

Meaningful Encryption of The Real Images

A Project Report submitted by

Raj Kumar

M21AI570

in partial fulfillment of the requirements for the award of the degree of

M.Tech



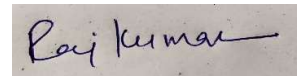
॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

Indian Institute of Technology Jodhpur
School of Artificial Intelligence and Data Science

July 2023

Declaration

I hereby declare that the work presented in this Project Report titled “**Meaningful encryption of the real images** ” submitted to the Indian Institute of Technology Jodhpur in partial fulfillment of the requirements for the award of the degree of M.Tech., is a bonafide record of the research work carried out under the supervision of **Dr. Gaurav Bhatnagar**. The contents of this Project Report in full or in parts, have not been submitted to, and will not be submitted by me to, any other Institute or University in India or abroad for the award of any degree or diploma.



Signature

Raj Kumar

M21AI570

Certificate

This is to certify that the Project Report titled “**Meaningful encryption of the real images**”, submitted by **Raj Kumar (M21AI570)** to the Indian Institute of Technology Jodhpur for the award of the degree of M.Tech., is a bonafide record of the research work done by him under my supervision. To the best of my knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.



Signature

Dr. Gaurav Bhatnagar

Abstract

This project report presents an innovative approach to image encryption using Deep Convolutional Generative Adversarial Networks (DCGANs). Our proposed encryption method provides a novel solution to the ever-increasing need for robust image data security.

The DCGAN model consists of a generator and a discriminator, working in tandem to iteratively produce increasingly realistic encrypted images. The generator starts with random noise and utilizes transpose convolutional layers to up-sample the image, aiming to produce outputs that convincingly deceive the discriminator. The discriminator, in turn, employs a neural network to distinguish between real and generated samples, continually updating its weights via an optimizer and a loss function.

The methodology utilizes a celebrity face attributes dataset, which provides an extensive range of face images for training the GAN. The encryption process begins with the generation of random noise, followed by a series of transpose convolutional layers that up-sample the image.

We used the comprehensive celeb faces attribute dataset to train our model, providing a rich diversity of facial images. After 2809 epochs, we observed significant improvements in the generator's ability to create plausible encrypted fake samples. This report underscores the vast potential of DCGANs in creating secure image encryption protocols and invites further exploration into optimizing this approach for more efficient and robust image data security.

Contents

Page

<i>Abstract</i>	1
Introduction and Background	3
Literature Survey on Image Encryption & Via GAN	6
Problem Statement and Objective	9
Motivation Behind GAN while Encrypting Images:	10
Work:	16
Learning Steps from Program / Code & their Outcome	23
Experimental result	27
Conclusion	28
<i>References</i>	29

1. Introduction and Background

Encryption is the process of transforming information so that only a designated recipient can understand it. In the case of images, encryption can be used to protect the content from unauthorized access. There are different methods for encrypting images, including classical encryption algorithms like AES, or more complex techniques using neural networks such as GANs.

Generative Adversarial Networks (GANs) are a class of machine learning models designed to generate new data instances that resemble a given set of training data. GANs consist of two networks, the generator, and the discriminator, that are trained together through a competitive process.

So, can say, basically, encryption is a crucial element of modern digital communication, ensuring the security and confidentiality of transmitted data. With the advent of deep learning and the sophistication of artificial intelligence (AI) models, the process of encryption has seen major advancements, especially in the field of image encryption.

Traditional image encryption techniques like Advanced Encryption Standard (AES), Data Encryption Standard (DES), and various chaos-based methods have significantly protected data. However, they sometimes fail to fully secure the information from modern hacking techniques and are computationally expensive with large data like high-resolution images.

As a solution to these issues, the application of deep learning models, specifically Generative Adversarial Networks (GANs), has emerged in the field of image encryption. GANs, proposed by Ian Goodfellow and his team in 2014, consist of The Generator and the Discriminator. The Generator tries to create data that is similar to the input data distribution, while the Discriminator attempts to distinguish between the real data and the data created by the Generator.

Image encryption is used to secure image data by transforming the original image into an unreadable format. It can be performed using various algorithms, like:

- **Symmetric Key Encryption:** Using the same key for both encryption and decryption.
- **Asymmetric Key Encryption:** Using a pair of keys, one for encryption and the other for decryption.
- **Chaos-based Encryption:** Utilizing chaotic systems in the encryption process.

Using GANs for image encryption is a relatively new approach that leverages the power of deep learning. Here's a general overview of how it can be done:

- **Training a GAN:** The generator is trained to create images that are similar to the real images, while the discriminator tries to distinguish between real and generated images.
- **Encryption Process:** The real image is fed into the generator, which then produces an encrypted (or transformed) version of the image. This encrypted image is designed to be incomprehensible without the correct decryption process.
- **Decryption Process:** A corresponding decryption model (which could be another neural network) is designed to take the encrypted image and revert it to its original form.

Using GANs for encryption offers the potential for highly complex and unique encryption patterns that are difficult to break without the correct key. However, it also presents challenges such as:

- **Model Complexity:** GANs are complex models that require careful tuning and extensive training.
- **Key Management:** Handling the encryption/decryption keys or model parameters securely can be challenging.
- **Efficiency:** Depending on the implementation, this approach might be computationally intensive, making it unsuitable for real-time applications.

The potential of GANs in image encryption lies in their ability to generate and learn complex data distributions, which enables them to create complex encrypted images that are challenging to decipher without the correct decryption key.

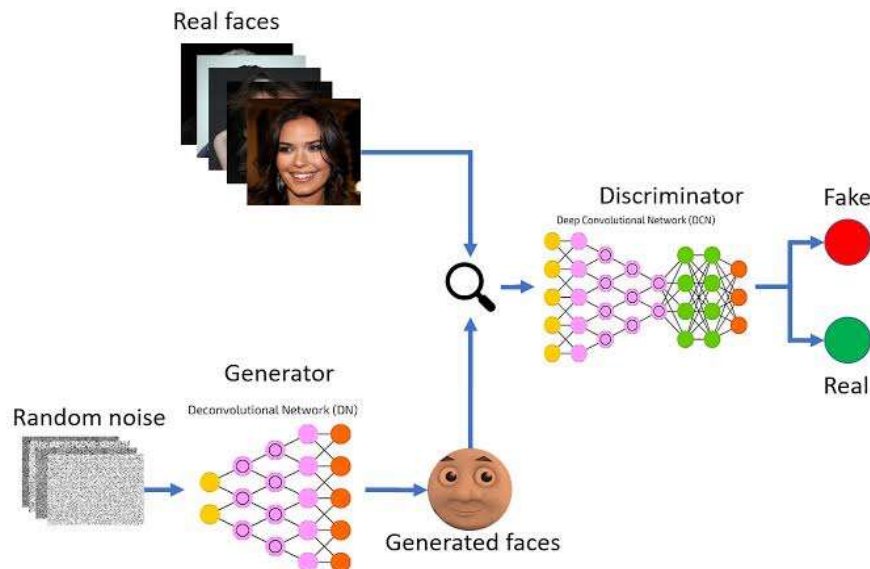


Figure 1: - The Generative Adversarial Network.

