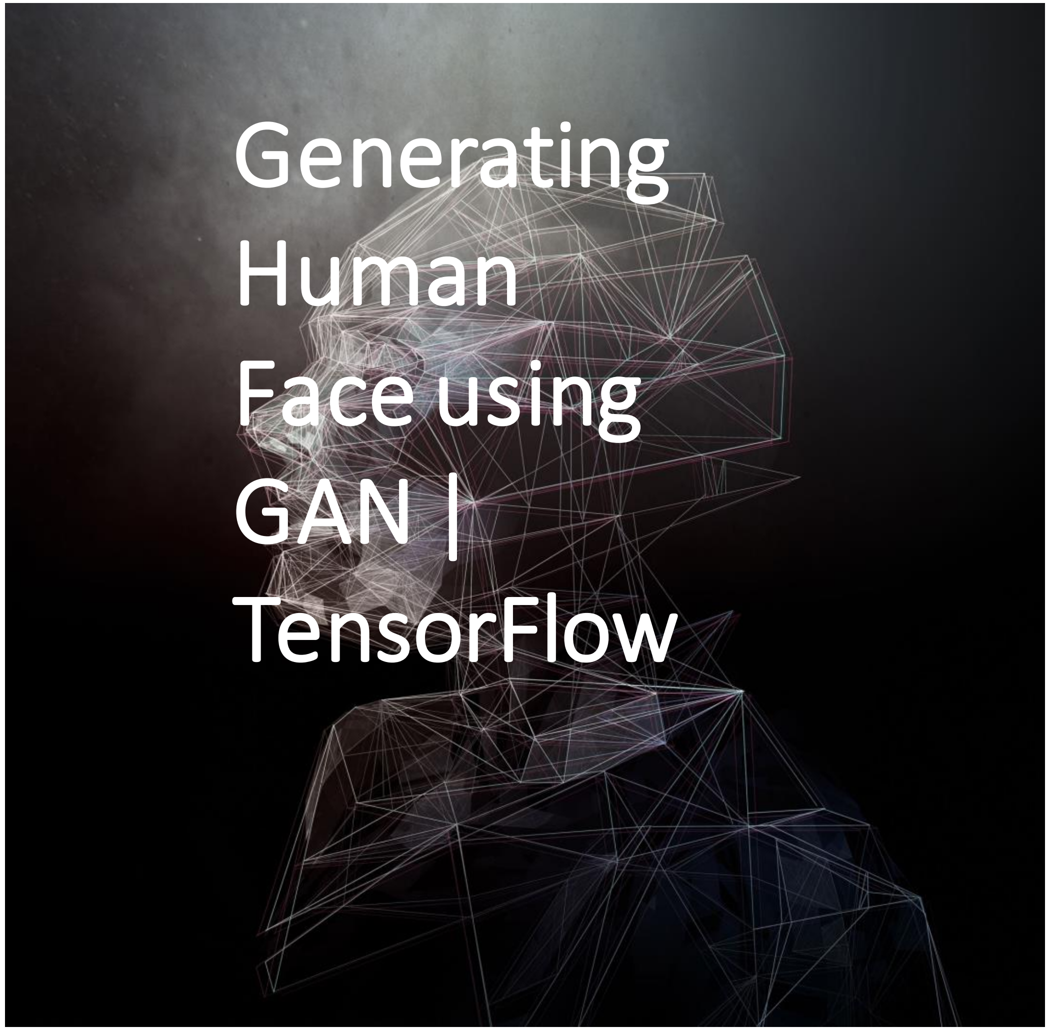
HUMAN FACE GENERATION USING DEEP LEARNING

Guided By,
Mr D Vikram
M.E. (Ph.D)..,
Assistant Professor

Presented By,
Amrish Ahamed B(721820243007)
Dhigin Chanikya g(721820243010)
Jayakumar C(721820243023)
Salha Nasreen M(721820243046)
Saru Preethika T(721820243052)



Introduction

- Generative Adversarial Networks (GANs)
- Deep Convolution GAN (DC-GAN)
- Variations of GANs for Face Genera
- Data Collection and Preprocessing
- Normalizing the images
- Creating the network
- Training and Fine-tuning
- Evaluation Metrics for Face Generation

