed the machine in a supervised learning setting to describe what type of email belongs to which category. As a result, the machine understands that there are two labels in which to sort the incoming data (emails). (Kubat, n.d.)

You could also develop a model to evaluate games over the last three years to predict who will win a basketball game. In order to determine who would win the following game, the total number of points scored and total number of points against could be evaluated. (Mohan et al., 2019) (Kubat, n.d.)

This information might subsequently be fed into a categorization model. We may use regression to forecast who will win based on the average of previous performances once the data has been categorised and plotted on a data plot. After that, the ultimate result would provide an answer based on total points. We've told the machine which categories to look at, just like in the first example (points for, and points against). As a result, the data has already been pre-tagged. Linear Regression, Logistic Regression, Neural Networks, and Support Vector Machine methods are examples of supervised algorithms with tags applied

2.1.1.2 Unsupervised

There is no such integrated feedback or use of tags in an unsupervised learning environment. Instead, the machine learning program must rely solely on clustering distinct data and modifying its algorithm in response to its initial discoveries - all without the benefit of human feedback. (Theobald, 2017) (Kubat, n.d.)

Unsupervised learning is exemplified by clustering techniques. Clustering brings together data items that have been identified to have similar characteristics. If you cluster data points based on the weight and height of 13-year-old high school kids, for example, you're likely to find two clusters form. One of the huge clusters will be male, while the other will be female.

This is due to the fact that when it comes to physical measurements, girls and boys have different commonalities. Unsupervised algorithms have the advantage of allowing you to identify patterns in your data that you may not have been aware of, such as the presence of two different sexes. (Mohan et al., 2019) (Theobald, 2017)

Following the discovery of certain groupings, clustering can be used as a springboard for additional investigation. Clustering algorithms and falling dimension algorithms are examples of unsupervised algorithms that do not require tags.

2.2 Deep Learning

Nowadays everyone knows Artificial Intelligence (AI), you might not know exactly how it works but you have some idea and can identify some of the tools around you that are powered by it, and one of the most important subsets of AI is Machine learning (ML) Which we just covered, And Deep learning (DL) which is a subset of ML, It's more advanced more accurate and requires less humane interference compared to ML.

Deep learning (DL) is a quickly evolving field, which has demonstrated amazing results in performing tasks that traditionally have been performed well only by humans. Examples of such tasks are image classification, generating natural language descriptions of images, natural language translation, speech-to-text, and text-to-speech conversion. And it relies on the notion of extracting features from raw data utilizing numerous layers to recognize various elements of input data.

Deep learning is a branch of machine learning that produces progressively complicated hierarchical models that aim to better resemble human cognitive processes than simple machine learning models.

As a result, deep learning is miles better but also more complicated than machine learning while remaining a subset of AI and a progression of machine learning. Deep learning, like so many other technologies that few people are aware of, may already have an impact on your life.

People are ecstatic to learn that Netflix predicts their viewing preferences. Or when Amazon shows them more things they might be interested in.

Deep learning algorithms power these recommendation engines, which can act on massive amounts of data to help enhance the quality of your life.

2.2.1 Deep Learning definition

“We do not know of a crisp definition of what DL is, but one attempt is that DL is a class of machine learning algorithms that use multiple layers of computational units where each layer learns its own representation of the input data. These representations are combined by later layers in a hierarchical fashion.” (Ekman, 2021)

“It comprises multiple hidden layers of artificial neural networks. The deep learning methodology applies nonlinear transformations and model abstractions of high level in large databases.”(Vargas et al., 2017)

“Deep learning is a computer-based modeling approach, which is made up of many processing layers that are used to understand the representation of data with several levels of abstraction.”(Mishra et al., 2021)

It's vital to remember that these systems are only labeled "deep" because of how they seem when we draw them stacked vertically. They aren't profound in the sense of having in-depth knowledge or insights. When a deep learning system assigns a name to a face in a photo, it doesn't know what faces are, or what humans are, or even the existence of people. The computer just analyses pixels and generates the most likely label based on the patterns it learned from the training data. (Glassner, 2021)

2.2.2 Deep Learning and Machine learning differences

The inability of machine learning to analyze raw input data limits its application.

Deep learning has helped to overcome this constraint because it can operate on vast amounts of data, making it an effective and valuable machine learning technique.

Deep learning has also accelerated as a result of computer hardware developments.

To turn the raw data into a format that the machine's subsystem could recognize and classify, extensive knowledge with feature extraction was required.

Data representation has a big impact on machine learning performance. As a result, much of the effort is spent on preprocessing design, making the algorithms time-consuming.

By learning data representation, representation learning is utilized to extract only useful information during data classification. Feature learning has taken the place of manual data representation, allowing the machine to automatically identify representations that are useful for classifying. Representation learning has been applied in both academic and industrial sectors.

In addition to this Deep Learning algorithms requires substantially more data than Machine Learning to Perform well.

So in summary, while in machine learning you need to identify features first then feed them to the AI which might lead to several issues, for example, lead the AI to be affected by the viewpoints of the people in charge and looking at the world from the perspective of a few people, But in Deep learning, You only feed

the AI the data and it has to figure out identify the distinct features by itself. And if the data volume is small then Machine Learning would be a better choice.

2.2.3 Deep Learning Applications

MAVIS-Microsoft speech recognition is done with the help of deep learning. In this learning human voices and speeches help the search of audios and video files.

For image searching, Google used the deep learning method in a Big Data environment, which assists in the creation of an understanding of images so that image tagging, indexing, and annotation can be simplified. (Sharma et al., 2021)

A Deep Dream is a software created initially by Google to help scientists understand what a DNN is seeing then later it turned into an art tool that can classify images and can make abstract art.

Self-Driving Cars The automotive industry is racing to get ahead in the autonomy field, which many believe is the future of cars and one of the most advanced companies in the field is tesla it jumped in the last couple of years to become the most valued car company and right now it's the 6th highest market cap company in the world valued roughly at 870B dollars.

Medical Imaging For image segmentation, disease identification, and prediction, Magnetic Resonance (MR) and Computed Tomography (CT) techniques use image recognition and object detection. Deep learning algorithms can effectively analyze imaging data by combining aspects such as tissue size, volume, and form. Important sections in photos can be highlighted using these models. Deep learning algorithms are utilized to diagnose diabetic retinopathy, Alzheimer's disease early detection, and many more. (Takimoglu, n.d.)

2.2.4 Deep learning techniques

There are several techniques and more techniques are implemented regularly and that is because using Deep Learning wasn't feasible until recent years which means Deep Learning is still at its early stages which makes it a fast-changing field.

In my research, a few techniques appealed to me more than others Mainly because of their connection to our project(Deep Fake)

- **Generative Adversarial Network (GAN)**

When working on a deep fake app we most likely are going to need to generate plausible pictures, and I find the way it works interesting especially since the Generator is always trying to fool the Discriminator.

- **Convolutional Neural Network (CNN)**

It is highly recommended that you use CNN for its pattern recognition, and since that is one of the key needs for deep fake I felt it is important to explore it a bit.

- **Recurrent Neural Network (RNN)**

I included RNN mainly because talking about DRAW won't be complete without talking about RNN which DRAW partially utilize

- **Deep Recurrent Attentive Writer (DRAW)**

The idea of a computer drawing an output a bit by bit sounded interesting to me, Just thinking of all the possible uses of this technique made me want to explore and learn it more, some of the uses that come to mind are drawing autogenerated faces, But most importantly we can use it to generate videos from scratch.

2.2.4.1 Generative Adversarial Network (GAN)

A Generative Adversarial Network (GAN) is a type of neural network architecture for generative modeling to produce new believable samples on demand. It involves detecting and learning regularities or patterns in input data for the model to generate or output new samples from the original dataset.

“GANs are composed of two neural networks, a generator G that creates new data having properties similar to the original data, and a discriminator D that predicts the likelihood of a subsequent sample being drawn from actual data rather than data provided by the generator. Thus in GAN modeling, both the generator and discriminator are trained to compete with each other. While the generator tries to fool and confuse the discriminator by creating more realistic data, the discriminator tries to distinguish the genuine data from the fake data generated by G.” (Sarker, 2021)

GAN network deployment is typically used for unsupervised learning tasks, but depending on the problem, it has also proven to be a better solution for semi-supervised and reinforcement learning. GANs are also employed in cutting-edge transfer learning research to ensure that the latent feature space is aligned. A conventional GAN model learns a mapping from a latent space to data distribution. GAN networks' potential application fields include healthcare, image analysis, video generation, voice generation, cybersecurity, and many others, all of which are quickly expanding. GANs have proven themselves as a comprehensive domain of independent data expansion and a solution to challenges that require a generative solution in general.

2.2.4.2 Convolutional Neural Network (CNN)

A convolutional network is an artificial neural network (ANN) that is primarily used for image analysis due to its ability to recognize patterns, but it may also be used for data analysis and classification. The convolutional network is based on the brain's biological processing.

The term "convolutional" refers to the network's data filtering. Several filters are utilized to recognize the pattern in each convolutional layer. The pattern here can be the edges, corners, objects, and shapes such as triangles and circles.

These filters are applied to a certain patch in the image. The data processed by the filter is passed to a set of pooling layers, where the patch recognized by the convolutional layer is halved, and then passed to the next set of convolutional and pooling layers, where the patches from the previous layers are acted upon by the filters, and so on. Then the data is sent to the fully connected layer at the end of the network, and the fully connected layer produces the images with the highest probabilities from the many sets of output from the pooling layer. The image with the highest probability is recognized as a patch or pattern.

“Convolutional networks have evolved a lot starting from the 1990s using networks for speech recognition, image recognition to document recognition of bank checks.

Convolutional networks have been preferred over the fully connected networks for document verification tasks like handwriting recognition by expanding the training set used.

Convolutional networks have also been used for face recognition and detecting multiple faces in a single image and capturing the faces in different poses and variance in scale or translation.” (Mishra et al., 2021)

2.2.4.3 Recurrent Neural Network (RNN)

Humans can understand the words in any paragraph depending on the previous words they read, and they can understand that paragraph based on the sequencing operations that run in their brains. RNNs (recurrent neural networks) act in a similar way. RNNs are neural networks that are often used for pattern recognition tasks and are used to handle sequence data.

The sequence data is broken into chunks and sent to the RNNs in the form of input.

They use sequential memory which makes it easier for the networks to recognize the patterns that are used in speech and language translations and image recognition.

The RNN architecture consists of text input with many features and hidden layers, as well as a time-dependent output.

The RNNs will analyze one word at a time, attempting to keep the words in order by looping the previous words with the new words provided as input to the hidden layers.

RNNs are also utilized in more complicated applications, such as translating languages with long sentences, and image search, in which the word we're looking for is turned into a picture and the appropriate image is produced as the output. Another application of RNN is image captioning. As an input, the system receives an image, which the RNN turns into a sentence. It is capable of identifying the things in the image as well as expressing their relationships. (Mishra et al., 2021)

2.2.4.4 Deep Recurrent Attentive Writer (DRAW)

When asked to sketch, paint, or otherwise recreate a visual scene, most people will do so in a sequential, iterative manner, reassessing their work after each change. Lines are sharpened, darkened, or erased, shapes are changed, and the final picture emerges. On the other hand, the majority of automatic image generation approaches, try to create full scenes at once.

This often means that all pixels in generative neural networks are conditioned on a single latent distribution.

The "one-shot" approach is essentially difficult to scale to larger images, and it also eliminates the option of repeated self-correction. The Deep Recurrent Attentive Writer (DRAW) architecture reflects a move toward a more natural type of image construction, in which components of a scene are formed independently of one another and rough sketches are modified over time.

A pair of recurrent neural networks form the foundation of the DRAW architecture: an encoder network compresses the real images supplied during training, and a decoder network reconstructs images after receiving codes. The combined system is trained using stochastic gradient descent from start to finish, with the loss function being a variational upper bound on the data's log-likelihood. As a result, it is a member of the family of variational auto-encoders, a recently developed mix of deep learning and variational inference that has led to substantial breakthroughs in generative modeling.

2.3 Deepfake

Recently, videos that are manipulated have had a great spread and interest in this new and controversial technology, especially after the deepfake technique that can manipulate images and videos also using deeplearning tools, The deepfake algorithm can switch whole facets of targeted video with a face from another video.

While the deepfake technology can be used for positive purposes, such as film industry and video production on virtual reality, but they can be used harmfully and dangerously, A large number of fake videos were posted online and often targeted at political figures and celebrities, for example in 2018 produced a video of Donald Trump giving a speech in which he called for Belgium to leave the Paris climate agreement, Trump on the other hand never gave the speech, it was a ruse.(Burchard, H. von der,2018)

That wasn't the first time a deepfake was used to generate deceptive films, and tech-savvy political gurus are anticipating a new wave of fake news including convincingly realistic deepfakes in the future.

Another example of a video in which deepfake technology has been used in which it may look exactly like actor Morgan Freeman, this video has been very popular on social media platforms and reflects the rapid development of deepfake technology created with the help of Generative Adversarial Networks (GAN). (Diep Nep, 2017)

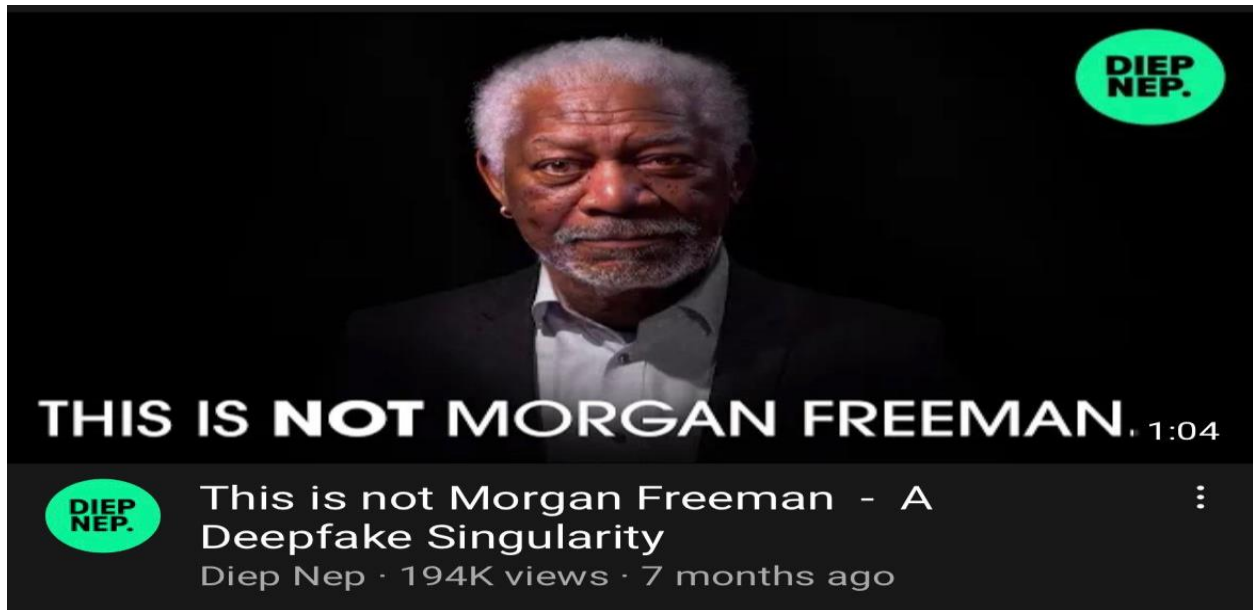


Fig. 1.example of high quality Deepfake video

"Deepfakes is the phenomenon of creation of realistic digital products"(Karnouskos, 2020) ,the name "deepfake" is derived from the underlying technology "deep learning," which is a type of artificial intelligence, deep learning algorithms are used to swap faces in the video and digital information to create realistic-looking fake media, deep learning algorithms train themselves how to solve issues when given vast volumes of data, deepfakes can be created in a variety of ways, but the most frequent is to use deep neural networks with autoencoders that use a face-swapping methodology.

You will need a target video to utilize as the deepfake's foundation, as well as a collection of video clips of the individual you wish to place in the target, deepfakes aren't only videos, deepfake audio is a rapidly expanding field with a wide range of applications realistic audio deepfakes can now be created using deep learning algorithms in just a few hours (or in some cases minutes) of sound from the person whose voice is being cloned, and after creating a model of that person's voice, it can be created to have them say something.

A deep learning system can create convincing fakes by examining photographs and videos of a target person from different angles and then mimicking their behavior and language patterns, the first attempt at face alteration can be seen in the renowned 1865 portrait of US President Abraham Lincoln.

Facetanipulation in digital photographs has grown much easier with the advancement of computer graphics technology, recent advances in the field of deep learning have made a significant contribution to the advancement of deepfake technology.

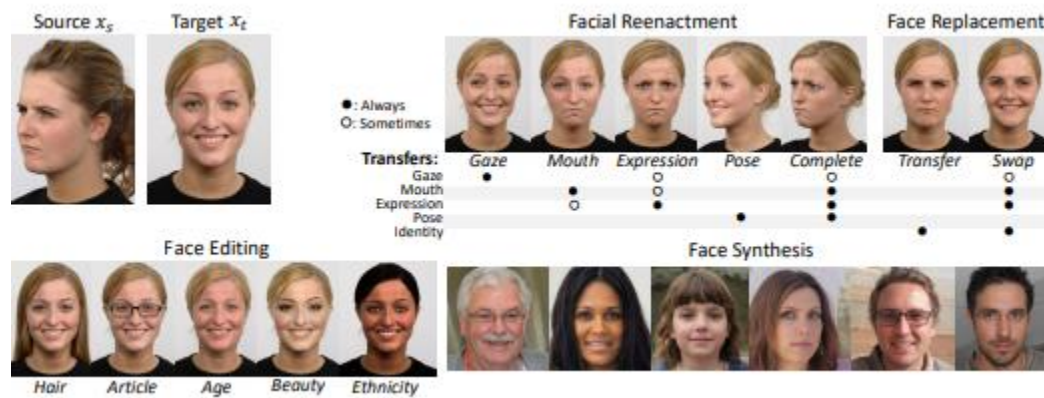


Fig. 2. Examples of reenactment, replacement, editing, and synthesis deepfakes of the human face

Not all deepfakes are malicious. However, because the technology makes it so easy to create believable media, malicious users are exploiting it to perform attacks. These attacks are targeting individuals and causing psychological, political, monetary, and physical harm (Mirsky & Lee, 2022).

2.3.1 Generative Adversarial Networks (GANs)

Another type of machine learning, known as Generative Adversarial Networks (GANs), is a big part of deep learning which is used to generate data completely from scratch, such as images, music or a combination of both. It can create a lot of things from data that actually does not exist; that's created from scratch.

There are two basic sub-models in Generative Adversarial Networks (GANs). The **generative** model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the **discriminative** model is analogous to the police, trying to detect the counterfeit currency (Goodfellow et al., 2020).

Generator Model how's generates new example and translates randomized data from a training dataset into a picture in this model, which is also known as a zero-sum game, which finds and improves any errors in the deepfake over numerous rounds, making it more difficult for deepfake detectors to decode it.

Discriminator Model is classify whether the generated image is real from the domain or fake, and tell the difference between actual and synthetic images, is mixed in with a stream of genuine images before being delivered to the discriminator. It's A neural network's purpose is to reduce mistakes.

This means minimizing the disparity between the fake and actual images in the case of deepfakes, the method is repeated until the output meets the desired degree of accuracy, with model-weight modifications.