So, can say, encrypting images using GANs presents a novel approach to secure image data, utilizing the complex patterns and deep learning capabilities of neural networks. While promising, this technique is still in the experimental phase and requires further research and development to be practically applicable for broader use. It opens up new possibilities in the field of secure communications and data protection, blending the disciplines of cryptography and machine learning. Image encryption using GANs adds another layer of security to the image encryption process. It makes it difficult for adversaries to extract any meaningful information without access to the original GAN model, thus providing a robust solution for image encryption.

2. Literature Survey on Image Encryption & Via GAN

Conducting a literature survey on the subject of image encryption and the use of Generative Adversarial Networks (GANs) provides a comprehensive view of the existing research, methodologies, applications, and challenges in this field. Here's a summary of what the literature survey might cover:

1. **Traditional Image Encryption Methods:** - Researchers have explored different ways to hide or encrypt images. One method involves using secret codes, like AES or RSA. In this approach, they've compared two main strategies. The first uses the same code to both hide and reveal the image, called symmetric encryption. The second uses two different codes, known as asymmetric encryption, one to hide the image and the other to reveal it. Additionally, some researchers have experimented with using chaos theory to hide images. This method employs unpredictable behavior, known as chaos, to encrypt the image, and they've examined both the benefits and potential problems with this approach.

Key Points

a. Symmetric and Asymmetric Key Algorithms

- Research on classical cryptographic methods like AES, DES, RSA, etc., for image encryption.
- Comparison of symmetric versus asymmetric key algorithms.

b. Chaos-based Image Encryption

- Exploration of chaos theory-based encryption methods.
- Challenges and benefits of using chaotic systems.

2. **Image Encryption using Neural Networks:** - The early work in hiding images used simple models that mimic the brain's function, known as neural networks. Researchers looked at what's good and bad about these simple neural network models for hiding images. Later, more complex brain-like models, known as deep learning models, were used to hide images. People compared these more complicated methods with traditional ways of hiding images to understand their advantages and limitations.

Key Points

a. Use of Basic Neural Networks

- Early works on using neural networks for image encryption.
- Advantages and limitations of basic neural network-based encryption.

b. Deep Learning Approaches

- Implementation of deep learning architectures for image encryption.
- Comparison with traditional methods.

- 3. GAN-based Image Encryption:** - GANs, or Generative Adversarial Networks, is a special type of model used in various ways, like creating new images. Researchers have explained how GANs work and explored their potential. Some studies specifically used GANs to hide images. They explored different ways to do this and examined possible applications, such as keeping messages private. Along with this, researchers also looked at potential problems and weaknesses with using GANs, like how hard they might be to use or any security issues they might have.

Key Points

a. Basic Principles of GANs

- Introduction to GANs and their working mechanism.
- Use of GANs in various applications, including data generation, image-to-image translation, etc.

b. GANs for Image Encryption

- Specific research papers focused on utilizing GANs for encrypting images.
- Different architectures and methods to perform the encryption/decryption process.
- Possible applications in secure communications, digital rights management, etc.

c. Challenges and Limitations

- Exploration of challenges related to the complexity, efficiency, and security of GAN-based image encryption.
- Studies on potential weaknesses and vulnerabilities.

- 4. Future Directions and Emerging Technologies:** - Looking forward, researchers are working on new methods or innovative uses of GANs to hide images. They're also considering how to combine GANs with other secret code principles to possibly improve the process. Ethical and societal considerations are also being thought about in this field. Researchers are reflecting on what's right and wrong with these methods and how they might affect people in broader societal contexts.

Key Points

a. New Algorithms and Techniques

- Upcoming methodologies or innovative applications of GANs in image encryption.
- Integration with other cryptographic principles.

b. Ethical and Societal Considerations

- Consideration of ethical aspects and potential societal impacts.

By reviewing the existing literature on image encryption and the specific use of GANs, one can gain an in-depth understanding of the current state of the field, emerging trends, and potential avenues for future research and development. It's crucial to rely on peer-reviewed journals, conferences, and reputable academic sources to ensure the credibility and validity of the information gathered in the survey.

The content concludes with a wrap-up of the main points, identifying what's still unknown and what might be explored next in this field. It encapsulates various ways people have tried to hide images, focusing particularly on the use of a complex model called GANs, and anticipates future directions in this exciting area of study.

3. Problem Statement and Objective

The *problem statement* for this project is centered around the use of Generative Adversarial Networks (GANs) for the generation of artificial or 'encrypted' faces utilizing a provided 'celebrity image' dataset. The primary objective is to develop a GAN model that is not just capable of creating synthetic faces, but also optimizing it to a level where the generated faces are highly realistic and high-resolution, potentially making them indistinguishable from actual human faces.

This task poses a significant challenge due to the intricate complexity of human facial features, their high variability across different individuals, and the need for a high degree of realism in the generated images. Successfully achieving this will involve a deep understanding of GAN architecture, effective training methodologies, and efficient optimization strategies.

In essence, this project seeks to advance the field of image synthesis using GANs by creating a model that can convincingly replicate human faces. Through this, it hopes to enhance our understanding of the capabilities and potential applications of GANs in image encryption and other related domains.

4. Motivation Behind GAN while Encrypting Images:

The motivation behind this task can be multi-fold: -

1. **Advancing AI Research:** GANs are a powerful tool in AI and have been instrumental in pushing the boundaries of what's possible, especially in the field of computer vision. By working on this task, you're contributing to the research and development in this area.
2. **Image Synthesis Applications:** The ability to generate realistic synthetic images has numerous practical applications, including but not limited to, entertainment (e.g., video games, movies), virtual reality, advertising, and even forensic science.
3. **Improving Understanding of GANs:** GANs are complex models and working on this task allows for a deeper understanding of these models, their limitations, their capabilities, and how to improve them.
4. **Data Privacy and Ethical Uses:** By generating synthetic faces, we can create large datasets for AI training without violating individual privacy rights. These faces can be used in place of real faces where sharing or using real data might raise privacy concerns.
5. **Future Readiness:** As AI continues to advance, there will be growing demand for professionals who are capable of designing, implementing, and managing GANs. Therefore, getting hands-on experience with these models now prepares me for potential future opportunities in the AI field.
6. **Innovation and Creativity:** Lastly, there is an intrinsic motivation that comes from creating something new. Generating convincing fake faces from GANs could lead to new, unforeseen applications, sparking innovation and creativity.