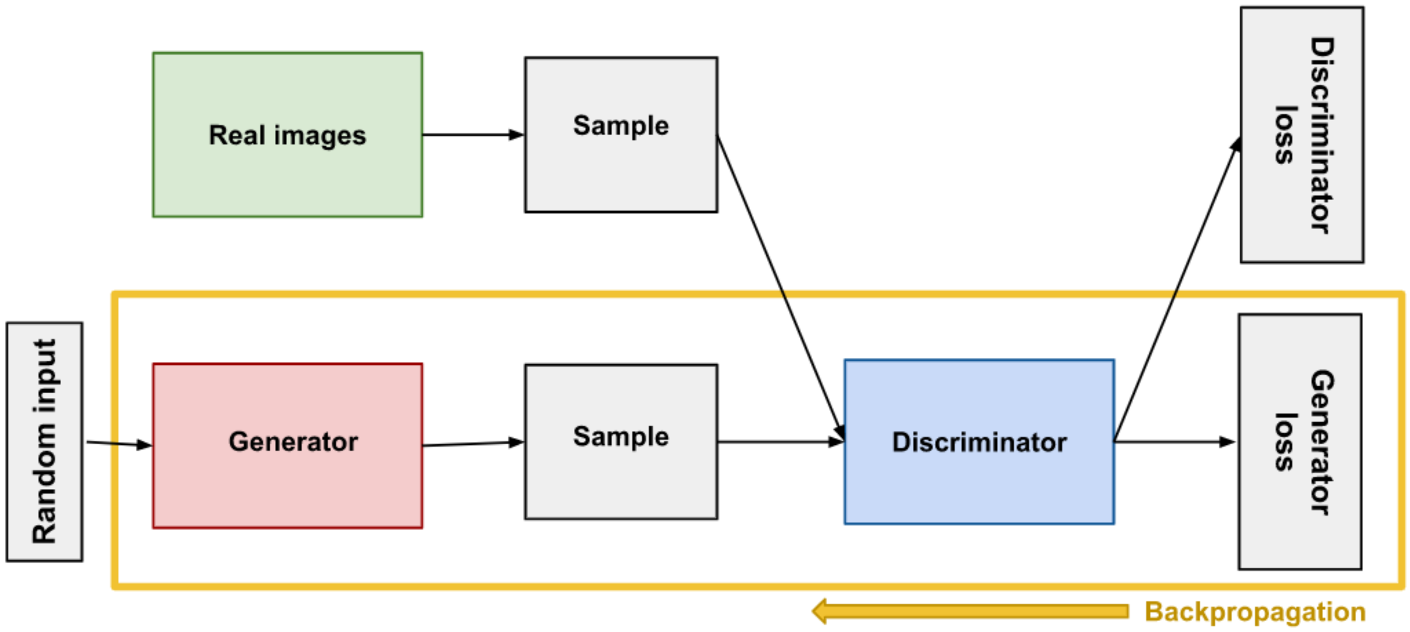
Overview

The concept of human face generation in deep learning revolves around using artificial intelligence techniques, specifically deep neural networks, to generate realistic and convincing human faces. Deep learning models, particularly Generative Adversarial Networks (GANs), have shown remarkable success in synthesizing high-quality facial images that resemble real human faces.

Here's a breakdown of the concept of human face generation in deep learning

A large dataset of real human faces is collected, typically consisting of thousands or even millions of facial images. These images should cover a wide range of demographics, poses, expressions, and lighting conditions to ensure diversity and generalization.

- Deep Convolution GAN (DC-GAN)
  GANs consist of two main components: a generator network and a discriminator network.
- The generator network takes random noise as input and generates synthetic samples, in this case, human faces.
- The discriminator network evaluates the generated samples and distinguishes them from real human faces.
- Both networks are trained simultaneously, with the generator aiming to fool the discriminator, and the discriminator striving to correctly classify real and fake samples.
- Through this adversarial process, GANs learn to generate increasingly realistic and visually appealing human faces..