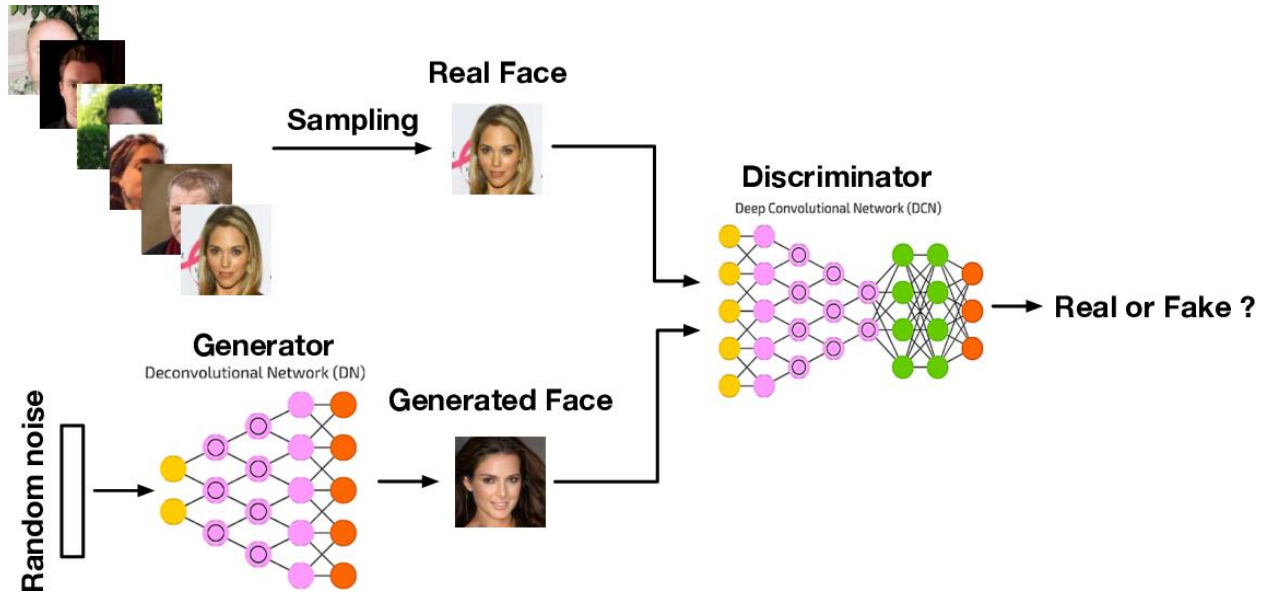


Fig. 3 A GAN's architecture. During the learning step, two deep neural networks (discriminator (D) and generator (G)) are trained concurrently. The discriminator is tuned to discriminate between genuine and created images, while the generator is tuned to deceive the discriminator into thinking it's seeing real images.

There are many aspects in human portraits that can be regarded as stochastic, such as the exact placement of hairs, stubble, freckles, or skin pores. Any of these can be randomized without affecting our perception of the image as long as they follow the correct distribution.(Karras et al., 2019)

2.3.2 Autoencoders

The autoencoder is a deep learning AI software tasked with analyzing video clips to determine how a person appears from various angles and environments and then mapping that person onto the individual in the target video using common traits. Today **data denoising** and **dimensionality reduction** for data visualization are considered as two main interesting practical applications of autoencoders(Hubens, n.d.), autoencoder is a type of unsupervised neural network that may reduce the dimensionality of raw data and produce an output that is identical to the input, there are two main types of autoencoders **encoders**, and **decoders** make up autoencoders.

The encoder compresses the image and transmits it to the decoder when data is fed via the first layer the input layer of the autoencoder's neural network, after then the decoder tries to reassemble the original data, deepfakes use autoencoders by training two network pairs, one for the source-image dataset and the other for the target-image dataset.

The encoder network, which allows the encoder to understand the structure of a human face, is shared by the two pairs. In the reconstruction phase, the source image is passed through a decoder that has been trained for the target image and synthesizes the two images.

The encoder and decoder can take any form, such as deep neural networks, convolutional neural networks (CNN) or recurrent neural networks, early autoencoders were used for dimensionality reduction, feature extraction and learning manifolds.(Daly et al., 2022)

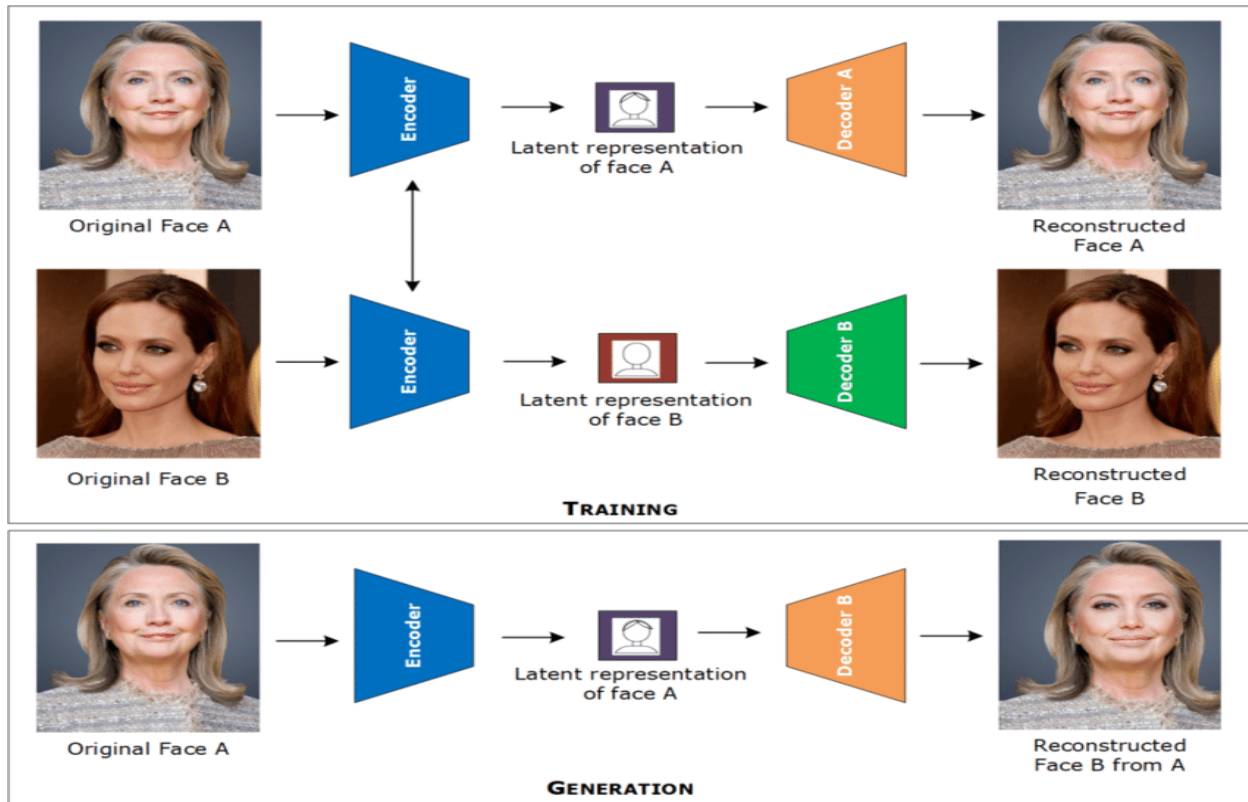


Fig. 4 Using an auto-encoder and decoder, a Deepfake is created. During training, the same encoder-decoder pair is used to learn the latent features of the faces, but during generation, the decoders are swapped, such that latent face A is exposed to decoder B, resulting in face A with the features of face B.

2.3.3 Face reenactment (face synthesis)

Face reenactment algorithms aim to manipulate people's facial expressions and movements and transmit them to another target person's face, that allows to create movies persuading someone to do something that does not exist.

Face reenactment was proposed using a face template that was updated using different expression parameters, the majority of future work is based on techniques like this after the computational power has improved in recent years, in which a parametric model is used to alter facial photographs, these approaches could produce highly realistic facial images, however the resulting images frequently lack temporal coherence.

Face reenactment it's categorized as a model-based approach & these methods consist of **three steps**:

First is Face capturing tracking the face template, so using visual flow as measurements of appearance and speed to fit the face in the database, or hiring RGB or RGB-D camera to pick up face movements.

Recent developments in facial detection methods enable us to track the facial component of input in an effective way like the eyes and mouth. **Second** after the face is picked up, Many studies will later fit into space movement or parameters, Including the head lay, Or look-eye, Or PCA coefficients on 3D model rules and even the details of 3D facial networks, **Third** Once a model has been fitted the next step is to re-introduce a new video.



Fig. 5. The proposed ReenactGAN is capable of manipulating a target face in a video by transferring movements and facial expressions from an arbitrary person's video, ReenactGAN, is capable of transferring facial movements and expressions from an arbitrary person's monocular video input to a target person's video(Wu et al., 2018).

2.3.4 Face swapping

Face swapping in which two people's identities are swapped in two videos has gained popularity in recent years. The very popular term "DeepFake" is referred to a deep learning based technique able to create fake videos by swapping the face of a person by the face of another person (Tolosana et al., 2020). And it's built convolutional neural networks (CNNs) to capture the appearance of target identity from an unstructured photo collection, allowing high-quality face-swapping images to be generated.

There is a novel method for generating films using a single RGB image and a source video sequence using source textures and a single target texture, a deep generative network was utilized to infer per-frame texture deformations of the target identity, the newly produced face might be composited onto the source footage, replacing the old face.

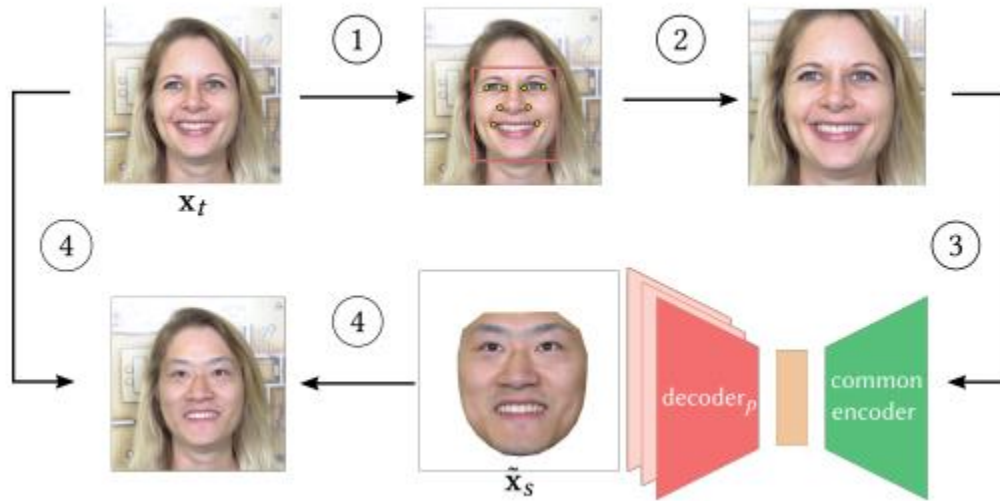


Fig. 6. they reverse the image normalization using the parameters saved from the second step and use their multi-band blending to swap the target with the source face producing realistic results.

The use of CNNs to generate face-swapping photos was widely credited as the basis for deepfake algorithms. A worldwide wave of people began making face-swapping movies for various reasons both positive and negative, Faceswap-GAN is a more advanced variant of the deepfake algorithms, to boost the efficiency of the autoencoder, adversarial loss and perceptual loss were introduced to generate more realistic faces.

The open-source deepfake generating framework was developed to provide an imperative and easy-to-use pipeline for people without professional experience, FaceShifter has been proposed in recent works for occlusion aware face shifting with great fidelity, FaceShifter develops high-fidelity swapped faces by undertaking complete integration of face traits, unlike prior face-swapping research that only used restricted information from target photos to synthesize faces, to integrate face information and recover an anomaly region, AEI-Net and HEAR-Net were used, respectively, experiments have shown that it outperforms previous face-swapping algorithms.

Recent deepfake technologies have produced highly lifelike videos that are difficult to tell apart from the human eye, to create a face-swapping video, the generative method must be used to all frames of the target video.

The deepfake algorithm which implements faceswapping while preserving the source expressions is without a doubt the most important aspect of video creation, Faceswapping deepfake techniques are generally based on autoencoder, which is commonly used for data reconstruction applications.

An encoder and a decoder are the two halves of an autoencoder the encoder extracts latent features from the image which are subsequently fed into the decoder to rebuild the original image.

Two autoencoders are taught to swap faces between source and target video frames in the deepfake algorithm, two encoders with the same weights are trained to extract similar features in source and target faces during the training phase, the extracted features are then sent into two decoders that rebuild faces the process of creating face-swapping video frames, each video frame begins with the detection of the face area

then to accomplish face alignment facial landmarks are extracted, after that the face-aligned image is fed into the deepfake algorithm (autoencoder or GAN), which generates a synthetic face.

The landmarks of the left and right brows, as well as the bottom mouth, are used to construct a customized mask to reduce blending artifacts only the content inside the mask is kept after merging the synthetic face to the source image in this way.

Finally, a postprocessing technique is added to the resulting image to make it more realistic.

2.3.5 DEEFAKE DETECTION

Deepfake films are becoming more and more dangerous to people's privacy and social security, several approaches for detecting altered videos have been proposed; early attempts mostly focused on inconsistencies created by the face synthesis process, but contemporary detection methods primarily focus on fundamental traits.

Detection of counterfeit content has raised attention in the last few years for the advances in deepfake generation. The rapid growth of machine learning techniques, particularly deep learning, can predict fake content in several application domains, including fake image and video manipulation(Passos et al., 2022).



Fig. 7. Conventional deepfake detection methods can only classify a given cropped face as real or fake. Recently developed methods can not only recognize the authenticity of faces but also localize manipulated areas at the bounding-box and pixel levels, namely detection and segmentation, respectively(Le et al., 2022).

There are five categories detection methods based on the features:

General network-based detection methods Deepfake video detection has been improved thanks to recent breakthroughs in face images classification that collected from the detected video are used to train the detection network in this method the trained network is then used to predict all of the frames in this video and finally the forecasts are calculated using an average or voting technique.

As a result, the neural networks are heavily reliant on detection accuracy, with no requirement to leverage certain identifiable traits existing network-based methods are divided into two types: transfer learning-based methods and detection approaches based on specially constructed networks.

Temporal-consistency-based methods video has a one-of-a-kind characteristic called time continuity video unlike photographs, is a sequence of several frames with a strong correlation and continuity between them due to flaws in deepfake algorithms the association between consecutive frames is broken when video frames are modified, resulting in face position shift and video flickering.

visual artifacts-based methods the generated face has to be blended into an existing background image in most existing deepfake methods resulting in intrinsic image discrepancies on the blending boundaries faces and background images come from different source images resulting in abnormal behavior of the synthetic image such as boundary anomaly and inconsistent brightness, Deepfake films are essentially detectable due to these visual abnormalities.

Camera-fingerprints-based methods Camera fingerprints are a type of low-energy noise that plays an essential role in forensics, particularly in source identification tasks Techniques based on camera fingerprints have gone through three stages: photoresponse nonuniformity (PRNU) patterns, noiseprint, and current video noise patterns.

Biological-signals-based methods, Detection based on biological signals is a novel concept that has gained traction in recent years. The main finding is that while GAN is capable of producing highly realistic looks, the naturally hidden biological signals are difficult to recreate, making it challenging to synthesize human faces with sensible behavior.

2.3.6 deepfake applications

Many industries, including custom web application development services, have profited from the acceleration of digital transformation and technology adoption. Deepfakes are one of the many amazing technologies that have arisen from it.

Deepfake technology combines artificial intelligence (AI), deep learning, and a Generative Adversarial Network (GAN) to create movies and images that appear real but aren't.

And this are examples of deepfake applications:

Zao

Zao, a Chinese deepfake technology software, gained traction and grew viral in China. Users may swap their features onto movie characters using Zao's deep fake technology, which allows them to upload any video and have a deepfake made in minutes. The program is exclusively available in China and makes stunningly realistic films in a matter of minutes. Users can choose from a large library of films and photographs in the app. Zao's algorithm is primarily trained on Chinese faces, therefore it may appear odd on other people's faces.

Reface

It is an artificial intelligence-powered tool that lets users swap faces in movies and GIFs. Reface was once known as Doublicat, a game that became viral shortly after its release. You can exchange faces with celebrities, memes, and make humorous movies with Reface. Face embeddings are used smartly by the app to make the swaps. Reface AI is the name of the technology, which is based on a Generative Adversarial Network. The most recent update is Reface's new feature, which allows users to post content other than selfies. Change Animation is a new feature that allows users to contribute content other than selfies, such as images of any humanoid being, animate it, and swap faces.

MyHeritage

is a genealogical website that features a deepfake feature in its app. Deep Nostalgia is a technique used by the startup that allows users to animate vintage photos. The MyHeritage nostalgia function went viral, and

social media were inundated with many experimental photos. The eyes, face, and mouth of the photographs are animated using this deepfake technology, which makes the eyes, face, and mouth move slightly.

FaceApp

Because of its unique capabilities that allow users to apply aging effects, this editing tool has recently become popular. In recent months, social networks have been filled with users who have experimented with the various filters. This is a free app, which makes it even more popular among users. FaceApp combines artificial intelligence, powerful machine learning, deep learning technologies, and an image recognition system to create a unique experience.

Deepfakes Web

It's a cloud-based deepfake software that works online. Deepfakes Web allows users to produce deepfake videos on the Web, and unlike other programs, curating a deep fake video takes nearly 5 hours. Using its deepfake AI-based algorithm and deep learning technology, it learns and trains from the videos and photos supplied. If you want to learn more about the technology behind deepfakes and the intricacies of computer vision, this platform is a wonderful place to start. It enables users to reuse learned models in order to improve the video and make deepfakes without having to utilize a trained model.

DeepArt

DeepArt is not a deepfake video program, as the name implies, but it does make deepfake photos by turning them into art. The software transforms uploaded photos into famous fine art paintings and recreates artistic imagery using a neural style transfer technique and AI. DeepArt is a free program that includes over 50 different art styles and filters. Standard, HD, and Ultra HD features are available in the app, with the latter two being paid editions. The photographs made by the program can be downloaded and shared by users.

Wombo

is an artificial intelligence-powered lip-sync tool that allows users to turn any face into a singing face. There is a collection of songs from which users can choose one and have the chosen character sing it in a picture. The program generates singing videos with a Photoshop-like appearance, making them appear dynamic and unreal. Wombo makes use of artificial intelligence (AI) to make the deepfake scenario possible.

DeepFace Lab

It is a Windows application that allows you to make deepfake films. Rather than viewing deepfake technology as a game, this software tool enables users to better grasp and comprehend the technology. Deep learning, machine learning, and human image synthesis are all employed. DeepFace Lab is a platform designed primarily for researchers in the fields of deep learning and computer vision. It is not a user-friendly platform. To use the program, the user must first understand the documentation and have a powerful PC with a high-end GPU.

Face Swap Live

is a real-time face-swapping smartphone application that allows users to switch faces with another individual. Users may also use the app to produce films, apply various filters to them, and publish them immediately on social media. Face Swap Live, unlike most other deepfake apps, does not use static photos and instead allows users to execute live face swaps using their phone's camera. Face Swap Live isn't a full-

fledged deepfake app, but it is a good place to start if you want to have some fun with deepfakes. Computer vision and machine learning are used well in the app.

AvengeThem

is a website that allows users to upload a GIF and have it superimposed on the faces of characters from the Avengers film franchise. Although it is not entirely deepfake, it does use a 3D model to replace and animate the faces. The website offers roughly 18 GIFs, and creating this effect, which does not look particularly realistic, takes less than 30 seconds.