So, can say, the task of generating synthetic faces using GANs offers a significant contribution to AI research and provides an opportunity to advance understanding of these complex models. This capability has extensive applications in various sectors like entertainment, virtual reality, and data privacy. Gaining hands-on experience prepares me for future career opportunities in the continually evolving AI field. Additionally, this task could lead to novel applications, fostering innovation and creativity.

4.1 Advantages of Generative Adversarial Networks GANs:

- GANs can generate high-quality synthetic data that is similar to the training data. This can be useful for tasks such as image generation or language translation, where it is difficult or expensive to obtain a large amount of real training data.
- GANs can learn to capture complex patterns and features in the training data, allowing them to generate highly realistic synthetic data.

- GANs can be trained on a small amount of data and still generate good results.
- GANs can be used to augment the training data for a supervised learning task, potentially improving the performance of the model.
- GANs can be used to perform unsupervised learning, which can be useful in situations where labeled training data is not available.

4.2 Disadvantages of Generative Adversarial Networks (GANs)

- Training GANs can be difficult, as the training process involves two neural networks competing with each other. This can make it hard to balance the training of the two networks, and it can be difficult to get them to converge.
- GANs can be sensitive to the choice of hyperparameters, and it can be difficult to find the right combination of hyperparameters to get good results.
- GANs can be prone to mode collapse, where the generator produces a limited range of outputs, rather than a diverse range of outputs.
- GANs can require a lot of computational resources, as they involve training two large neural networks.
- GANs are not well suited to tasks that require deterministic output, as the output of a GAN is probabilistic.
- GANs can be difficult to debug, as it can be hard to understand why they are not producing the desired output.

4.3 GAN Applications, Their Types & Use Cases

Generative Adversarial Networks (GANs) are a powerful class of machine learning models that are capable of generating synthetic data that is difficult to distinguish from real data. Generative Adversarial Networks (GANs) are a type of artificial intelligence algorithm used to generate new, previously unseen data that is similar to a training dataset. They have been used for a wide range of applications, including generating images, audio, and text. They have been used for a wide range of applications, including:

4.3.1. Applications Using GAN

- **Image generation:** GANs have been used to generate realistic images of faces, landscapes, and other objects.
- **Text generation:** GANs have been used to generate coherent and natural-sounding text, including news articles and stories.

- **Music generation:** GANs have been used to generate music in various styles and formats.
- **Video generation:** GANs have been used to generate synthetic videos that are difficult to distinguish from real videos.
- **Domain adaptation:** GANs can be used to transfer the characteristics of one domain to another, allowing machine learning models to perform better on tasks in a new domain.
- **Data privacy:** GANs can be used to generate synthetic versions of sensitive datasets that can be used for machine learning tasks while protecting the privacy of the original data.
- **Generating realistic images:** GANs have been used to generate realistic images of people, animals, landscapes, and other objects. They have also been used to create images of non-existent objects, such as hybrid animals or fantastical landscapes.
- **Improving image quality:** GANs have been used to improve the resolution of images, by generating higher-resolution versions of low-resolution images.
- **Data augmentation:** GANs can be used to generate additional training data for other machine learning models, allowing them to perform better on tasks such as image classification or object detection.
- **Medical image analysis:** GANs have been used in medical image analysis to generate synthetic medical images for training and testing other machine learning models.

4.3.2. Types of GAN Network

- Vanilla GAN:** - The original and foundational GAN model consists of a generator and a discriminator. The generator tries to create fake data, while the discriminator attempts to distinguish between real and fake data.

Uses: - Basic image generation, Conceptual understanding, and educational purposes.

Examples: - Creating simple digital images, like generating numbers similar to the MNIST dataset.

- Deep Convolutional GAN (DCGAN):** - DCGAN introduces convolutional layers, specifically designed for image data, into the GAN structure. This provides more detailed and structured image outputs.

Uses: - Image generation with more clarity, Style transfer, and Image-to-image translation.

Examples: - Generating faces of non-existent people, Enhancing the resolution of pixelated images.

- Conditional GAN (cGAN):** - cGANs allow for directed data generation by conditioning the generation process on some external information, like labels.

- Uses:** - Directed image generation, Targeted data augmentation, and Image-to-image translation based on specific conditions.
- Examples:** - Generating a particular type of clothing, Turning specific segmentation maps into realistic images.
- d. CycleGAN:** - CycleGAN can translate images from one domain to another without the need for paired data, using cycle-consistency loss to ensure that the transformation is accurate and reversible.
- Uses:** - Style transfer for unpaired datasets, Image-to-image translations without examples.
- Examples:** - Converting photos of apples to oranges, Converting paintings to photos.
- e. StackGAN:** - In StackGAN, the generation process happens in stages. The first stage captures basic shapes and colors, and the second refines the details.
- Uses:** - Image generation from textual descriptions, Multi-stage image refinement
- Examples:** - Generating an image of a bird described as "having blue wings and a red belly".
- f. Wasserstein GAN (WGAN):** - WGAN modifies the traditional GAN loss function with the Wasserstein distance, improving the stability of training and addressing mode collapse.
- Uses:** - Stable GAN training, Reducing mode collapse, More realistic image generation.
- Examples:** - Generating diverse and realistic facial structures without repeated or collapsed outputs.
- g. Progressive GAN:** - **Starts** by generating a low-resolution image, and progressively increases the detail and resolution in stages.
- Uses:** - Producing extremely high-resolution images, Gradual refinement of details.
- Examples:** Creating detailed 1024x1024 images of landscapes, animals, or faces.
- h. BigGAN:** - BigGAN scales up the size of the GAN architecture, using more parameters and producing detailed images.
- Uses:** - State-of-the-art image generation.
- Examples:** -Detailed images of diverse categories such as animals, scenes, or objects.
- i. StyleGAN & StyleGAN2:** = Introduces a style-based generator architecture where variations are injected at multiple layers, providing fine control over image details.
- Uses:** - Extremely high-quality image generation, Specific feature manipulation in images, Morphing and merging styles between different images.

Examples: - Photorealistic, high-resolution faces, Combining features of two faces to produce a synthetic third face.

4.4 DCGAN: - Deep Convolutional Generative Adversarial Network

Certainly! DCGAN, or Deep Convolutional Generative Adversarial Network, is a particular variant of GAN that has become highly influential in the field of machine learning. DCGAN is a model composed of two deep neural networks that are trained together in a kind of game, often referred to as a "cat-and-mouse" game. One network, the generator, tries to create data (like images) that are similar to some real data. The other network, the discriminator, tries to tell the difference between real data and the fake data created by the generator. What sets DCGAN apart from other GANs are its specific architectural choices, which make the training of the networks more stable and efficient.

Deep Convolutional GAN (DCGAN) is not universally "better" than all other GANs, but it does introduce several advancements over the basic Vanilla GAN, making it particularly suitable for image generation tasks. Here's why DCGAN is often preferred for specific applications:

- **Use of Convolutional Layers:** Both the generator and discriminator make extensive use of convolutional layers. These are the same kinds of layers used in traditional convolutional neural networks (CNNs), which are powerful tools for image recognition tasks.
- **Batch Normalization:** DCGAN introduces batch normalization into both networks. This helps in stabilizing training by normalizing the input layer by adjusting and scaling the activations.
- **Elimination of Fully Connected Layers:** In traditional GANs, there are often fully connected or dense layers. DCGAN architecture tends to avoid these, relying more on convolutional layers.
- **Activation Functions:** DCGAN uses specific activation functions, such as the ReLU activation function in the generator and the LeakyReLU activation function in the discriminator, to help with the training process.
- **Image Generation:** One of the most notable applications of DCGAN is in generating images that look like real photographs of human faces, animals, or other objects. They've been used in art, design, and various entertainment fields.

DCGANs represent a significant step in the evolution of GANs, and their architecture has influenced many subsequent GAN models. They have helped in overcoming some of the training challenges associated with earlier GANs, leading to more widespread use and experimentation with generative models. In many ways, DCGAN has become a foundational architecture for those looking to explore and innovate within the world of generative deep learning.

However, while DCGANs offer numerous advantages, they are not always the best choice for every application. Other variants of GANs, such as WGAN, CycleGAN, or StyleGAN, introduce their own set of innovations that address other challenges in the GAN training process or are tailored to specific applications. The choice of GAN often depends on the specific problem at hand and the challenges associated with it.

5. Work:

This report explains how to encrypt an Image using a Deep Convolutional Generative Adversarial Network (DCGAN). The GAN has a generator and discriminator. The generator generates fake samples and the discriminator determines if a sample is fake or real. Initially, the generator produces random outputs the discriminator identifies as fake. Over time, the generator improves and produces more realistic results.

We aim to utilize the DC GAN network for implementation. There are various types of GAN networks, such as Style GAN, Cycle GAN, and DC GAN. Each of these GAN networks has the basic component of a generator and a discriminator.

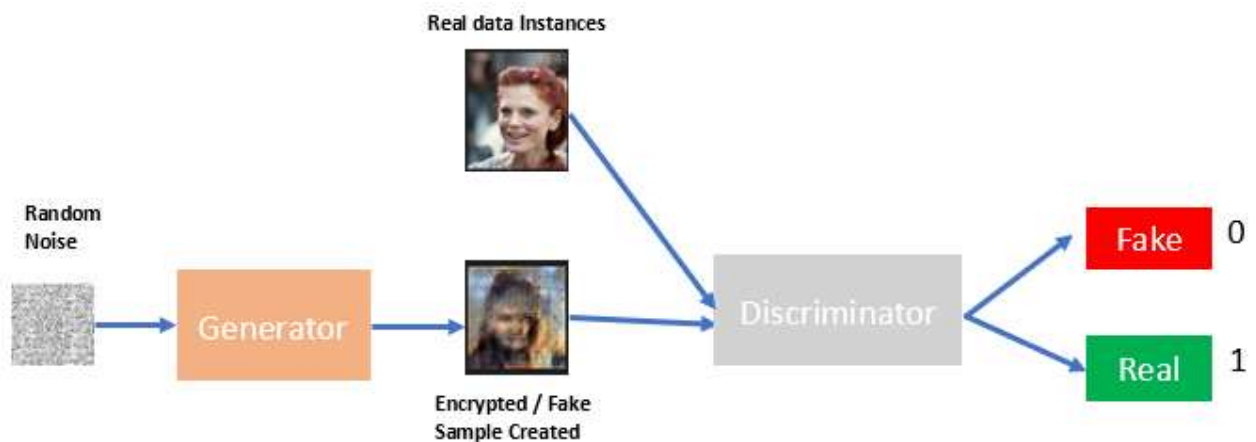


Figure 2: - The Generative Adversarial Network with the Real Data Set

- **Generator** is responsible to generate fake samples. So, in the above diagram, the fake sample is generated after 2809 epochs. and Real data instances are one of the samples from our real data set. The generator code starts with random noise reshaped and passed through transpose convolutional layers to up-sample the image. The goal is to produce realistic encrypted fake samples fooling the discriminator.
- The **discriminator** code has a neural network taking real and fake samples. It classifies them as real (1) or fake (0). An optimizer and loss function update the discriminator's weights.

Initially, when we start the training, initially generator will take some random sample and produce some random output only and this discriminator can easily find out that this is a fake sample because that will be a random noise only. But over time, our generator will start giving us more realistic results, after a few epochs, Let's say after 1000 epochs or 1500 epochs or 2500 epochs mean after over time, the generator will start giving us more realistic results. By getting the feedback from Discriminator, after every pass, the discriminator will give feedback to the generator, and based on that, the generator will improve. In the same

way, over time, the generator will also improve, and over time, and discriminator will also improve in the results by comparing the generated data and the real data.

The dataset (**celeb faces attribute**) is used to train this model and contains celebrity face images. CSV files provide information like *face landmarks*, *bounding boxes*, and *attributes* for the images. The model trains for **2809** epochs. Generated fake face images at different epochs show the generator improving. Viewers are encouraged to try the code and see results after more epochs. Now both the real data sample and the fake data sample which is generated by the generator, will go to Discriminator. And discriminator is a binary classifier that will tell whether the sample is a fake sample or a real sample.

5.1 GANs: - Basic Discriminator Network

The Discriminator in a Generative Adversarial Network (GAN) plays a critical role as one of the two main components of the network. It operates in tandem with the other component, the Generator. The Discriminator's primary function is to distinguish between real and fake data. For instance, if we are dealing with image data, the Discriminator would receive images from both the real-world dataset and the images created by the Generator. It then has to classify these inputs, determining which images are real (from the dataset) and which are fake (generated by the Generator).

This is typically achieved through a binary classification model, where the Discriminator assigns a probability to each input image that represents how likely it is to be real. If the Discriminator is working effectively, it will assign higher probabilities to real images and lower probabilities to fake ones.

The goal of training the Discriminator is to improve its accuracy in classifying images correctly. Meanwhile, the Generator is trained to fool the Discriminator into thinking the images it produces are real. This adversarial process makes the Discriminator (and the GAN as a whole) better over time, leading to the generation of highly realistic synthetic data.