CHAPTER 3: METHODOLOGY

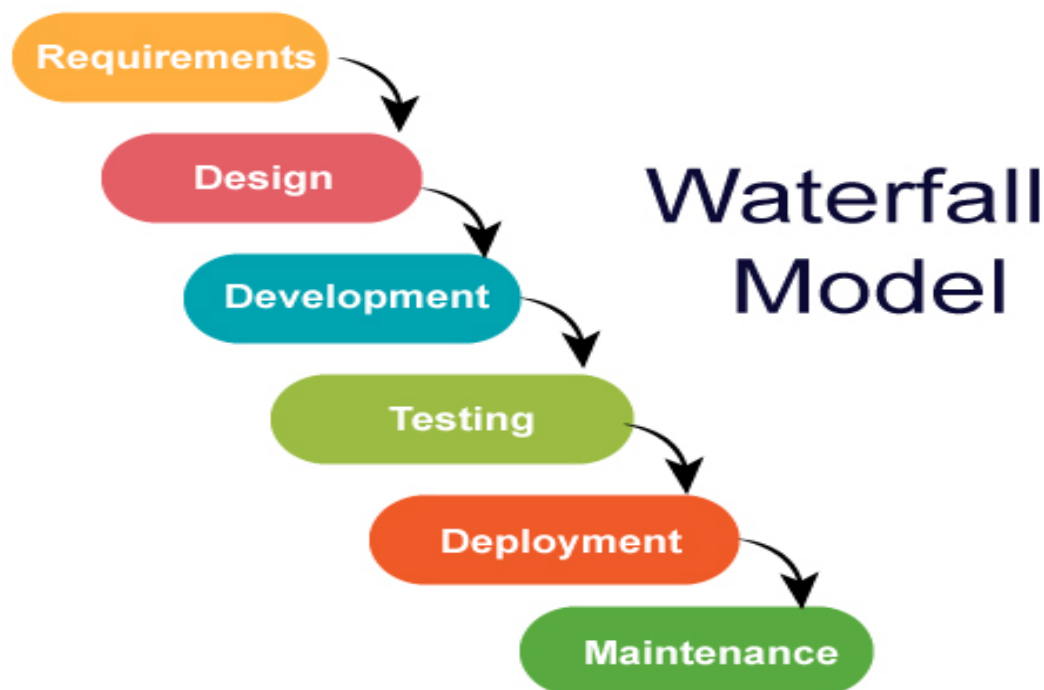
Many businesses utilize the software development life cycle (SDLC) to plan, develop, and test the quality of software products. The SDLC's main goal is to deliver a high-quality product. The software life cycle is defined as the time between when a software product is created and when it is no longer available for use, according to the IEEE standard lexicon of software engineering vocabulary.

There are many models in SDLC's but we decided to use waterfall model for our project and we chose waterfall because even though it is the oldest Model, but it is very simple and clear and we can work on it and deliver it in the time, and one of its benefits is that it is understood and its environment is stable and the requirements are available and unchanging and the required tools and techniques are available.

The waterfall model represents the software development process as a linear, sequential process, which means that any phase of the development process can only begin once the previous one has completed. In the waterfall model the phases do not overlap.

The Waterfall Approach was the first SDLC Model to be widely used in Software Engineering to ensure project success. The "waterfall" technique divides the entire software development process into various phases. In this Waterfall approach, the output of one phase is typically used as the input for the next phase.

The different phases are illustrated in the diagram below.



The activities of waterfall model phases:

Requirements:

In this phase, you should search for information and figure out the requirements of the project.

There are different ways to gather information about the project, for example, Brainstorm, interviews, and most importantly, nowadays the Internet.

At the end of this phase, you should have a clear understanding to the requirements and share the document with the team

Design:

In this phase the team creates the system based on the specified requirements. During this phase, no coding is done, but the team develops specifications such as what programming language to use and what are the hardware requirements.

Development:

In this phase the team start coding. Programmers take the data from the previous step and turn it into a working product. They usually write code in little chunks that are combined at the end of this phase or the start of the next.

Testing:

After each unit has been tested, all of the units built during the implementation phase are merged into a system. The program that has been created must be constantly tested to see if there are any defects or problems. Testing is performed to ensure that the client does not have any issues installing the software.

Deployment:

After the program has been fully tested and verified, a best practice is to load the application using a test system before putting it into production. The product is deployed in the client environment or released into the market once functional and non-functional testing is completed.

Maintenance:

After installation, this process involves making changes to the system or a single component to change qualities or improve performance. These changes are made in response to customer requests for changes or to defects discovered while the system is in use. The developed software is maintained and supported.

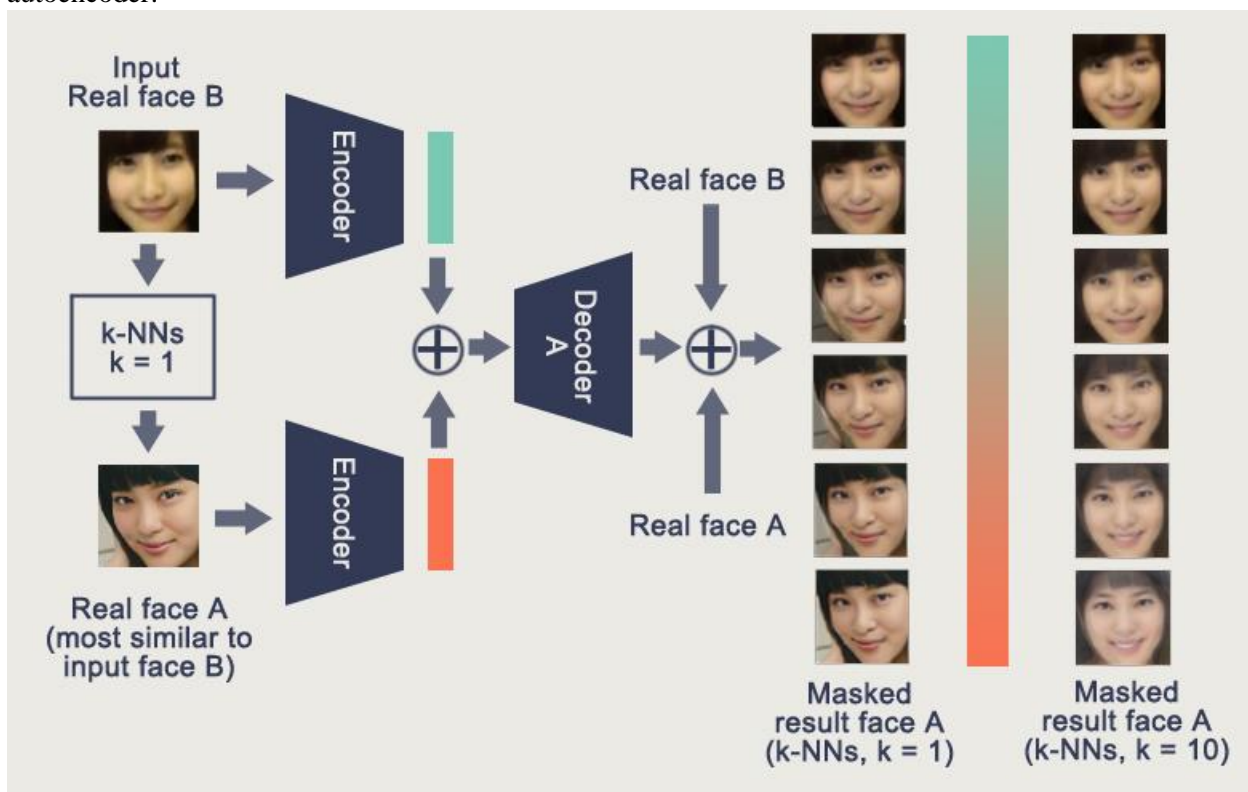
CHAPTER 4: SYSTEM ANALYSIS

4.1 Product Features:

4.1.1 Face Swap:

Face Swap is a popular technology used by many applications in which someone's faces are replaced by someone else's face with a video or photo, and it is a deepfake technology.

It also created convolutional neural networks (CNNs) to capture the appearance of a target identity from an unstructured photo collection, allowing for the creation of high-quality face-swapping images, CNNs were widely acknowledged as the foundation for deepfake algorithms since they were used to generate face-swapping pictures. Faceswap-GAN is a more advanced form of the deepfake techniques, with adversarial loss and perceptual loss added to generate more realistic faces to enhance the efficiency of the autoencoder.



An autoencoder is made up of two parts:

An encoder and a decoder. The encoder collects latent features from the image, which are then fed into the decoder to reconstruct the original image.

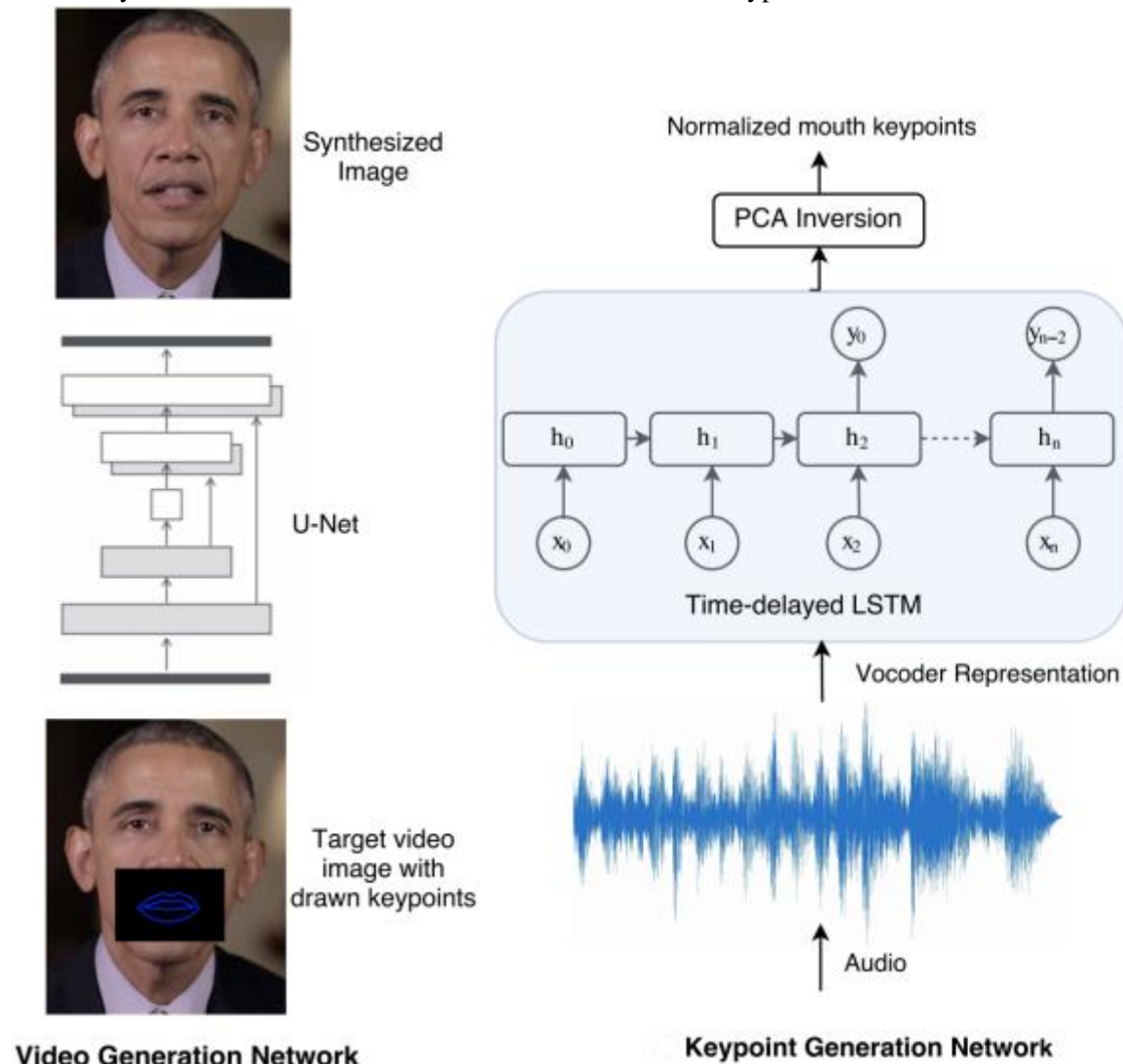
In the deepfake algorithm, two autoencoders with the same weights are trained to extract similar features in source and target faces during the training phase, and the extracted features are then sent into two decoders that rebuild faces the process of creating face-swapping video frames, each video frame begins with the detection of the face area, then to achieve face alignment facial landmarks a (autoencoder or GAN).

4.1.2 Lip Synchronization:

The Lip Syncing is the outcome of combining the latest research on generating faces using ML. and the astonishing progress in speech synthesis, which is less popular than generating faces and synthesizing speech. Thanks to that it is possible nowadays to generate artificial videos of a person speaking an arbitrary text. using models that can be trained by using videos that clearly shows the mouth of a person speaking, and the corresponding text.

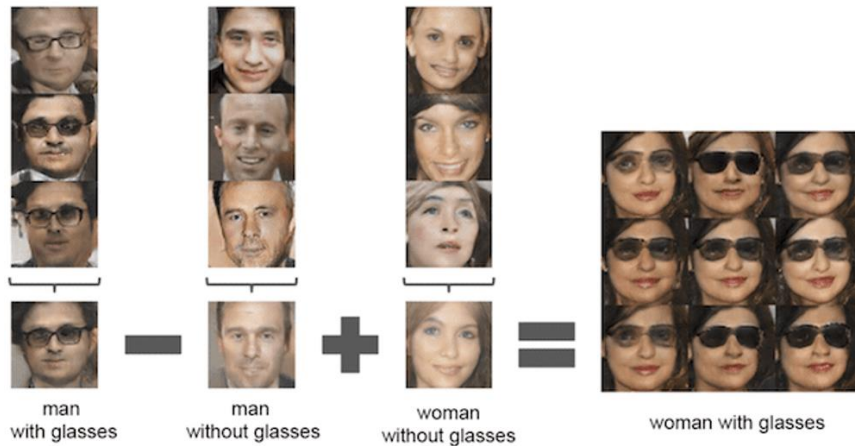
As a result, a system is created that generates speech from any text and alters the mouth region of an existing video to make it look natural and lifelike.

It is mainly divided into two networks, Video Generation and Keypoint Generation



4.1.3 Face Generation

The task of producing (or interpolating) new faces from an existing dataset is known as face generation.



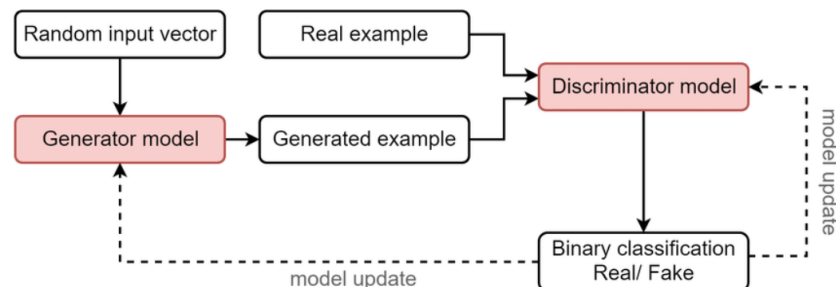
Example of Vector Arithmetic for GAN-Generated Faces

When given a random vector or matrix, Generator creates a fake sample. The discriminator attempts to determine whether the output of the generator is genuine or not. The discriminator and generator are trained one after the other: the discriminator for a few epochs, then the generator for a few epochs, and so on. As a result, both the generator and the discriminator improve.

It has two neural networks: a **Generator** and a **Discriminator**.

The generator generates a fake sample when a random vector or matrix is given. The discriminator tries to detect if the generator's output is real or fake. Training happens one after the other: the discriminator is trained for a few epochs, then the generator for a few epochs, and repeat. This way both the generator and the discriminator get better.

Generative Adversarial Network is used for training.



4.2 Functional Requirements:

Use-Case 1

Identifier		UC-1
Purpose		Input video
Priority		High
Pre-conditions		High quality source face , Target face (picture or video)
Post-conditions		Remove noisy and outliers frames
Typical Course of Action		
S#	Actor Action	System Response
1	Input video source face	The train model will take all frames of the source face from different angles
2	Input video target face	The train model will take all frames of the target face from different angles
Alternate Course of Action		
S#	Actor Action	System Response
1	The entry source picture is bad quality	System display enter high quality source picture

Use-Case 2

Identifier	UC-2	
Purpose	Train model	
Priority	High	
Pre-conditions	Super computer with high CPU and Graphic card, cleaned data to train	
Post-conditions	Data training will take a lot of time	
Typical Course of Action		
S#	Actor Action	System Response
1	Face Swap	Train model on extracted source frames and swap the target frames
Alternate Course of Action		
S#	Actor Action	System Response
1	User save the project (so he can stop the training process and continue again).	Project saved

Use-Case 3

Identifier	UC-3	
Purpose	Synchronizing the lips with the audio	
Priority	High	
Pre-conditions	Text or high-quality audio	
Post-conditions	If the input is text turn it to audio using TTS, otherwise clean the voice.	
Typical Course of Action		
S#	Actor Action	System Response
1	Input text.	Scan the text and make it ready to be synthesized as a speech.
2	Choose the preferred voice for the text.	Synthesize the text into speech in the chosen voice.
Alternate Course of Action		
S#	Actor Action	System Response
1	Input voice	Check the voice for noise and remove it.
2	Record a voice and submit it	Check if the recorded voice is useable if not show a prompt to record again.

Use-Case 4

Identifier	UC-4	
Purpose	Generate a new picture	
Priority	High	
Pre-conditions	Use more than one picture	
Post-conditions	Improve the final image quality	
Typical Course of Action		
S#	Actor Action	System Response
1	Input the picture	Upload the images and make them ready for the new image generation process
2	Choose the main image	check the picture and start the generation process
Alternate Course of Action		
S#	Actor Action	System Response
1	Save the result	the picture generated and saved

Use-Case 5

Identifier	UC-5	
Purpose	Generative Adversarial Networks (GANs)	
Priority	High	
Pre-conditions	Using high quality codes & model	
Post-conditions	Using specialized software to run it	
Typical Course of Action		
S#	Actor Action	System Response
1	Generator model	generates new example and translates randomized data from a training dataset into a picture in this model
2	Discriminator model	is classify whether the generated image is real from the domain or fake
Alternate Course of Action		
S#	Actor Action	System Response
1	Error with the video	Enhance codes and model

4.3 Nonfunctional Requirements

Performance Requirements

- The duration of the training process for each feature should not take more than a day.
- The resulting models should be able to be used in real time without causing significant performance cost.
- The input lag should not exceed 60 milliseconds.

Safety Requirements

- Keeping backup copies of databases in several places to protect them from loss or damage
- Databases must be encrypted and kept in secure locations and security precautions should be increased to prevent unauthorized access
- The appropriate age for the user must be specified.
- The mobile number or the national ID is mandatory to know the identity of the user.
- Taking into account not to use the program illegally, such as impersonating and harming others.
- System administrators must take strict and appropriate measures to maintain and ensure the integrity of the system by the target user.
- The system administrators follow up in strict confidence the reports of misuse observed by other users.