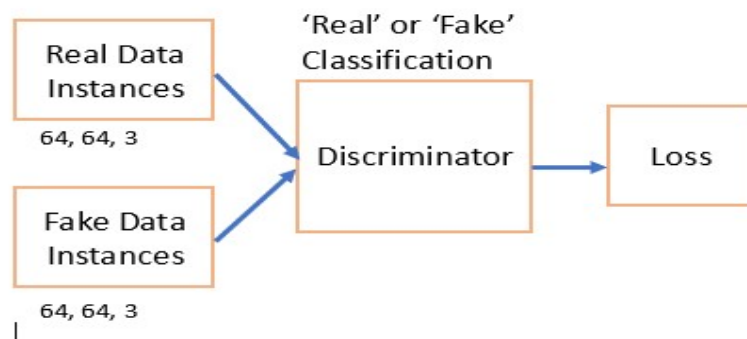


Figure3: - Discriminator Basic Architecture Diagram

So first of all, we are starting with the discriminator. So, the above diagram is the basic discriminator. The discriminator will get real data from the training data, and fake data instances from Generator. The generator will generate fake samples similar to the original size, and in our data set, we are using this size 64 X 64 X 3. So that's why through the generator we have to perform up-sampling on our fake image, i.e. the image which we will generate from this generator. The size of that image should be similar to this size.

Now these two values, these two images will go to the discriminator. Now discriminator is a neural network and the task of this neural network is to finally classify whether the image is a real image or a fake image. If the image is a real image, this is a binary classifier. It will give us output as one (1). If the image is a real image and if the image is fake then it will return zero (0).

So then after that, we have a loss function. So, when we calculate loss after that we want to update the weights right then only, we will run this training for several epochs to improve the results. So, after every epoch, if we want to improve the results then we have to update the weights, and for weight updating, we are using Adam optimizer.

So, in summary, we can divide the steps to define the Role of a Discriminator.

- **GANs Training Process**

- Generator begins with a random sample to produce the initial output
- Discriminators can easily identify these as fake due to their randomness

- **Generator Improvement Over Time**

- With the progression of epochs (for instance, 1000, 1500, and 2500 epochs), the generator starts producing more realistic results

- **Role of Feedback in Training**

- The discriminator provides feedback to the generator after every pass
- This feedback is used by the generator to improve its output

- **Simultaneous Improvement of Both Components**

- Both the generator and discriminator improve over time
- The discriminator's accuracy improves by comparing the generated data with the real data.

5.2 GANs: - Discriminator Network: A Deep Inside

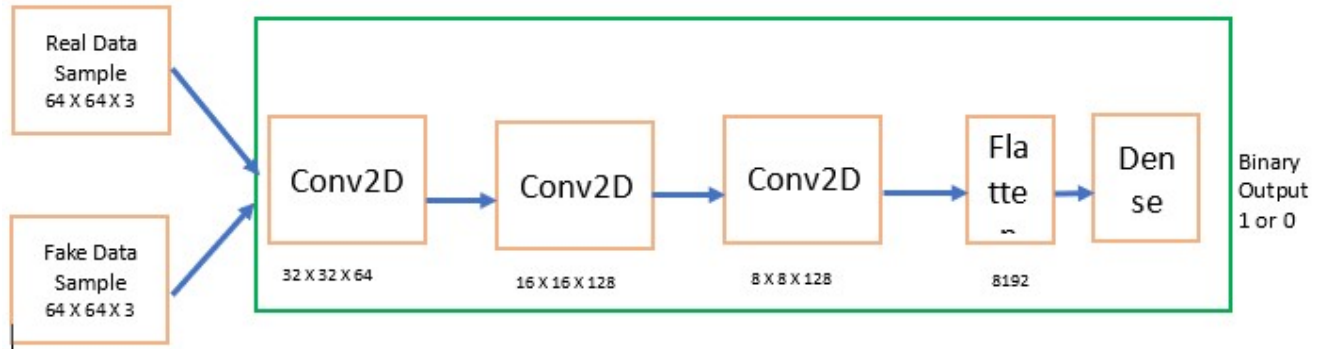


Figure4: - Discriminator Architecture Low-Level Diagram

So, based on the above Clarifications (in Section - 5), if we look at this whole green block is a discriminator and has several layers which are inside the discriminator's real data. Fake data will go to a discriminator and the very first layer is a convolutional layer. So now the image size of the real sample is $64 \times 64 \times 3$. Height and width are 64 and the number of channels is three because this is a colored image. In the same way, the fake sample which we have created should have the same dimension. So that's why we have this.

Now we need to down-sample this image, when we'll apply the first convolutional layer, after that convolutional layer 64×64 will get reduced to 32×32 and 64 number of filters we are using. And after that over here we are having convolutional layer and we are using batch normalization and ReLU.

So, after that, this $32 \times 32 \times 64$ will go to the next convolutional layer. And over here also we are applying down-sampling, and after this, we will get an image would be of this size. This will go to the next layer convolutional layer. And this convolutional layer will down-sample the image to this $8 \times 8 \times 128$.

We have three convolutional layers in it. After that, we are flattening our image. Why we need to flatten our image because now we want to classify whether the image is a fake image or a real image. And the classifier always accepts data in a one-dimensional format. So that's why we are flattening our image. Flattening simply means multiplying this last bit i.e. $8 \times 8 \times 128$ is 8192.

So, we have flattened the image. After that, we have a dense layer over here. We are performing a classification right between this real data and fake data and the activation function which we are using over here is Sigmoid, and finally, one simply means real data and zero means fake data is received.

So, in summary, we can divide the steps to define the Role of a Discriminator at a very Low Level.

- **Discriminator Structure and Input**

- The green block represents the discriminator, consisting of multiple layers
- Both real and fake data, with dimensions of 64x64x3, are fed into the discriminator

- **Convolutional Layers and Down-sampling**

- The first layer is a convolutional layer that reduces the image size from 64 X 64 to 32 X 32 using 64 filters.
- Each subsequent convolutional layer further down samples the image, ultimately reaching 8 X 8 X 128.

- **Use of Batch Normalization and ReLU**

- Batch normalization and ReLU activation functions are applied after each convolutional layer

- **Image Flattening for Classification**

- The image is flattened to a one-dimensional form (8192) for classification
- Flattening is necessary as classifiers require one-dimensional data

- **Final Dense Layer and Output**

- A dense layer is used for classification between real and fake data, using a sigmoid activation function
- Outputs are binary, with '1' indicating real data and '0' indicating fake data.

5.3 GAN: - Generator Network

The Generator in a Generative Adversarial Network (GAN) is one of the two key components in the network, the other being the Discriminator. The role of the Generator is to create new data instances that resemble the real data. Its function is akin to a team of counterfeiters, trying to produce fake currency and use it without detection. In the context of GANs, the Generator tries to create, or 'generate', new data (for example, images) that are indistinguishable from the real data to the best extent possible.

The Generator begins the process with a set of random numbers, often referred to as a random noise vector or latent vector. This vector serves as a seed for the generation process. Through the network layers, this random noise is transformed into data of the same shape as the real data. Over time, with feedback from the Discriminator, the Generator learns to produce data that is increasingly similar to the real data. The primary goal of training the Generator is to improve its ability to create data that the Discriminator cannot distinguish from real data. In the back-and-forth process of the GAN, the Generator is continually trying to 'fool' the Discriminator into misclassifying its generated data as real.

Now, let's talk about generators. Now Generator, the task of the Generator is to create a fake sample. And those fake faces will go to the discriminator and the discriminator will tell whether this face is a fake face or a real face. And the motive of the Generator is to work hard to create more realistic faces so that our Generator actually Generator wants to fool the discriminator means if Generator is giving fake data. And what the generator wants is generator wants to produce more realistic results, more realistic fake images. So, the discriminator would not be able to distinguish whether this is a real face or a fake face created by the generator.

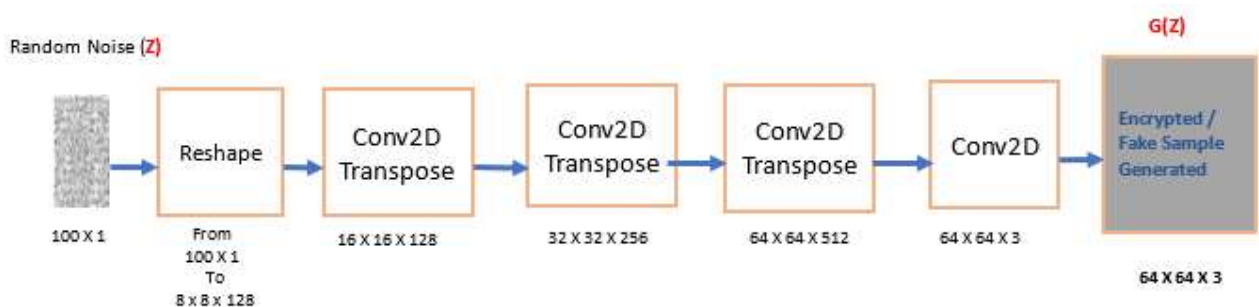


Figure5: - Generator Architecture Low-Level Diagram

So, in the above diagram

- The first layer expands the random Noise.
- The Network goes from 100 X 1 to 8 X 8 X 128.
- This Network takes a 100 X 1 Noise Vector (Z) and maps it into the G(Z) Output which is 64 X 64 X 3

So initially, the generator will always start with a random noise. So, in the above diagram (Z) is our random noise. And this is a vector size of 100×1 . This is a one-dimensional vector, so this vector, is the shape of 100×1 . So, we are reshaping it and from 100×1 , 2, 3,, we have reshaped this random noise into this shape, $8 \times 8 \times 128$. So, the very first step is to reshape this random noise. So, we have reshaped it after reshaping, we are using conv 2D transpose. Now, why we are using these conv 2D transpose layers? Because what we want is we want to up-sample our image after every layer. So that's what we are doing We are reshaping our noise to $8 \times 8 \times 128$. And conv 2D transpose will do up-sampling. Up-sampling means it will give you a higher version of that, higher resolution, it will perform up-sampling. So, after going through this Convolution layer, the size will become $16 \times 16 \times 128$. After that, this will go to the next conv 2D transpose layer.

And this layer will also perform up-sampling and it will become $32 \times 32 \times 256$. And now this value will go to the next quantity transpose layer. This will again do some up-sampling and we will get the value. finally, we have a convolutional layer and we will get this image size, so basically, the output of the generator fake sample and that sample should be of the same size of $64 \times 64 \times 3$. Remember, this image size should be similar to the image sizes of our real images. So, our task is to up-sample it to this. But we start with a low-resolution image, so this is how it works.

So, in summary, we can divide the steps to define the Role of a Generator at a very Low Level.

- **Generator's Task and Goal**

- The generator creates fake samples, aiming to produce increasingly realistic images
- The ultimate objective is to fool the discriminator into classifying these fake samples as real

- **Random Noise as Initial Input**

- The generator starts with a random noise vector of size 100×1
- This vector is reshaped into $8 \times 8 \times 128$, a common practice in GANs to begin with low-resolution images

- **Convolutional Transpose Layers**

- Conv2D Transpose layers are used for up-sampling, increasing the image resolution
- The first Conv2D Transpose layer up samples the image from $8 \times 8 \times 128$ to $16 \times 16 \times 128$

- **Continued Up-sampling**

- Further Conv2D Transpose layers continue to up-sample the image, reaching dimensions of $32 \times 32 \times 256$, and then $64 \times 64 \times 128$

- **Final Output**

- A final convolutional layer creates an image of size $64 \times 64 \times 3$, matching the real image sizes
- The generator output is thus a fake sample with the same dimensions as the original real images

6. Learning Steps from Program / Code & their Outcome

The outcomes or results of a Generative Adversarial Network (GAN) can be quite impressive, as they are capable of generating highly realistic synthetic data. These results are contingent on the type of data the GAN was trained on. At the same time, it's important to note that training a GAN can be quite challenging. This is due to the adversarial nature of the training process, which can lead to problems like mode collapse, where the generator produces a limited variety of outputs. Nevertheless, when successfully trained, the results of GANs can be highly compelling. When a Generative Adversarial Network (GAN) is used for the generation of fake and encrypted images, the results can be profound, especially concerning the level of complexity and fidelity in the generated images.

As we discussed GANs can generate incredibly realistic synthetic images. The discriminator's feedback helps the generator to continually refine its outputs, eventually creating images that are almost indistinguishable from the real ones. The concept of using GANs for image encryption is based on leveraging the complexity of the generative model to create encrypted images that are extremely difficult to decipher without the right key (which is usually the trained GAN model). The generator can transform a simple input (like a random noise vector) into a complex, encrypted image that, to an untrained observer, appears as noise or an unrelated image.