Security Requirements

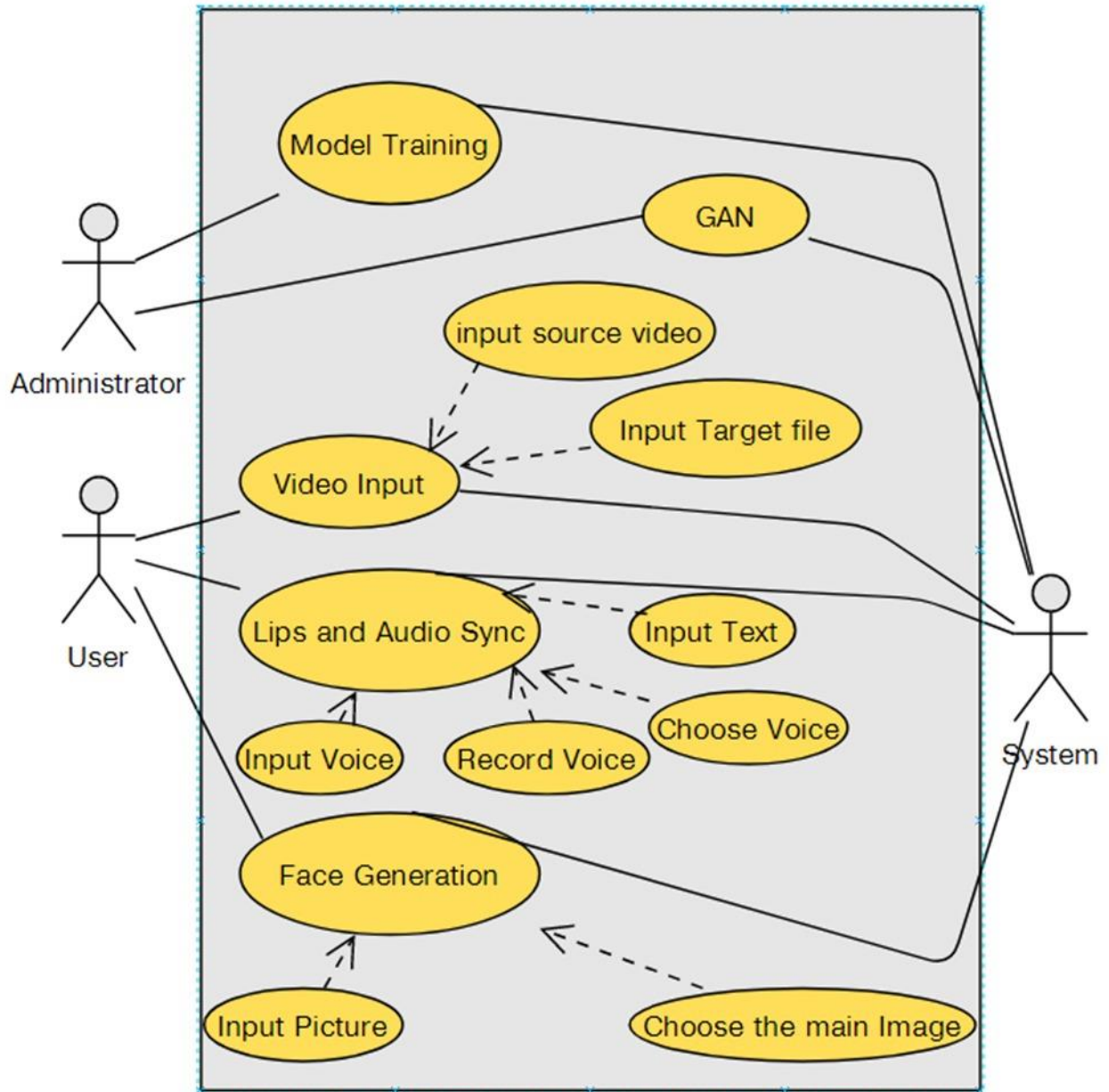
- If permanent storage is required, sensitive data should be encrypted; otherwise, it should be erased after use.
- During operation, only user-owned hardware should have access to user data; no cloud processing is allowed.

Software Quality Attributes

- The system in the backend that is responsible for making the DeepFake should be online 24/7.
- The app should be easy to use with a friendly UI that account for casual users and power users.
- The app should be able to process multiple DeepFake at the same time.
- The app should be available for Android and IOS devices.

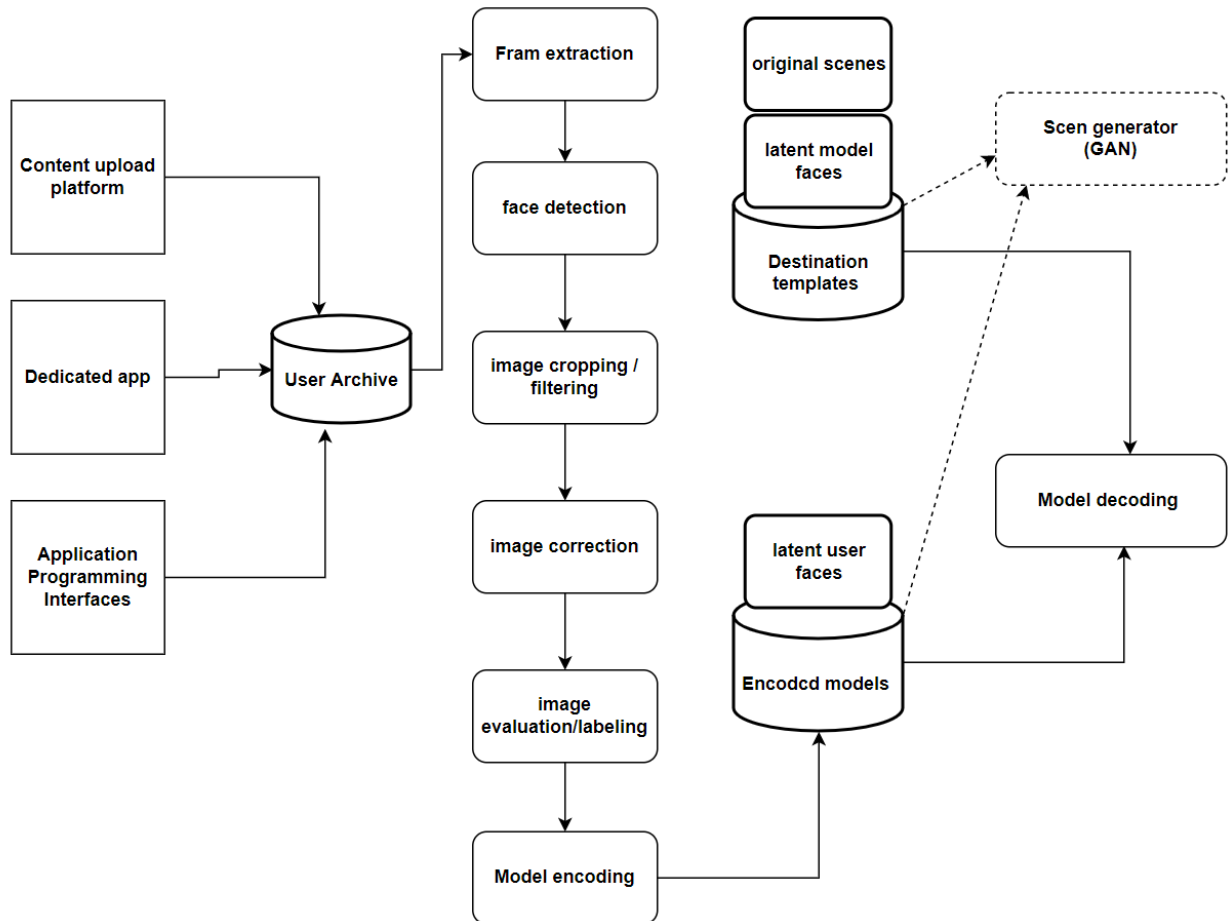
4.4 Analysis Models

Use-Case Diagram

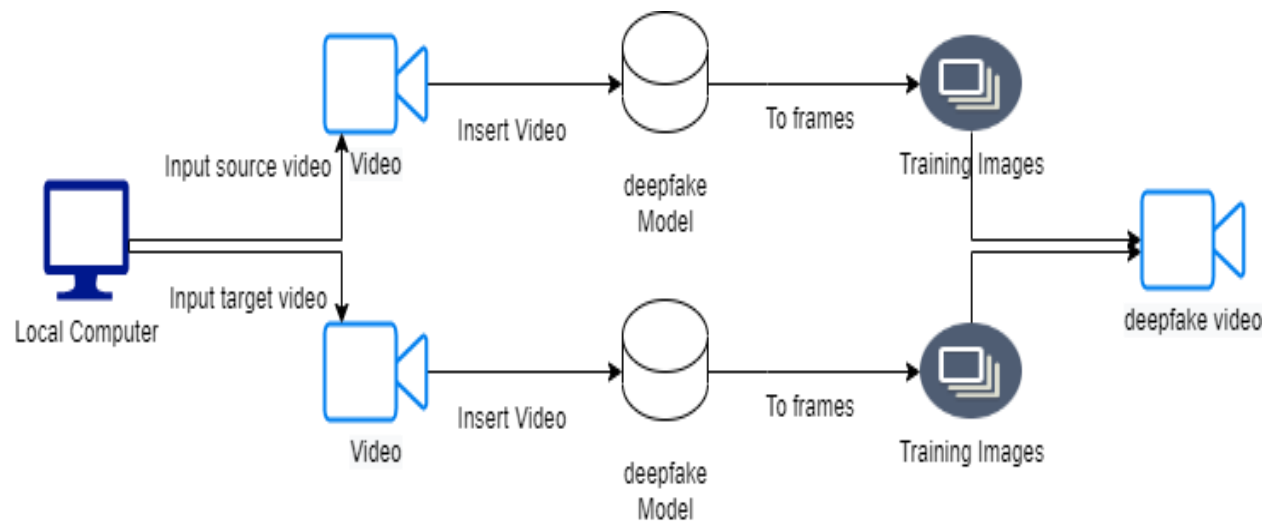


CHAPTER 5: SYSTEM DESIGN

component diagram



deployment diagram



CHAPTER 6: SYSTEM IMPLEMENTATION

We downloaded the Jupyter Notebook application, Anaconda is a Python distribution (prebuilt and preconfigured collection of packages) that is commonly used for data science. The Anaconda distribution includes the Conda package manager in addition to the preconfigured Python packages and other tools.

6.1 Application Implementation

The whole project will be implemented in Python and with the help of Python tools and libraries, we chose Jupyter Notebook application to implement the entire project as it supports all Python libraries and all tools to do artificial intelligence and deep learning projects, and we have prepared the system requirements required to work the program efficiently and correctly.

6.2 Deepfake applications

We have selected three applications for Deepfake to work on it "Face swap for pictures", "face swap in real time with webcam pc", "Face Generation using GAN's".

6.2.1 Face swap for pictures

At Face Swap we used artificial intelligence and deep learning and making a model that detects faces from photos and recognizes the details of the face with 68 marks such as the nose and mouth etc. and identifies and replaces them with another face from the input image.

6.2.1.1 importing the required libraries

```
import cv2
import numpy as np
import dlib
```

cv2 (Computer vision II): OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems.

NumPy: It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

Dlib: is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems.

6.2.1.2 using Dlib (c++ toolkit) and its functions to detect the face landmark points

```
|  
frontal_face_detector = dlib.get_frontal_face_detector()  
frontal_face_predictor = dlib.shape_predictor("C:\\dataset\\shape_predictor_68_face_landmarks.dat")
```

initialized dlib library's face detector, create dlib library's facial landmark predictor.

the pretrained model which have already complete training for detecting the face landmark points, we have around 68 landmarks points that we can input face image then the program will detect where the eyes nose etc.

we have downloaded the trained model from <http://dlib.net/files/> , file name is shape_predictor_68_face_landmarks.dat.

6.2.1.3 using cv2 (computer vision library) and its functions to Reading source & destination image and convert them to gray scale and show him.

```
source_image = cv2.imread("C:\\images\\ww.jpg")  
source_image_grayscale = cv2.cvtColor(source_image, cv2.COLOR_BGR2GRAY)  
cv2.imshow("source_image",source_image)  
|  
destination_image = cv2.imread("C:\\images\\jason.jpg")  
destination_image_grayscale = cv2.cvtColor(destination_image, cv2.COLOR_BGR2GRAY)  
cv2.imshow("destination_image",destination_image)
```

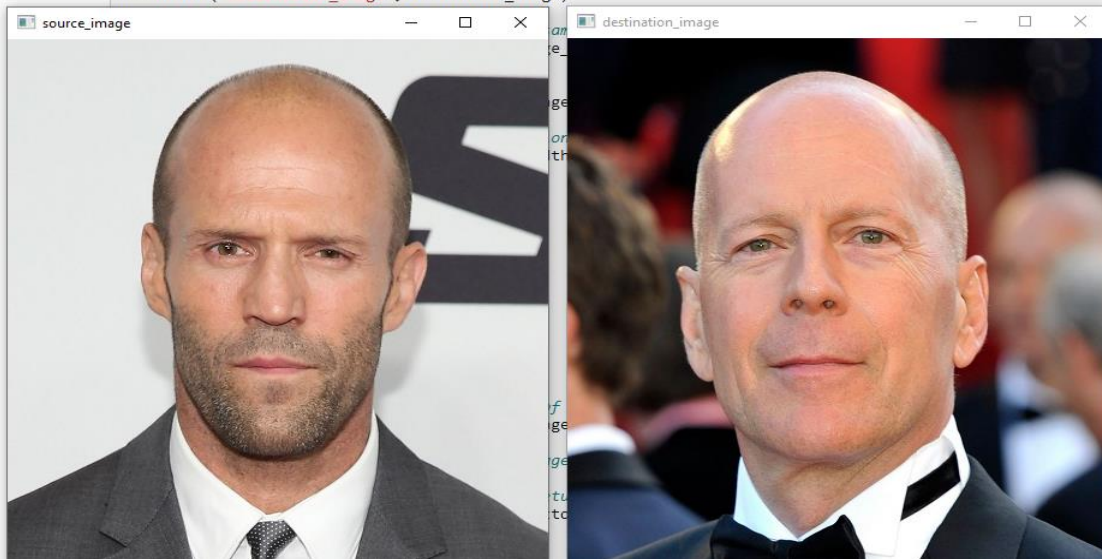

6.2.1.4 Showing Source & destination image using imshow function

```
In [*]: import cv2
import numpy as np
import dlib

frontal_face_detector = dlib.get_frontal_face_detector()
frontal_face_predictor = dlib.shape_predictor("C:\\dataset\\shape_predictor_68_face_landmarks.dat")

source_image = cv2.imread("C:\\images\\jason.jpg")
source_image_grayscale = cv2.cvtColor(source_image, cv2.COLOR_BGR2GRAY)
cv2.imshow("source_image", source_image)

destination_image = cv2.imread("C:\\images\\www.jpg")
destination_image_grayscale = cv2.cvtColor(destination_image, cv2.COLOR_BGR2GRAY)
cv2.imshow("destination_image", destination_image)
```

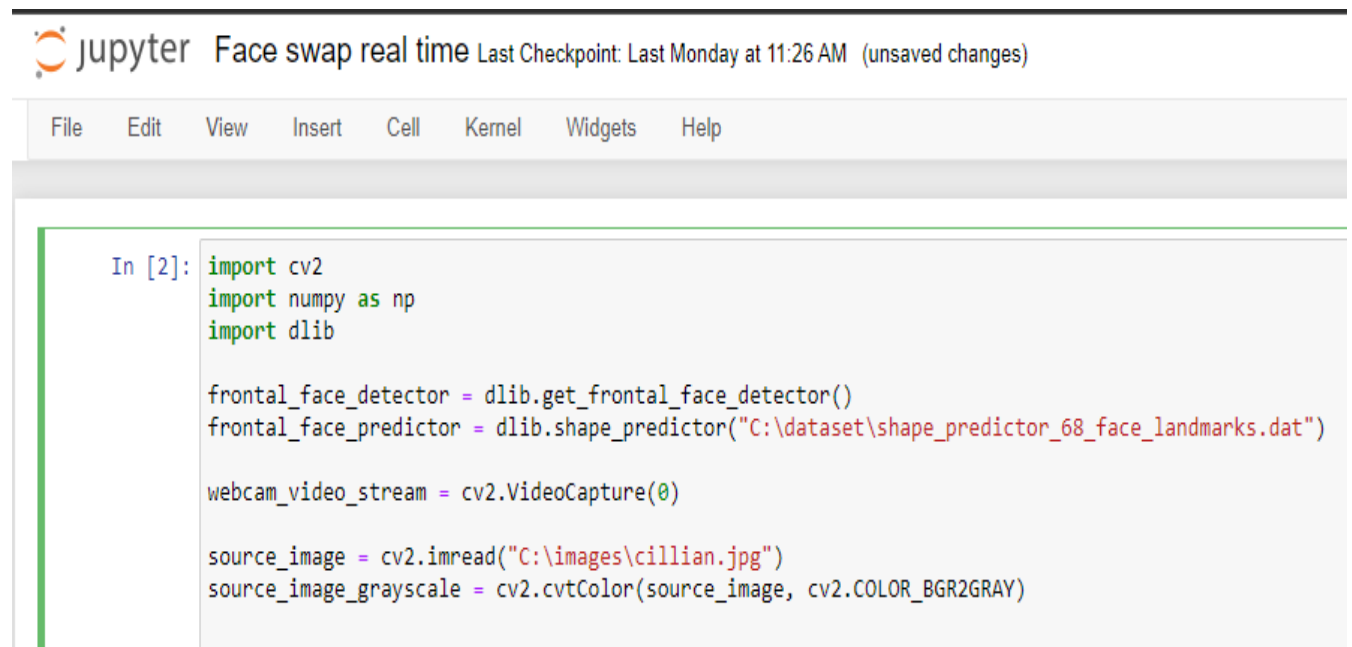


6.2.2 Face swap in real time with webcam pc

At Face Swap in real time we used artificial intelligence and deep learning and making a model that detects faces in real time from the webcam and replace them with source image and recognizes the details of the face with 68 marks such as the nose and mouth etc.

6.2.2.1 importing the required libraries

We do the same steps we did in Faceswap but this time we added Function to get the webcam video steam from my pc **cv2.VideoCapture(0)**.



The screenshot shows a Jupyter Notebook window titled "Face swap real time". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". Below the menu bar, the notebook content area displays the following code in a cell:

```
In [2]: import cv2
import numpy as np
import dlib

frontal_face_detector = dlib.get_frontal_face_detector()
frontal_face_predictor = dlib.shape_predictor("C:\\dataset\\shape_predictor_68_face_landmarks.dat")

webcam_video_stream = cv2.VideoCapture(0)

source_image = cv2.imread("C:\\images\\cillian.jpg")
source_image_grayscale = cv2.cvtColor(source_image, cv2.COLOR_BGR2GRAY)
```

6.2.2.2 Create infinite while loop

loop through the frames and returns true if video capturing has been initialized already using cv function **.isOpened**, and get the frame from **webcam_video_stream** using cv function **.read** which Grabs, decodes and returns the next video frame and returns false if no frames has been grabbed.

```
while (webcam_video_stream.isOpened):
    ret, current_frame = webcam_video_stream.read()
```

6.2.2.3 read the destination face

read the destination face image and convert it to grayscale using **cv2.cvtColor**

```
destination_image = current_frame
if destination_image is not None:
    destination_image_grayscale = cv2.cvtColor(destination_image, cv2.COLOR_BGR2GRAY)
```

6.2.2.4 loop through all the 68 landmark points in source image

loop through all faces found in the source image and the predictor takes human face as input and returns the list of facial landmarks then looping through all the 68 landmark points and add them into a tuple

```
for source_face in source_faces:
    source_face_landmarks = frontal_face_predictor(source_image_grayscale, source_face)
    source_face_landmark_points = []

    for landmark_no in range(0,68):
        x_point = source_face_landmarks.part(landmark_no).x
        y_point = source_face_landmarks.part(landmark_no).y
        source_face_landmark_points.append((x_point, y_point))
        cv2.circle(source_image, (x_point, y_point), 2, (255, 255, 0), -1)
        cv2.putText(source_image, str(landmark_no), (x_point, y_point), cv2.FONT_HERSHEY_SIMPLEX, .3, (255, 255, 255))
    cv2.imshow("1: landmark points of source", source_image)
```

6.2.2.5 detecting the faces in the webcam

for the destination video in the pc webcam find the faces in destination image Returns a numpy array containing a histogram of pixels in the image, the if condition to check if at least one face is present in the frame if there is a face that will print a number 1 last sentences, if there is no face in the frame will print the number 0.

```
destination_faces = frontal_face_detector(destination_image_grayscale)
print("The number of faces in video stream: {}".format(len(destination_faces)))

if(len(destination_faces)==1):
    for destination_face in destination_faces:
        destination_face_landmarks = frontal_face_predictor(destination_image_grayscale, destination_face)
        destination_face_landmark_points = []
```

6.2.3 Face Generation Using GAN's

The technology behind these kinds of AI is called a GAN, or “Generative Adversarial Network”. A GAN takes a different approach to learning than other types of neural networks(NN). GANs algorithmic architectures that use two neural networks called a Generator and a Discriminator, which “compete” against one another to create the desired result.