➤ Input Image Data Set: -

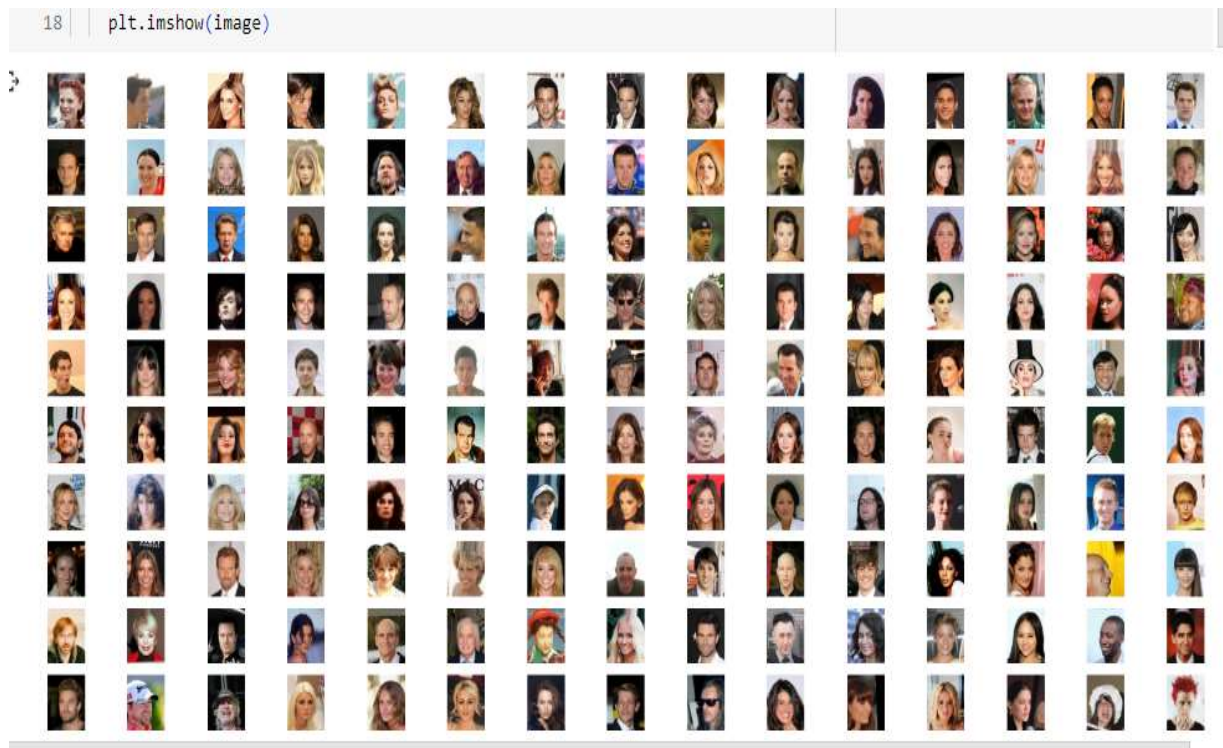


Figure6: - Celeb Image Data Set

➤ Landmark

	image_id	lefteye_x	lefteye_y	righteye_x	righteye_y	nose_x	nose_y	leftmouth_x	leftmouth_y	rightmouth_x	rightmouth_y
0	000001.jpg	69	109	106	113	77	142	73	152	108	154
1	000002.jpg	69	110	107	112	81	135	70	151	108	153
2	000003.jpg	76	112	104	106	108	128	74	156	98	158
3	000004.jpg	72	113	108	108	101	138	71	155	101	151
4	000005.jpg	66	114	112	112	86	119	71	147	104	150

<matplotlib.image.AxesImage at 0x7de1b013a620>

original image



Image with landmarks



Figure7: - Landmark is set of 000002.jpg

➤ Bounding Box Data

	image_id	x_1	y_1	width	height
0	000001.jpg	95	71	226	313
1	000002.jpg	72	94	221	306
2	000003.jpg	216	59	91	126
3	000004.jpg	622	257	564	781
4	000005.jpg	236	109	120	166

<matplotlib.image.AxesImage at 0x7de1a83133a0>

original image




Image with bbox and landmarks



Figure8: - Bounding Box is set and Landmarks

➤ Partition DataSet



	image_id	partition
0	000001.jpg	0
1	000002.jpg	0
2	000003.jpg	0
3	000004.jpg	0
4	000005.jpg	0

```
0    162770
2    19962
1    19867
Name: partition, dtype: int64
```

Figure8: - Partition is counted

We have zero, two, and one means we have three kinds of different partitions in it.

- partition is equal to zero in our train images, which simply means partition zero is the images that we will be using to train our algorithm.
- Partition one, partition one is all those images that we will be using for validation of our data, validating our data.
- partition equals two. That simply means these are the images that we will be using to test our algorithm.

➤ Attribute DataSet

	image_id	5_o_Clock_Shadow	Arched_Eyebrows	Attractive	Bags_Under_Eyes	Bald	Bangs	Big_Lips	Big_Nose	Black_Hair	...	Sideburns	Smiling	Straight_Hair	Wavy_Hair	Wearing_E
0	000001.jpg	-1	1	1	-1	-1	-1	-1	-1	-1	...	-1	1	1	-1	
1	000002.jpg	-1	-1	-1	1	-1	-1	-1	1	-1	...	-1	1	-1	-1	
2	000003.jpg	-1	-1	-1	-1	-1	-1	1	-1	-1	...	-1	-1	-1	1	
3	000004.jpg	-1	-1	1	-1	-1	-1	-1	-1	-1	...	-1	-1	1	-1	
4	000005.jpg	-1	1	1	-1	-1	-1	1	-1	-1	...	-1	-1	-1	-1	

5 rows × 17 columns

➤ The discriminator is defined.

```
Model: "discriminator"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 64)	3136
leaky_re_lu (LeakyReLU)	(None, 32, 32, 64)	0
conv2d_1 (Conv2D)	(None, 16, 16, 128)	131200
leaky_re_lu_1 (LeakyReLU)	(None, 16, 16, 128)	0
conv2d_2 (Conv2D)	(None, 8, 8, 128)	262272
leaky_re_lu_2 (LeakyReLU)	(None, 8, 8, 128)	0
flatten (Flatten)	(None, 8192)	0
dropout (Dropout)	(None, 8192)	0
dense (Dense)	(None, 1)	8193

Total params: 404,801
 Trainable params: 404,801
 Non-trainable params: 0

➤ The Generator is defined.