Our goal is to create a model capable of generating good human images that don't actually exist using deepfake and GAN's technology.

6.2.3.1 Importing required libraries

For the construction of this face generation model, we will utilize the TensorFlow and Keras deep learning frameworks for achieving our goals. we will also utilize other essential libraries, such as matplotlib, numpy, and OpenCV, for performing visualizations, computations, and graphics-related operations accordingly.

```
In [1]: import tensorflow as tf
import keras
from keras import layers
import numpy as np
import matplotlib.pyplot as plt
import cv2
import os
from tqdm import tqdm
import re
from tensorflow.keras.utils import img_to_array
```

6.2.3.2 Loading the dataset

This dataset contains 10,000 HD images, these images are great for training and testing models for face detection, particularly for recognizing facial attributes such as finding people with brown hair, are smiling, or wearing glasses. Images cover large pose variations, background clutter, diverse people, supported by a large number of images and rich annotations.

We have downloaded the dataset from Kaggle. Our objective is to create a model capable of generating realistic human images that do not exist in reality.

```
In [2]: # to get the files in proper order
def sorted_alphanumeric(data):
    convert = lambda text: int(text) if text.isdigit() else text.lower()
    alphanum_key = lambda key: [convert(c) for c in re.split('([0-9]+)',key)]
    return sorted(data, key = alphanum_key)
# defining the size of the image
SIZE = 128
_img = []
path = 'C:\without_mask'
files = os.listdir(path)
files = sorted_alphanumeric(files)
for i in tqdm(files):
    if i == 'seed9090.png':
        break
    else:
        img = cv2.imread(path + '/' + i, 1)
        # open cv reads images in BGR format so we have to convert it to RGB
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        #resizing image
        img = cv2.resize(img, (SIZE, SIZE))
        img = (img - 127.5) / 127.5
        imh = img.astype(float)
        _img.append(img_to_array(img))
```

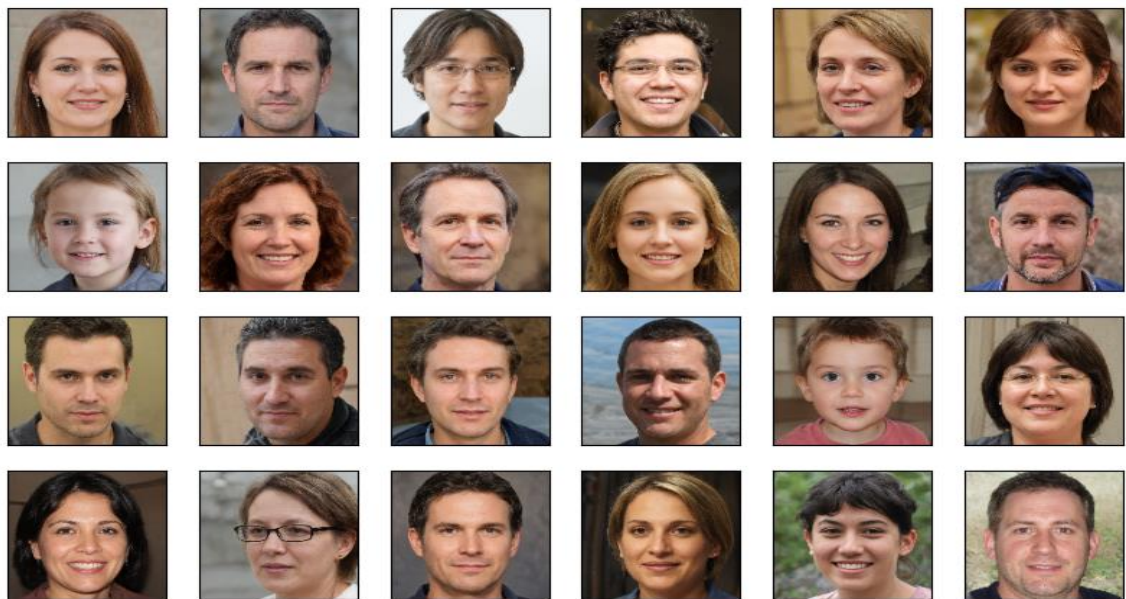
91%

| 9090/10000 [09:14<00:55, 16.38it/s]

6.2.3.3 Visualize dataset images

```
In [3]: def plot_images(sqr = 5):
plt.figure(figsize = (10,10))
plt.title("Real Images", fontsize = 35)
for i in range(sqr * sqr):
    plt.subplot(sqr,sqr,i+1)
    plt.imshow(_img[i]*0.5 + 0.5)
    plt.xticks([])
    plt.yticks([])

# to plot images
plot_images(6)
```



6.2.3.4 Constructing the Generator Model

Generator takes an array of random numbers having size equal to seed size and generates an image and the generated images with equal number of real face images.

```
In [5]: latent_dim = 100
def Generator():
    model = tf.keras.Sequential()
    model.add(layers.Dense(128*128*3, use_bias=False, input_shape=(latent_dim,)))
    model.add(layers.Reshape((128,128,3)))
    # downsampling
    model.add(tf.keras.layers.Conv2D(128,4, strides=1, padding='same',kernel_initializer='he_normal', use_bias=False))
    model.add(tf.keras.layers.Conv2D(128,4, strides=2, padding='same',kernel_initializer='he_normal', use_bias=False))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.LeakyReLU())
    model.add(tf.keras.layers.Conv2D(256,4, strides=1, padding='same',kernel_initializer='he_normal', use_bias=False))
    model.add(tf.keras.layers.Conv2D(256,4, strides=2, padding='same',kernel_initializer='he_normal', use_bias=False))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.LeakyReLU())
    model.add(tf.keras.layers.Conv2DTranspose(512, 4, strides=1,padding='same',kernel_initializer='he_normal',use_bias=False))
    model.add(tf.keras.layers.Conv2D(512,4, strides=2, padding='same',kernel_initializer='he_normal', use_bias=False))

    model.add(tf.keras.layers.LeakyReLU())
    #upsampling
    model.add(tf.keras.layers.Conv2DTranspose(512, 4, strides=1,padding='same',kernel_initializer='he_normal',use_bias=False))
    model.add(tf.keras.layers.Conv2DTranspose(512, 4, strides=2,padding='same',kernel_initializer='he_normal',use_bias=False))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.LeakyReLU())
    model.add(tf.keras.layers.Conv2DTranspose(256, 4, strides=1,padding='same',kernel_initializer='he_normal',use_bias=False))
    model.add(tf.keras.layers.Conv2DTranspose(256, 4, strides=2,padding='same',kernel_initializer='he_normal',use_bias=False))
    model.add(tf.keras.layers.BatchNormalization())

    model.add(tf.keras.layers.Conv2DTranspose(128, 4, strides=2,padding='same',kernel_initializer='he_normal',use_bias=False))
    model.add(tf.keras.layers.Conv2DTranspose(128, 4, strides=1,padding='same',kernel_initializer='he_normal',use_bias=False))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.Conv2DTranspose(3,4,strides = 1, padding = 'same',activation = 'tanh'))

    return model
```

6.2.3.5 Constructing the Discriminator Model:

We will construct the discriminator model for the generative network to classify the images as real or fake.

```
def Discriminator():
    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Input((SIZE, SIZE, 3)))
    model.add(tf.keras.layers.Conv2D(128,4, strides=2, padding='same',kernel_initializer='he_normal', use_bias=False))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.LeakyReLU())
    model.add(tf.keras.layers.Conv2D(128,4, strides=2, padding='same',kernel_initializer='he_normal', use_bias=False))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.LeakyReLU())
    model.add(tf.keras.layers.Conv2D(256,4, strides=2, padding='same',kernel_initializer='he_normal', use_bias=False))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.LeakyReLU())
    model.add(tf.keras.layers.Conv2D(256,4, strides=2, padding='same',kernel_initializer='he_normal', use_bias=False))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.LeakyReLU())
    model.add(tf.keras.layers.Conv2D(512,4, strides=2, padding='same',kernel_initializer='he_normal', use_bias=False))
    model.add(tf.keras.layers.LeakyReLU())
    model.add(tf.keras.layers.Flatten())
    model.add(tf.keras.layers.Dense(1,activation = 'sigmoid'))
    return model
```


6.2.3.6 Training and Generating Images

The final step of constructing our project is to train the models for a number of epochs.

```
: def train_steps(images):
    noise = np.random.normal(0,1,(batch_size,latent_dim))
    with tf.GradientTape() as gen_tape , tf.GradientTape() as disc_tape:
        generated_images = generator(noise)
        fake_output = discriminator(generated_images)
        real_output = discriminator(images)

        gen_loss = generator_loss(fake_output)
        dis_loss = discriminator_loss(fake_output, real_output)

    gradient_of_generator = gen_tape.gradient(gen_loss, generator.trainable_variables)
    gradient_of_discriminator = disc_tape.gradient(dis_loss, discriminator.trainable_variables)

    optimizer.apply_gradients(zip(gradient_of_generator,generator.trainable_variables))
    optimizer.apply_gradients(zip(gradient_of_discriminator, discriminator.trainable_variables))

    loss = {'gen loss':gen_loss,
           'disc loss': dis_loss}
    return loss

def plot_generated_images(square = 5, epochs = 0):

    plt.figure(figsize = (10,10))
    for i in range(square * square):
        if epochs != 0:
            if(i == square //2):
                plt.title("Generated Image at Epoch:{}\n".format(epochs), fontsize = 32, color = 'black')
        plt.subplot(square, square, i+1)
        noise = np.random.normal(0,1,(1,latent_dim))
        img = generator(noise)
        plt.imshow(np.clip((img[0,...]+1)/2, 0, 1))

        plt.xticks([])
        plt.yticks([])
        plt.grid()

    import time
def train(epochs,dataset):

    for epoch in range(epochs):
        start = time.time()
        print("\nEpoch : {}".format(epoch + 1))
        for images in dataset:
            loss = train_steps(images)
        print(" Time:{} ".format(np.round(time.time() - start),2))
        print("Generator Loss: {} Discriminator Loss: {}".format(loss['gen loss'],loss['disc loss']))

#training the dataset
train(5,dataset)
```

CHAPTER 7: TESTING & EVALUATION

Identifier	TC-1
Priority	High
Related requirements(s)	UC-1
Short description	Picture face swap
Pre-condition(s)	none
Input data	(Picture to be used as Destination face)
Detailed steps	...
Expected result(s)	A clear picture of a face that can be scanned
Post-condition(s)	Destination picture processed and prepared for Applying the source image to it.

Identifier	TC-2
Priority	High
Related requirements(s)	UC-2
Short description	Picture face swap
Pre-condition(s)	none
Input data	(Picture to be used as source face)
Detailed steps	...
Expected result(s)	A clear picture of a face that can be scanned
Post-condition(s)	Source picture processed and prepared to be applied to the destination face.

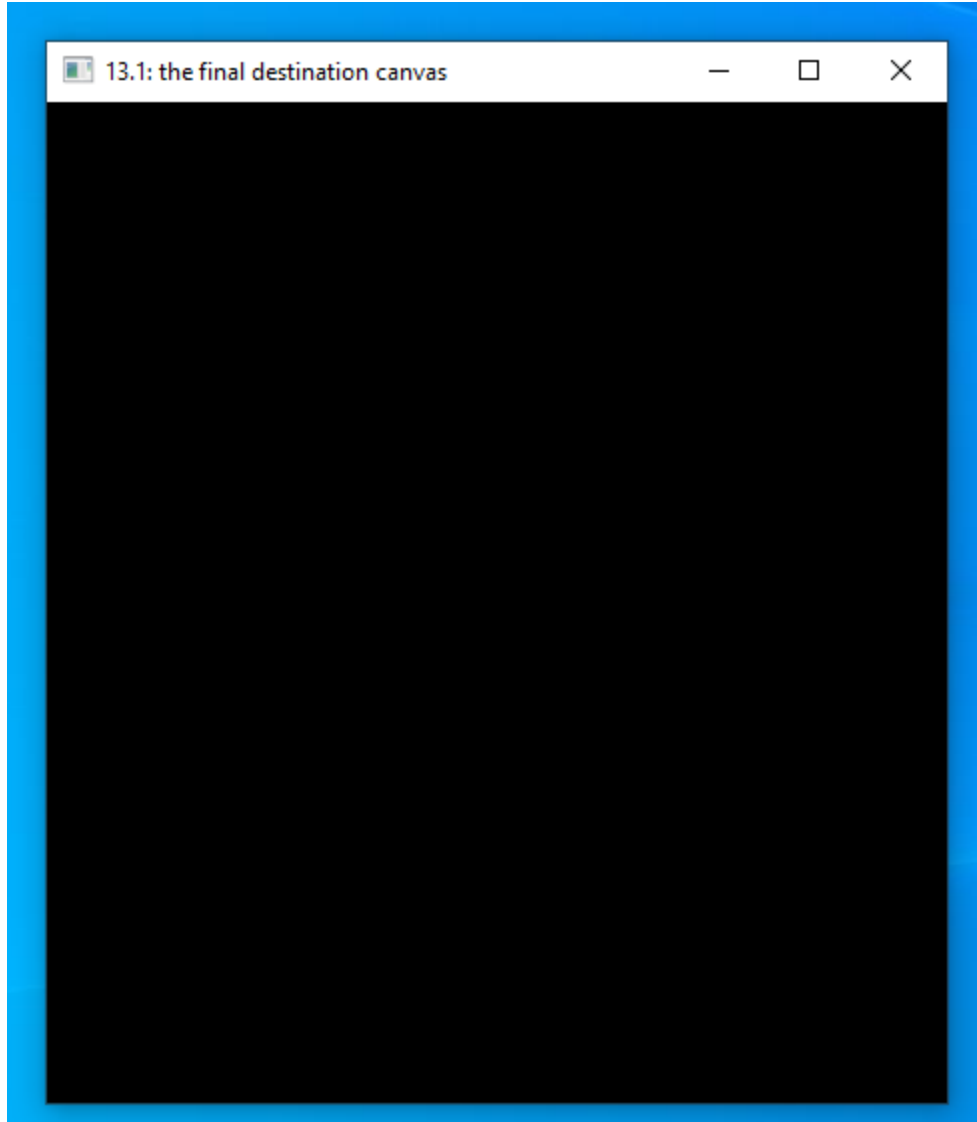
Identifier	TC-3
Priority	High
Related requirements(s)	UC-3
Short description	Webcam face swap
Pre-condition(s)	Source face
Input data	(Webcam face)
Detailed steps	...
Expected result(s)	Clear and fully visible face through webcam
Post-condition(s)	Webcam face(destination face) is processed in real time and applying the source face to it

CHAPTER 8: RESULTS AND ANALYSIS

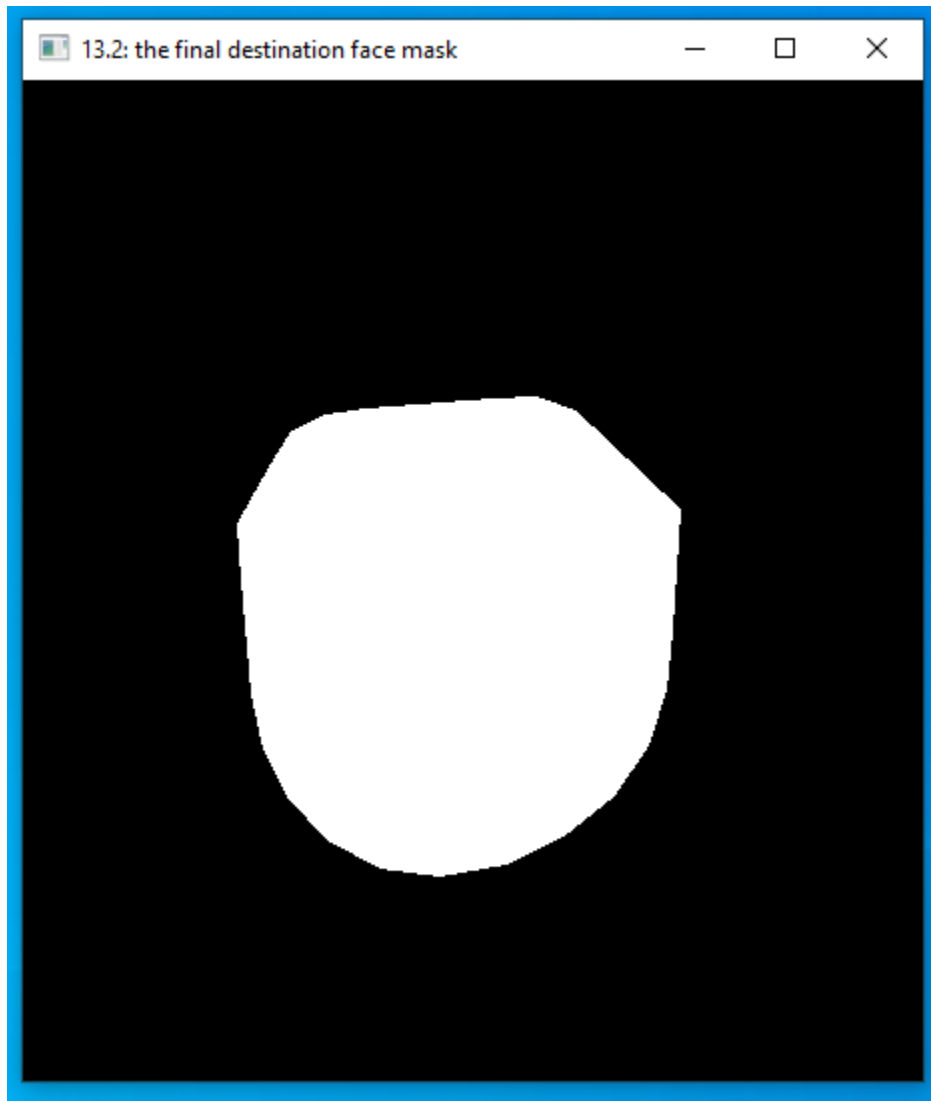
8.1 Face swap for pictures

Swap by Masking the Destination face and placing the newly created face

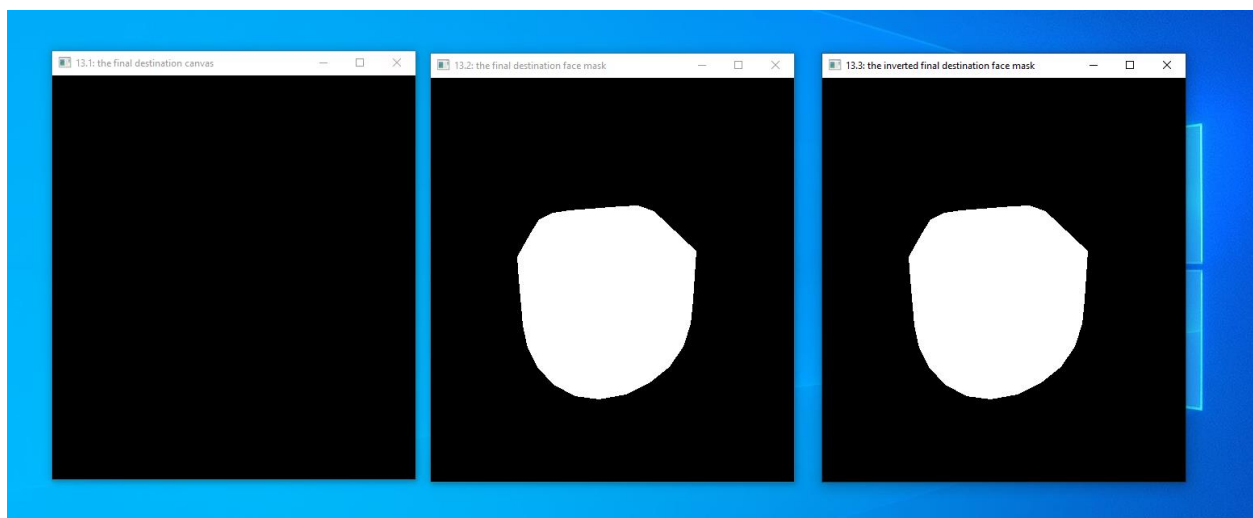
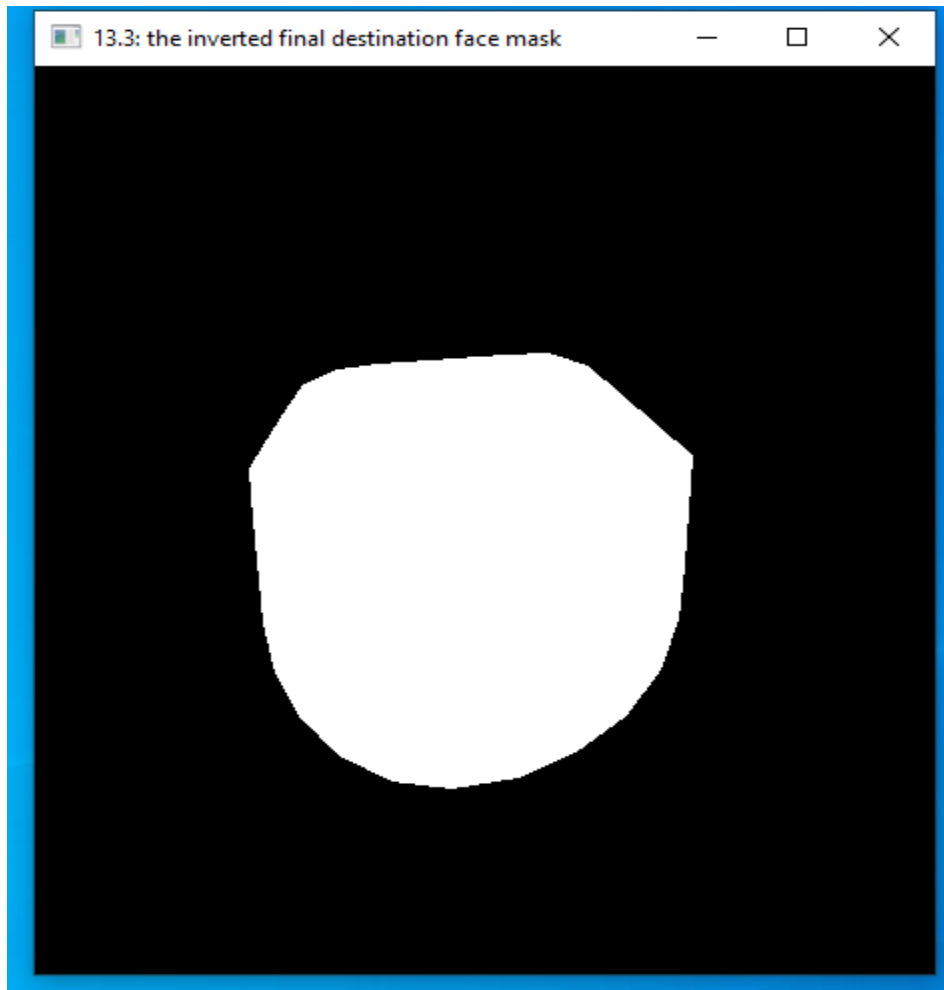
8.1.1 create a new canvas for final image in exactly the same size of destination image



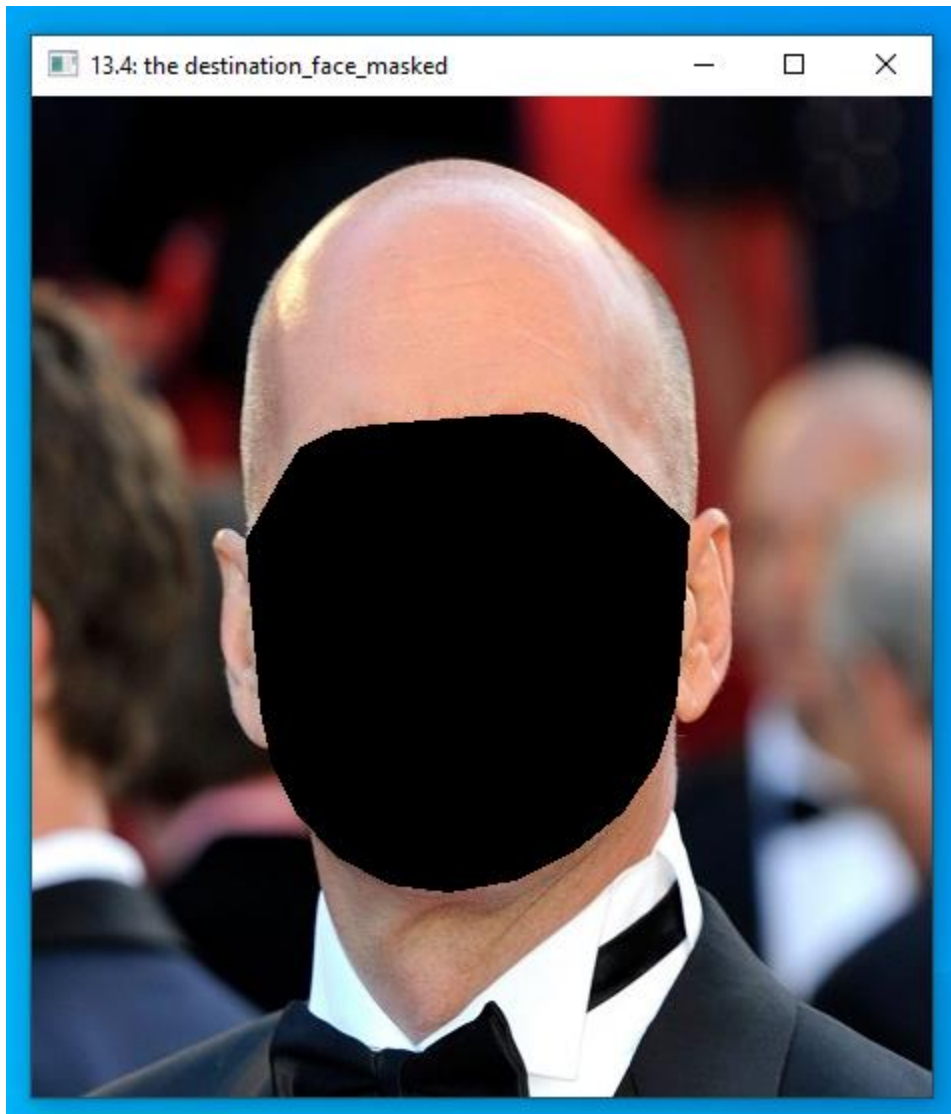
8.1.2 create the destination face mask



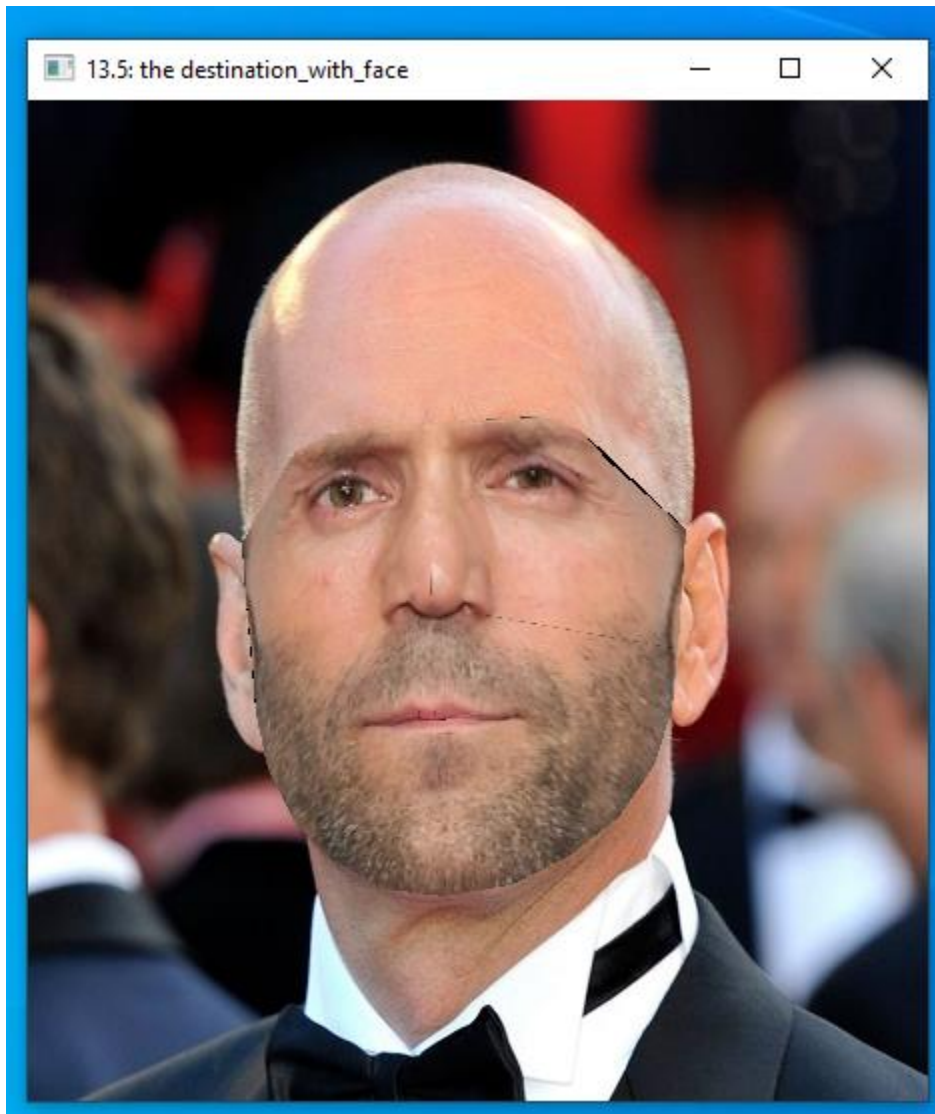
8.1.3 invert the face mask color



8.1.4 mask destination face

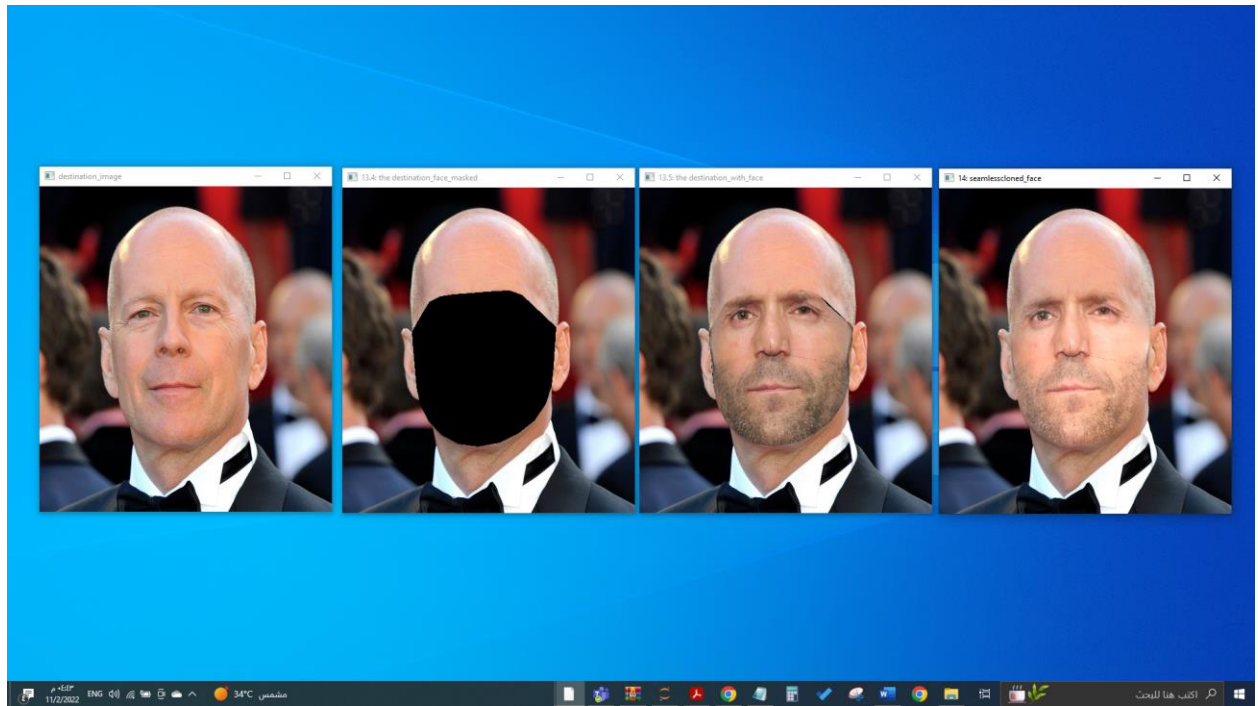


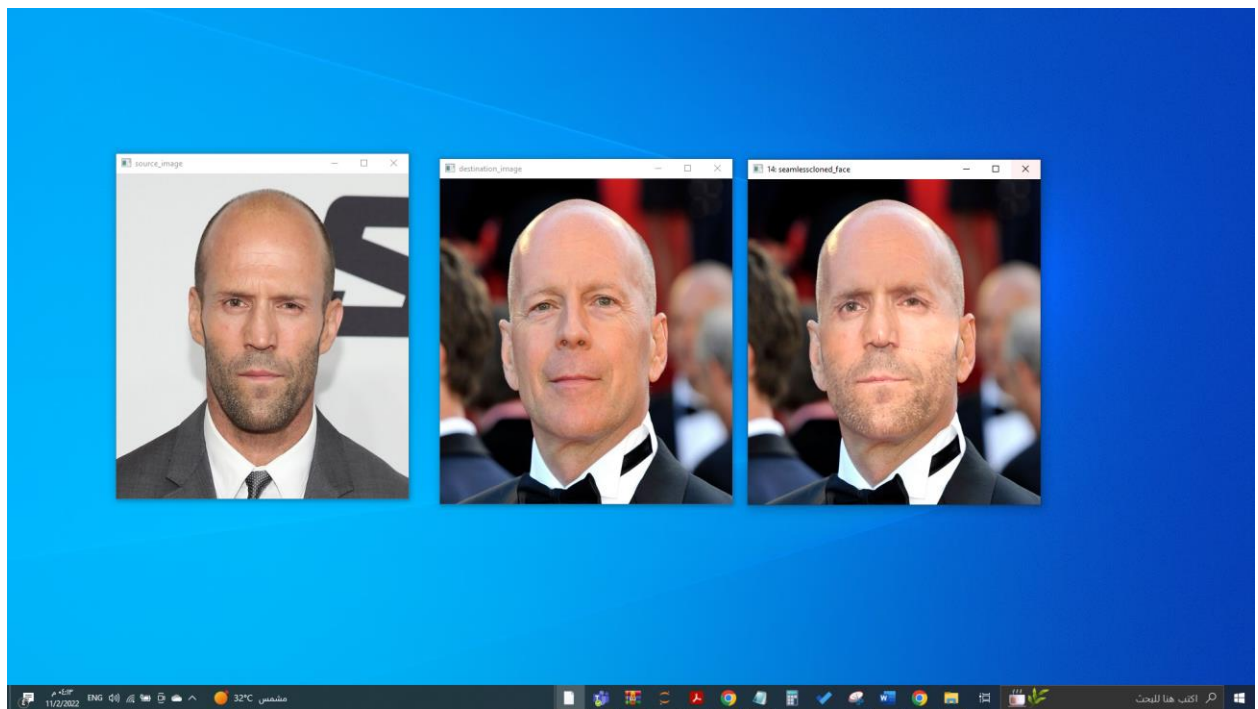
8.1.5 place new face into destination image



8.1.6 Do seamless clone to make the attachment blend with the surrounding pixels



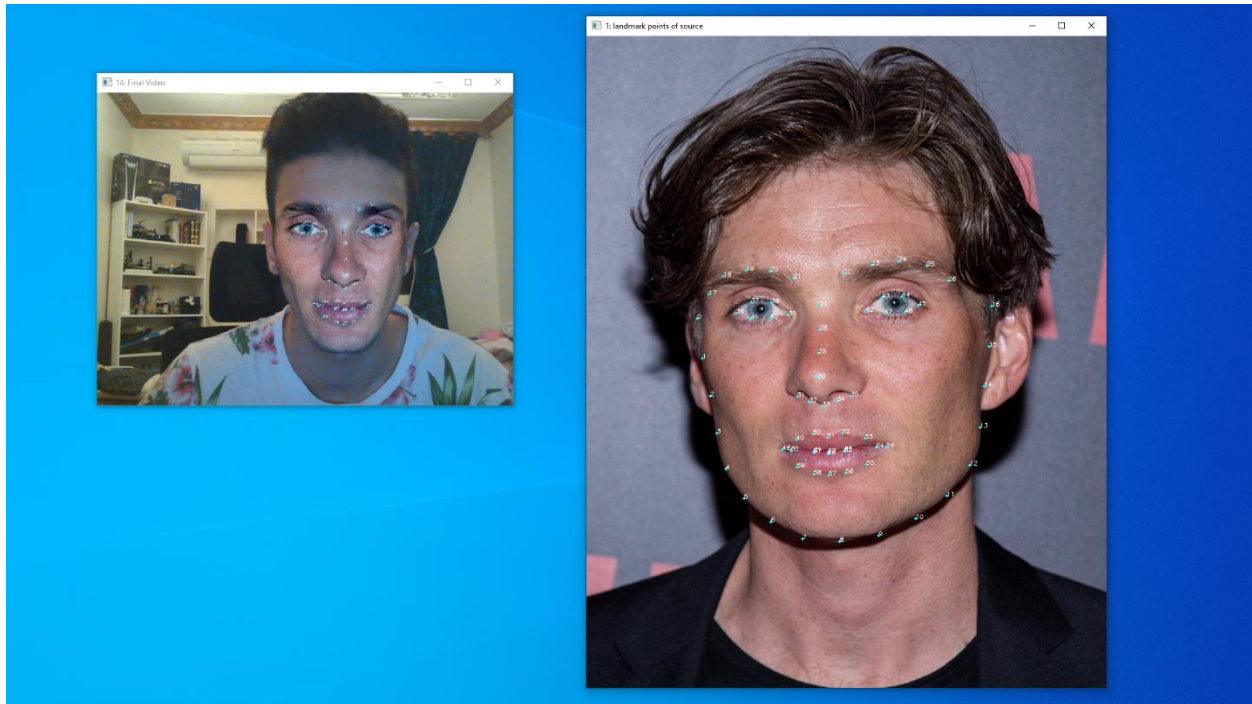




8.2 Face swap in real time with the pc cam

8.2.1 Landmark points of source and destination

We see the loop through the face found in the source image and the predictor takes the human face as input and returns the list of facial landmarks then loops through all the 68 landmark points.