```
Model: "generator"
```

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 8192)	827392
reshape (Reshape)	(None, 8, 8, 128)	0
conv2d_transpose (Conv2DTranspose)	(None, 16, 16, 128)	262272
leaky_re_lu_3 (LeakyReLU)	(None, 16, 16, 128)	0
conv2d_transpose_1 (Conv2DTranspose)	(None, 32, 32, 256)	524544
leaky_re_lu_4 (LeakyReLU)	(None, 32, 32, 256)	0
conv2d_transpose_2 (Conv2DTranspose)	(None, 64, 64, 512)	2097664
leaky_re_lu_5 (LeakyReLU)	(None, 64, 64, 512)	0
conv2d_3 (Conv2D)	(None, 64, 64, 3)	38403

Total params: 3,750,275
 Trainable params: 3,750,275
 Non-trainable params: 0

7. Experimental result

➤ Generated Image Samples (Output)

This data set variable is having the Real Celeb images in it and epochs are 2809. So finally, below is the block of an image that will show us some generated images. So over here, these images are very small. So, these are the fake samples, fake faces which we have generated using a generator.








	GAN Image							
Real Image	Epoch	1000	1500	2000	2500	2565	2808	2809

Figure9: - Generated Image Samples at Epoch

The first image shown in the above box is for 1000 epochs. If you'll see these images from the very first epoch, you're not understanding them right now, but as soon as we increased the epoch, the image can be seen and available above. So, this is the fake image, Generated by our generator after 1000, 1500, 2000, 2500, 2565, 2808, and 2809 epochs. So, these all images are fake images generated by our generator. So right now, the results are not that much good, but can say close to the Real Image. So, to get the image much closer than the Real Image, we need to run our code for more epochs to get good results.

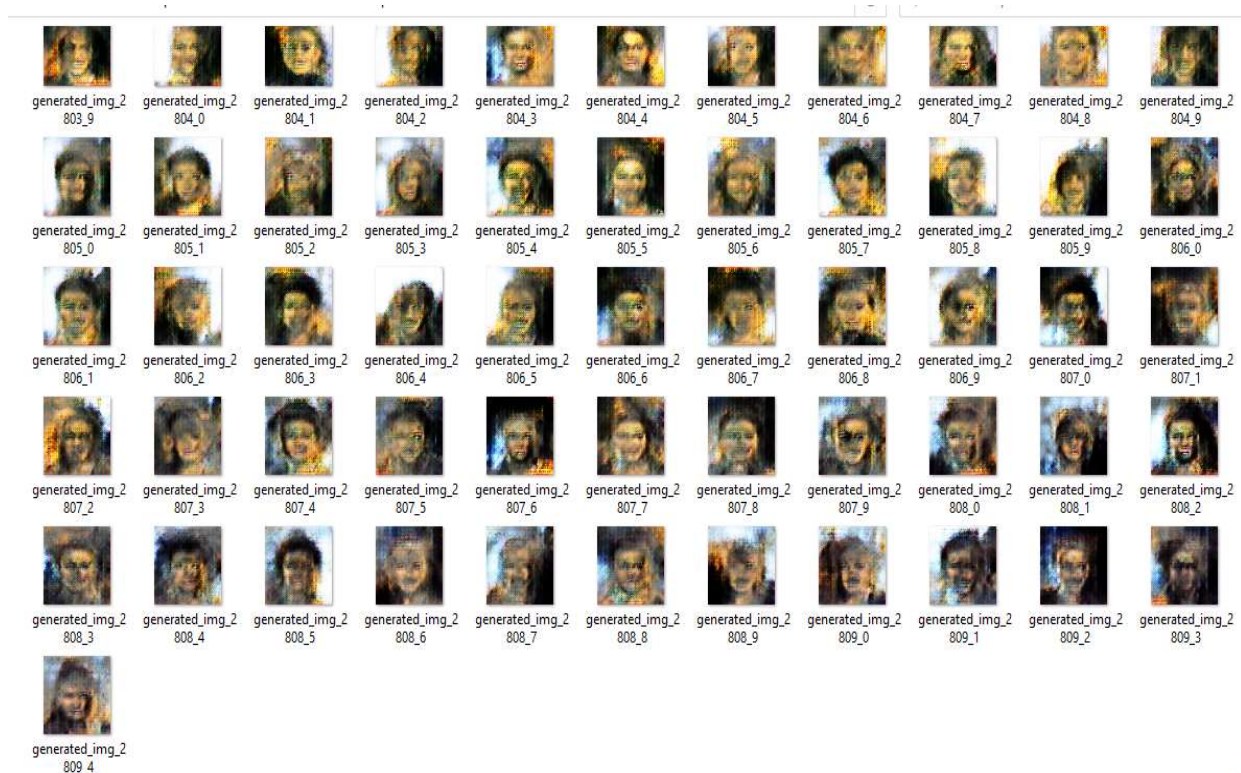


Figure10: - Generated Image Samples at various Epoch

8. Conclusion

In conclusion, leveraging Generative Adversarial Networks (GANs) for image encryption has proven to be an effective and innovative approach to secure image data transmission. This method excels at preserving the high-level features of the image during encryption, thus ensuring image data integrity. By using GANs, we can achieve higher levels of security against various types of cyber-attacks. However, as we move forward, it is crucial to consider the computational cost and complexity of this approach. Therefore, researchers should strive for a balance between robustness, computational efficiency, and ease of implementation. As AI continues to evolve, exploring the integration of other emerging technologies like quantum computing into the field of image encryption using GANs could open up new possibilities for enhanced security measures.

The application of Deep Convolutional Generative Adversarial Networks (DCGANs) for image encryption, as discussed in this presentation, exemplifies the profound possibilities of leveraging advanced AI in secure image data. The process involves the interplay between the GAN's generator and discriminator components, with the former progressively improving its production of 'fake' images to mislead the latter. The training dataset used in this demonstration, the celeb faces attribute, which contains a vast range of celebrity face images, allows the GAN to learn and generate a wide variety of outputs.

The discriminator's role as a classifier and the importance of using an optimizer and loss function to update its weights were highlighted. Simultaneously, the generator's process of beginning with random noise and using transpose convolutional layers to up-sample the image was shown to produce increasingly realistic encrypted fake samples. The training process of the model, set to run for 2809 epochs, showed clear progression and improvement in the generated images, implying the potential robustness of this encryption approach. It was suggested that more training could further enhance this. As a whole, this exploration of DCGAN-based image encryption opens exciting avenues for future research and development in secure image data transmission.

9. References

Book

- Generative Adversarial Networks for Image Generation by Xudong Mao , Qing Li
- Artificial Intelligence: A Modern Approach” by Stuart Russel and Peter Norvig Sec. 5.1-5.4

Conference Paper (Paper Presented at a Conference)

- Learnable Image Encryption Masayuki Tanaka, Member, IEEE Published in: 2018 IEEE [1] International Conference on Consumer Electronics-Taiwan (ICCE-TW) Date of Conference: 19-21 May 20
- Image to Perturbation: An Image Transformation Network for Generating Visually Protected Images for Privacy-Preserving Deep Neural Networks. Date of publication April 22, 2021

Journal

- An Overview of Compressible and Learnable Image Transformation with Secret Key and its Applications Hitoshi Kiya¹ by April Pyone Maung Maung¹, Yuma Kinoshita¹, Shoko Imaizumi² and Sayaka Shiota¹. [4]. supported, in part, by JSPS KAKENHI Grant Number JP21H01327, JST CREST Grant Number JPMJCR20D3, and Support Center for Advanced Telecommunications Technology Research, Foundation (SCAT).
- Color Image Encryption Based on Deep Learning and Block Embedding by Yi Liu ,Gang Cen,Bijun Xu and Xiaogang Wang [3] Research Article | Open Access Volume 2022 | Article ID 6047349