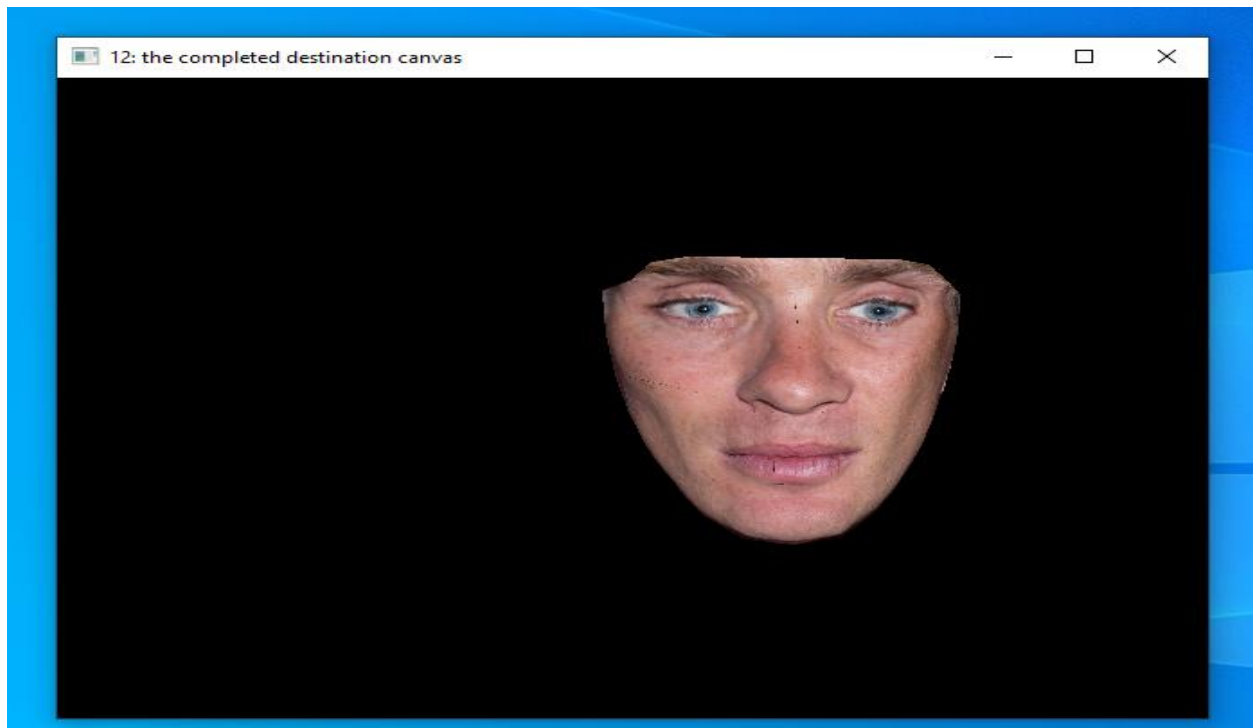
8.2.2 Detector faces in frame check

We see the detector check if there's face in the frame first, he prints a number 1 which means he detect the face, after that I went away from the camera and printed 0 end of the sentence

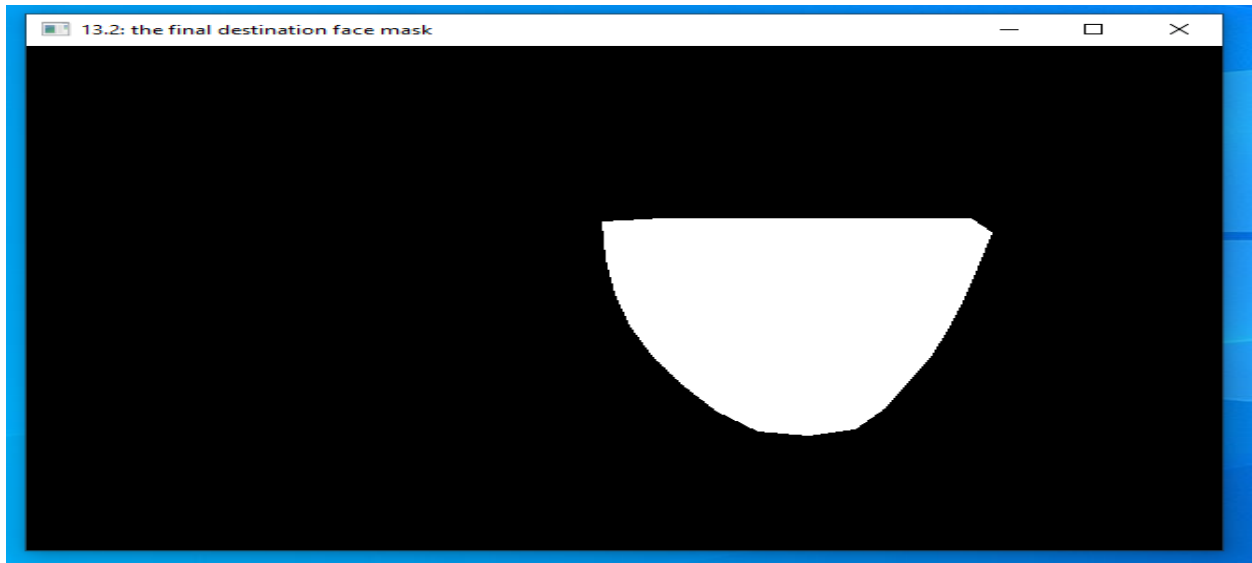
```
The number of faces in video stream: 1
The number of faces in video stream: 1
The number of faces in video stream: 1
The number of faces in video stream: 1
The number of faces in video stream: 1
The number of faces in video stream: 1
The number of faces in video stream: 1
The number of faces in video stream: 1
The number of faces in video stream: 1
The number of faces in video stream: 1
The number of faces in video stream: 1
The number of faces in video stream: 0
The number of faces in video stream: 0
The number of faces in video stream: 0
The number of faces in video stream: 0
```

8.2.3 The completed destination canvas

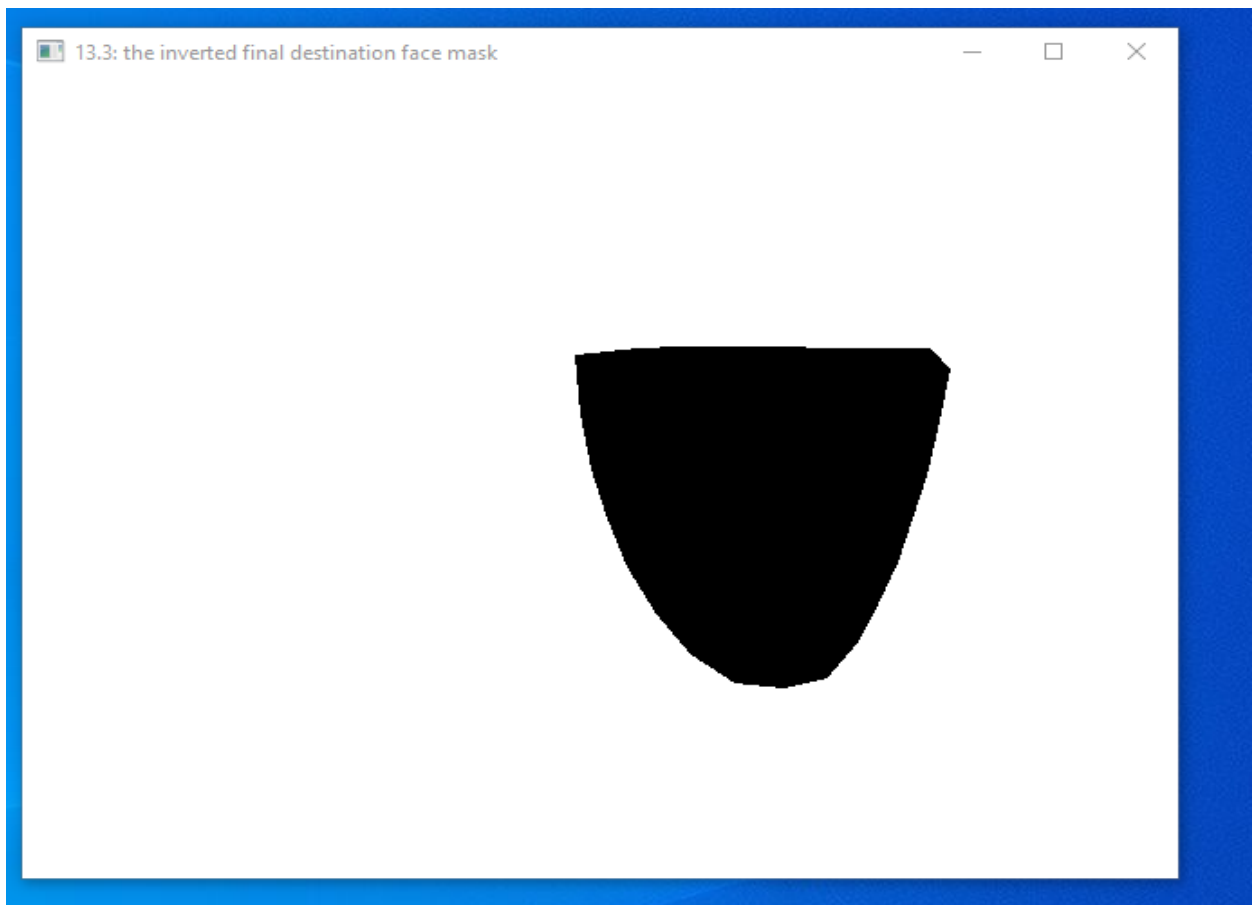
reconstructing destination face in an empty canvas the size of destination image steps to cut off the white lines in the triangle using a mask small rectangular slice of destination canvas in the shape of warped rectangle



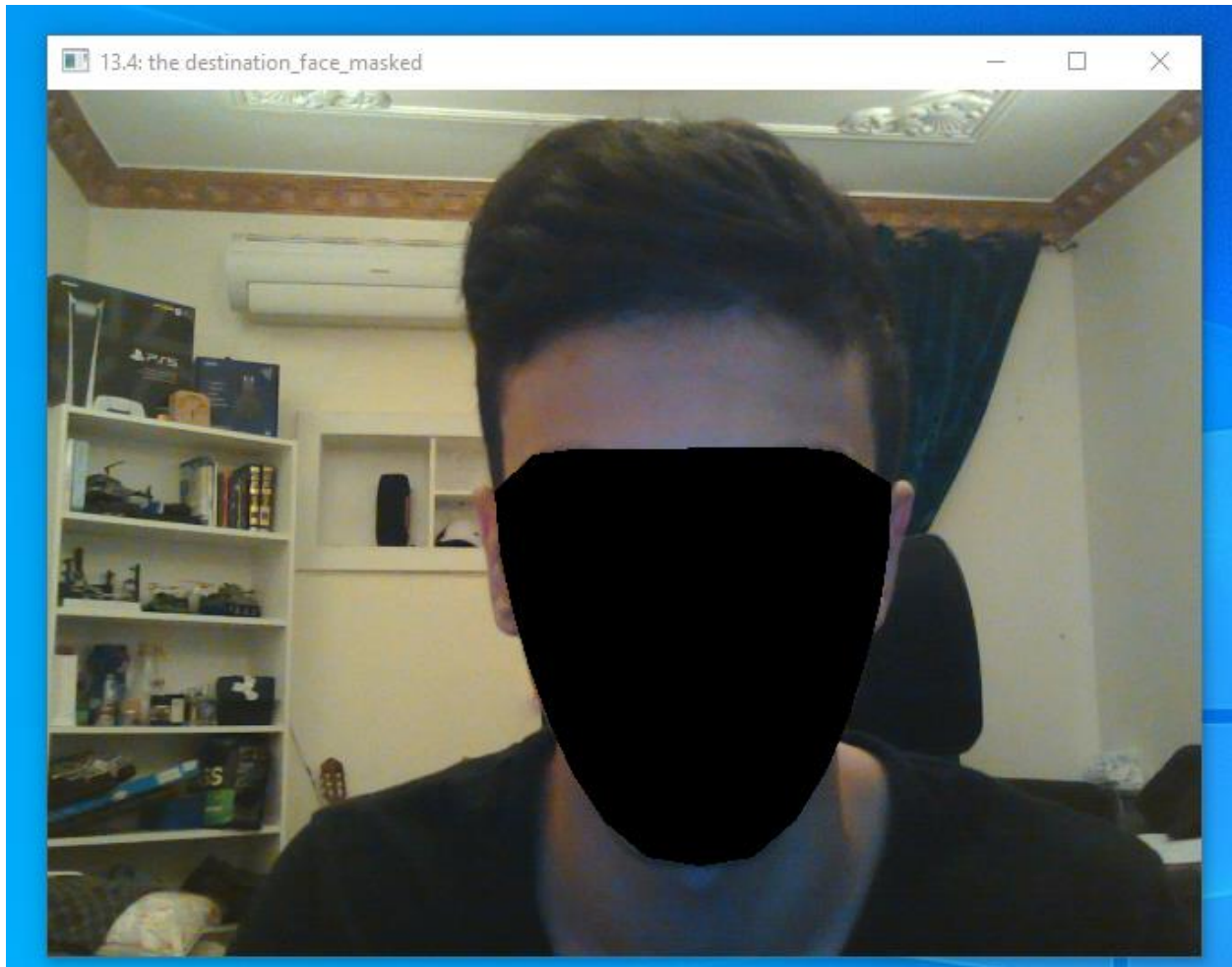
8.2.4 the final destination face mask



8.2.5 the inverted final destination face mask

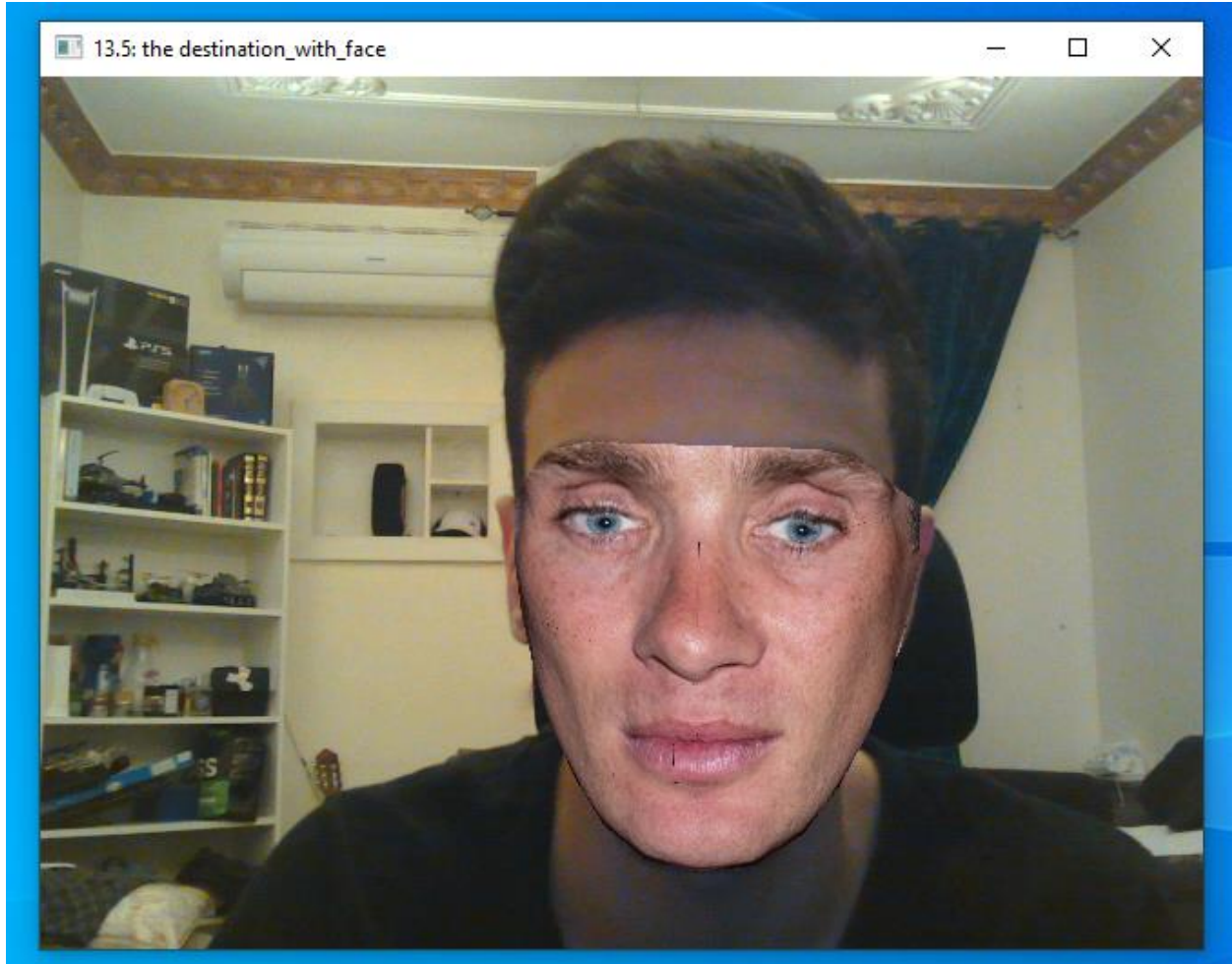


8.2.6 The destination face on webcam masked



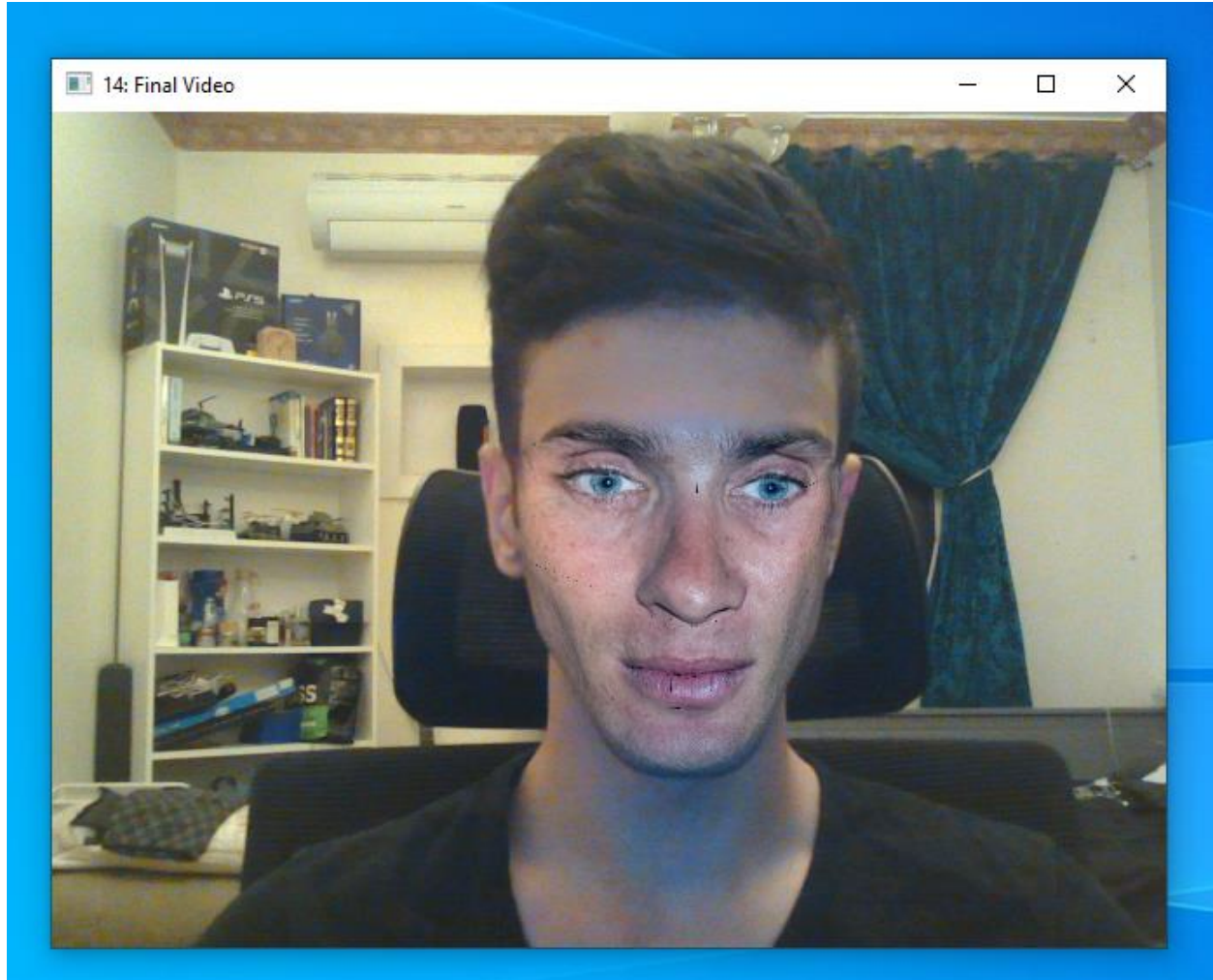
8.2.7 Result before seamless cloning

Putting a new face in the destination image before the seamless cloning process we see the edges of the images look unrealistic



8.2.8 Final result after doing seamless cloning

We see the result in the end after doing the image enhancement and doing a seamless cloning process to look more realistic





8.3 Face Generation Using GAN's

8.3.1 Training epochs and time

I had trained the dataset with 5 epochs and total amount of time taken is around 23 Hours

In [22]:

```
train(5,dataset)|
```

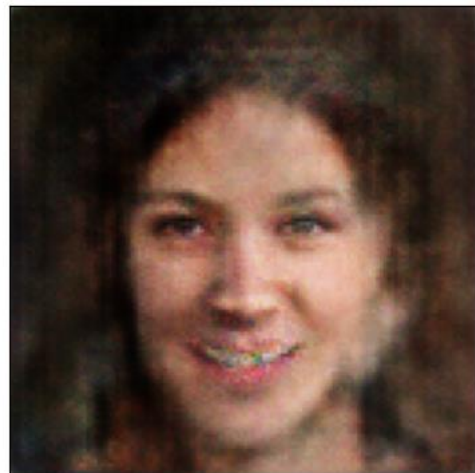
```
Epoch : 1  
Time:16140.0  
Generator Loss: 0.666143536567688 Discriminator Loss: 1.5003554821014404  
  
Epoch : 2  
Time:16880.0  
Generator Loss: 0.7759445309638977 Discriminator Loss: 1.4436471462249756  
  
Epoch : 3  
Time:16402.0  
Generator Loss: 0.752646803855896 Discriminator Loss: 1.1706032752990723  
  
Epoch : 4  
Time:16548.0  
Generator Loss: 0.6490166187286377 Discriminator Loss: 1.165968418121338  
  
Epoch : 5  
Time:16192.0  
Generator Loss: 0.5854569673538208 Discriminator Loss: 1.0999709367752075
```

8.3.2 Plot generated images and final result

```
In [23]: plot_generated_images(1)
```



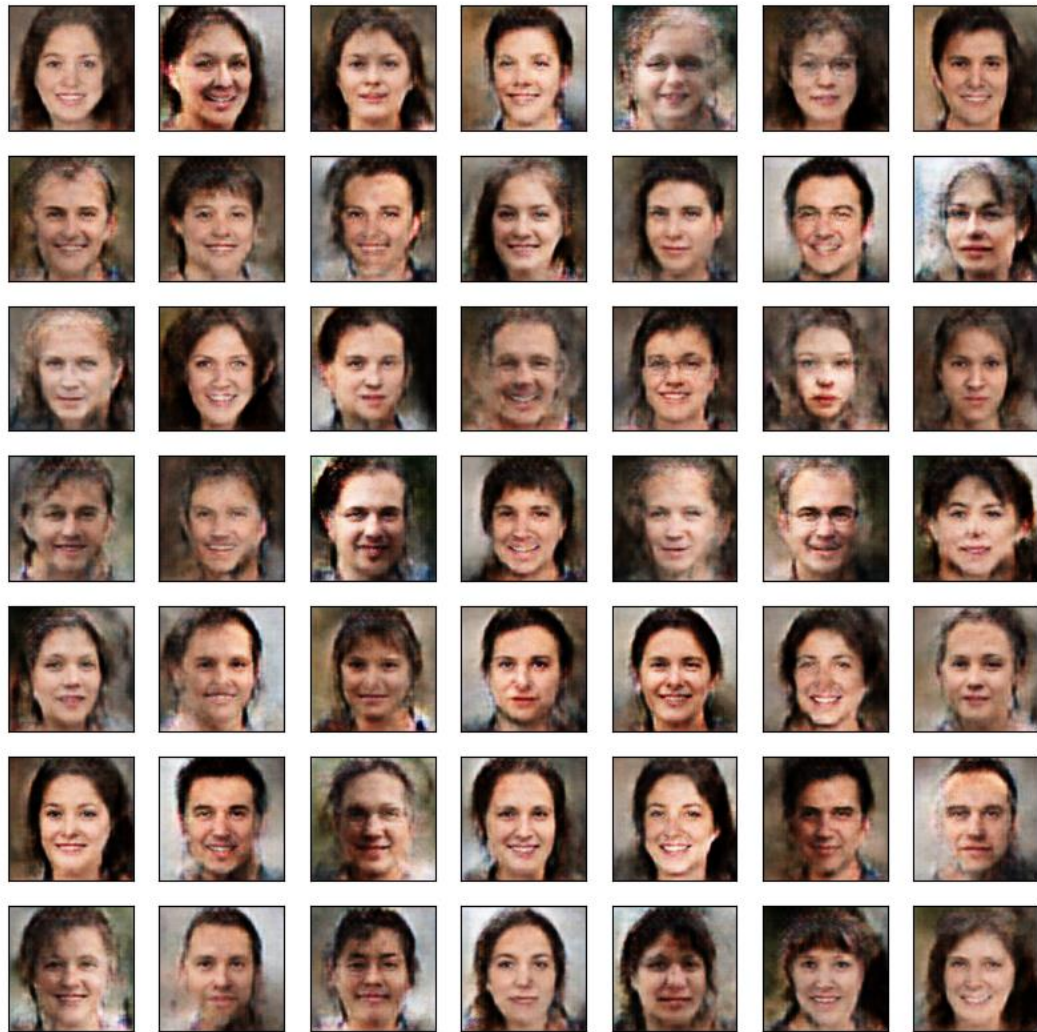
```
In [24]: plot_generated_images(2)
```



```
In [27]: plot_generated_images(5)
```




```
In [26]: plot_generated_images(7)
```



CHAPTER 9: CONCLUSION AND FUTURE WORK

4.5 Conclusion

Our project is all about deepfake technology and the term "deepfake" is a combination of the phrases "deep learning" and "fake," and it refers to artificially created realistic audiovisual content that is usually created using generative adversarial networks (GAN), to make a fake, a branch of machine learning uses neural net simulation on large data sets.

This technology is very modern, and here I'm going to write some of its history, the deep learning was demonstrated as a superior approach to image recognition in a major study published in 2012, and one of the most important components of deepfakes is Generative Adversarial Networks (GANs), which were invented in 2014 by Ian Goodfellow, a PhD graduate who later went on to work for Apple.

In 2017, Reddit users shared footage of celebrity faces being shared with other people's faces and have been very popular.

So, it's a very new technology and it's going to have big uses in the future.

This technology can be implemented in several fields such as:

Educational - deep fake technology has the ability to improve our history courses by engaging with deep samples of historical personalities, preserving stories, and attracting attention.

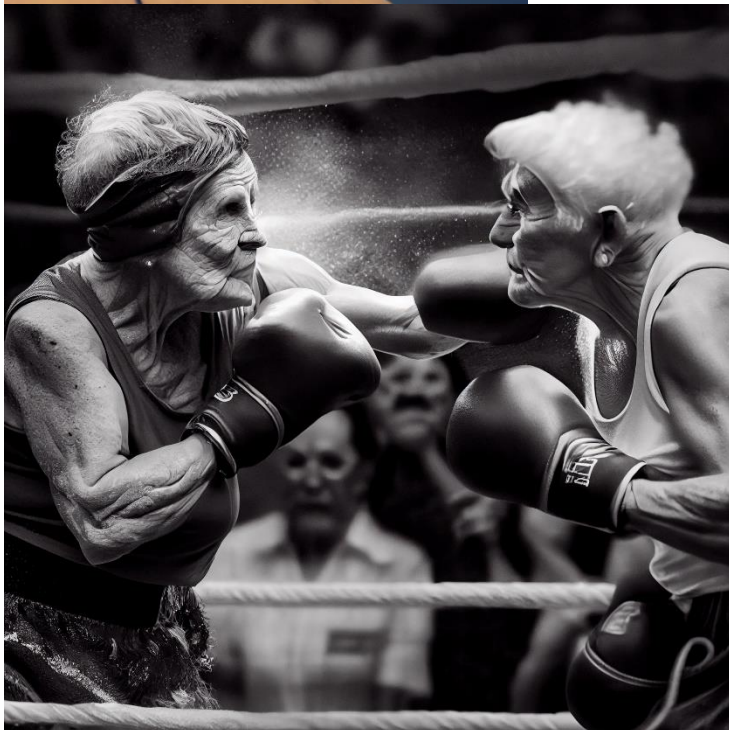
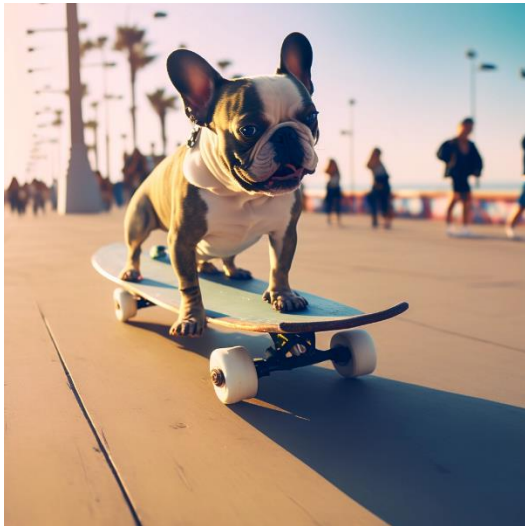
Entertainment Industry - An actor's role can be filled with deep fake technology, for example, the late Peter Cushing's recreation of Star Wars: Rogue One (2017), who died in 1994.

Working on deepfake technology will help to understand machine learning, deep learning and artificial intelligence techniques professionally and will be applied to specialized tools that will be programmed in Python language and require advanced experience so this project will be a complete work of what we have learned within the university.

4.6 Future Work

Deep Fake and Deep Learning are still considered new fields and the pace of improvements and new discoveries is astonishing, some examples of AI that I think will shape the future:

- 1- MidJourney is an AI that can draw original pictures using prompts, So you write any prompt and the AI will try to draw something related, And its drawings are really good to the point that someone used them in a digital drawing contest and won and none of the organizers suspected anything.



- 2- Rytr is an AI that can write for you, you can choose the type of writing for example email or blog and write the subject then it will start writing for you and it has a lot of tools for you to improve the writing

This is a sample from the site I just wrote “Rytr” in the subject field and this is the outcome

“What is Rytr and Who are the Founders?

Rytr is a content marketing platform that was created by the founders of Copyblogger Media.

The platform has been developed to help marketers create, distribute and manage their content in a more efficient way.

Rytr is a powerful content marketing platform that helps marketers create, distribute and manage their content.

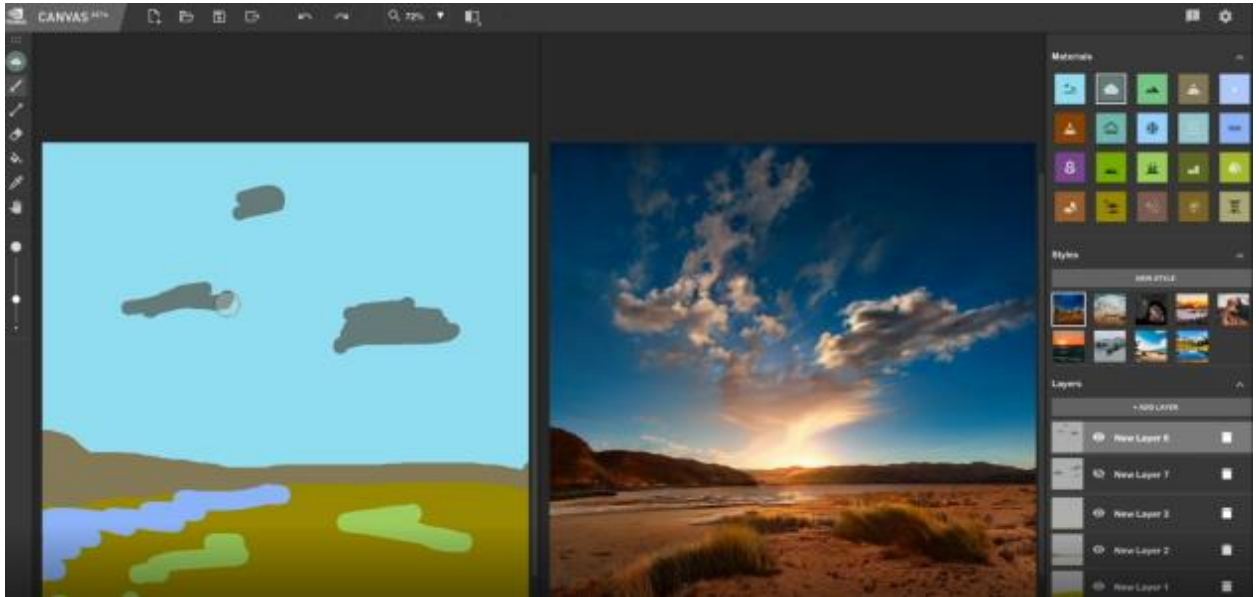
Rytr was founded by the founders of Copyblogger Media - Brian Clark and Sonia Simone.

Brian Clark is an entrepreneur and best-selling author who co-founded Copyblogger Media in 2004.

Sonia Simone is also an entrepreneur who co-founded Copyblogger Media with Brian Clark in 2004.”

As we can see it’s good at writing and with how fast deep learning is advancing, I imagine in the future writing AI will be able to write a complete report while needing even less oversight from humans

- 3- Nvidia canvas is a tool that turns your basic drawing into realistic and beautiful drawings using AI



- 4- Google Duplex is an AI that can make calls on your behalf to schedule appointments and in the demo the AI sounds like a human with no difference at all

Now Just think about when these four types of AI's get considerably better and someone combine them into one deep fake tool that can draw a realistic background and can draw people and write a script and talk with people normally

REFERENCES

- Kubat, M. (n.d.). *An Introduction to Machine Learning* (Third Edition).
- Mishra, P., Varadharajan, V., Tupakula, U., & Pilli, E. S. (2019). A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection. *IEEE Communications Surveys Tutorials*, 21(1), 686–728. <https://doi.org/10.1109/COMST.2018.2847722>
- Mohan, S., Thirumalai, C., & Srivastava, G. (2019). Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques. *IEEE Access*, 7, 81542–81554. <https://doi.org/10.1109/ACCESS.2019.2923707>
- Theobald, O. (n.d.). *Machine Learning For Absolute Beginners* (Third Edition).
- Theobald, O. (2017). *Machine learning for absolute beginners: A plain English introduction*. Scatterplot press.
- Ekman, M. (2021). *Learning deep learning: Theory and practice of neural networks, computer vision, nlp, and transformers using tensorflow* (First edition). Addison-Wesley.
- Glassner, A. S. (2021). *Deep learning: A visual approach*. No Starch Press.
- Mishra, R. K., Reddy, G. Y. S., & Pathak, H. (2021). The Understanding of Deep Learning: A Comprehensive Review. *Mathematical Problems in Engineering*, 2021, 1–15. <https://doi.org/10.1155/2021/5548884>
- Sarker, I. H. (2021). Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Computer Science*, 2(6), 420. <https://doi.org/10.1007/s42979-021-00815-1>
- Sharma, N., Sharma, R., & Jindal, N. (2021). Machine Learning and Deep Learning Applications-A Vision. *Global Transitions Proceedings*, 2(1), 24–28. <https://doi.org/10.1016/j.gltp.2021.01.004>
- Takimoglu, A. (n.d.). *12 Deep Learning Use Cases / Applications in Healthcare in 2022*. <https://research.aimultiple.com/deep-learning-in-healthcare/>
- Vargas, R., Mosavi, A., & Ruiz, L. (2017). DEEP LEARNING: A REVIEW. *Deep Learning*, 10.
- Daly, G. A., Fieldsend, J. E., & Tabor, G. (2022). Variational Autoencoders Without the Variation. *ArXiv:2203.00645 [Cs, Stat]*. <http://arxiv.org/abs/2203.00645>
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11), 139–144. <https://doi.org/10.1145/3422622>
- Hubens, N. (n.d.). *Deep inside: Autoencoders*. <https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f>
- Karnouskos, S. (2020). Artificial Intelligence in Digital Media: The Era of Deepfakes. *IEEE Transactions on Technology and Society*, 1(3), 138–147. <https://doi.org/10.1109/TTS.2020.3001312>
- Karras, T., Laine, S., & Aila, T. (2019). A Style-Based Generator Architecture for Generative Adversarial Networks. *ArXiv:1812.04948 [Cs, Stat]*. <http://arxiv.org/abs/1812.04948>

- Le, T.-N., Nguyen, H. H., Yamagishi, J., & Echizen, I. (2022). Robust Deepfake On Unrestricted Media: Generation And Detection. *ArXiv:2202.06228 [Cs]*. <http://arxiv.org/abs/2202.06228>
- Mirsky, Y., & Lee, W. (2022). The Creation and Detection of Deepfakes: A Survey. *ACM Computing Surveys*, 54(1), 1–41. <https://doi.org/10.1145/3425780>
- Passos, L. A., Jodas, D., da Costa, K. A. P., Júnior, L. A. S., Colombo, D., & Papa, J. P. (2022). A Review of Deep Learning-based Approaches for Deepfake Content Detection. *ArXiv:2202.06095 [Cs]*. <http://arxiv.org/abs/2202.06095>
- Tolosana, R., Vera-Rodriguez, R., Fierrez, J., Morales, A., & Ortega-Garcia, J. (2020). DeepFakes and Beyond: A Survey of Face Manipulation and Fake Detection. *ArXiv:2001.00179 [Cs]*. <http://arxiv.org/abs/2001.00179>
- VON DER BURCHARD, H. (n.d.). *Belgian socialist party circulates “deep fake” Donald Trump video*. <https://www.politico.eu/article/spa-donald-trump-belgium-paris-climate-agreement-belgian-socialist-party-circulates-deep-fake-trump-video/>
- Wu, W., Zhang, Y., Li, C., Qian, C., & Loy, C. C. (2018). ReenactGAN: Learning to Reenact Faces via Boundary Transfer. *ArXiv:1807.11079 [Cs]*. <http://arxiv.org/abs/1807.11079>