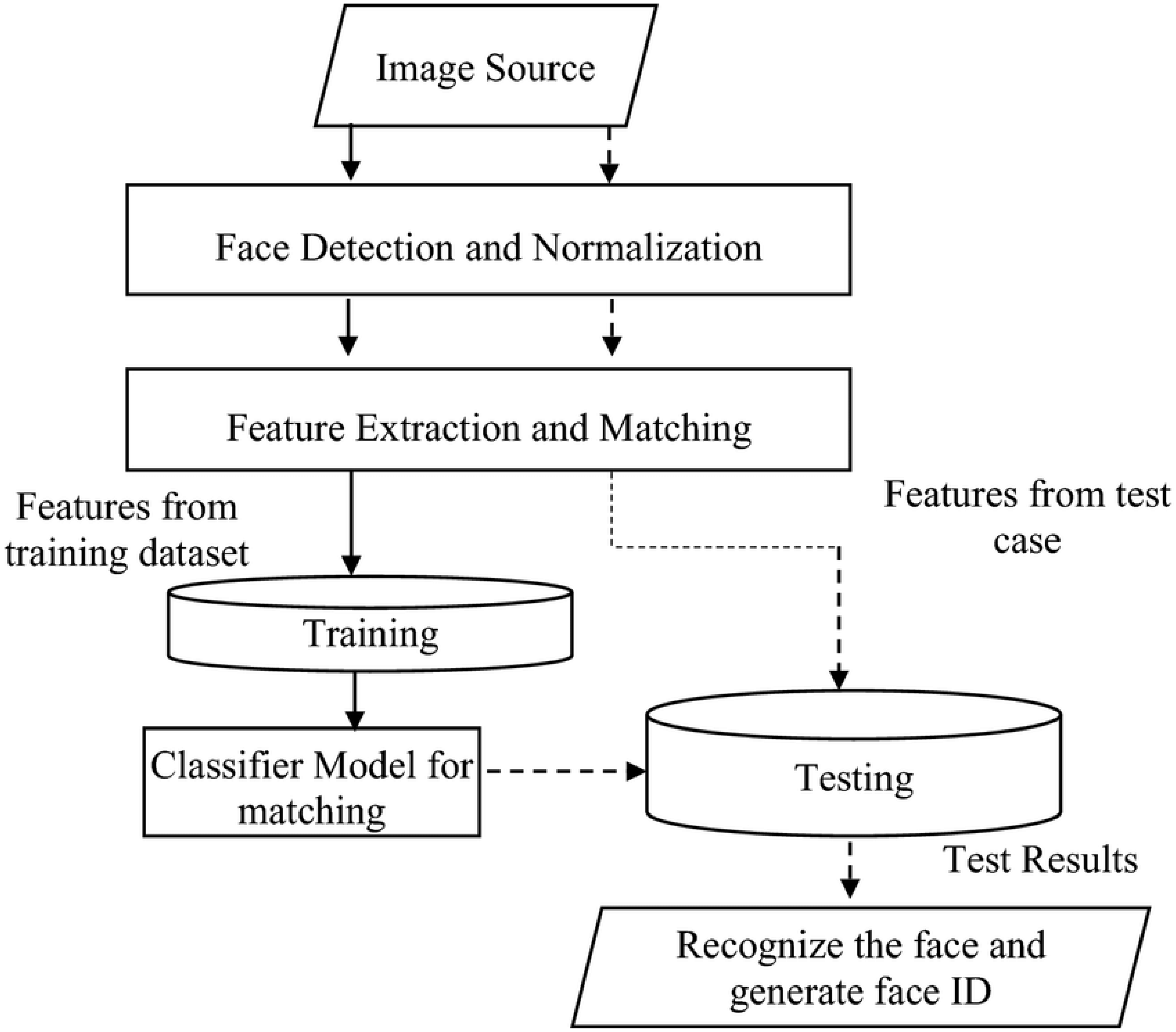
- This is how our **pipeline** will look like
- 1.Normalize the images.
- 2.Create Generator and Discriminator network.
- 3.Train the network and generate new faces.

Proposed System

As a first step, we import libraries that we will make use of.

We load all the images using PIL. While loading the images we crop all images around the face and resize them to (64, 64, 3)

These images are in the range of (0, 255). We squash the bit range of these images between (-1, 1), which is in the range of tanh activation.



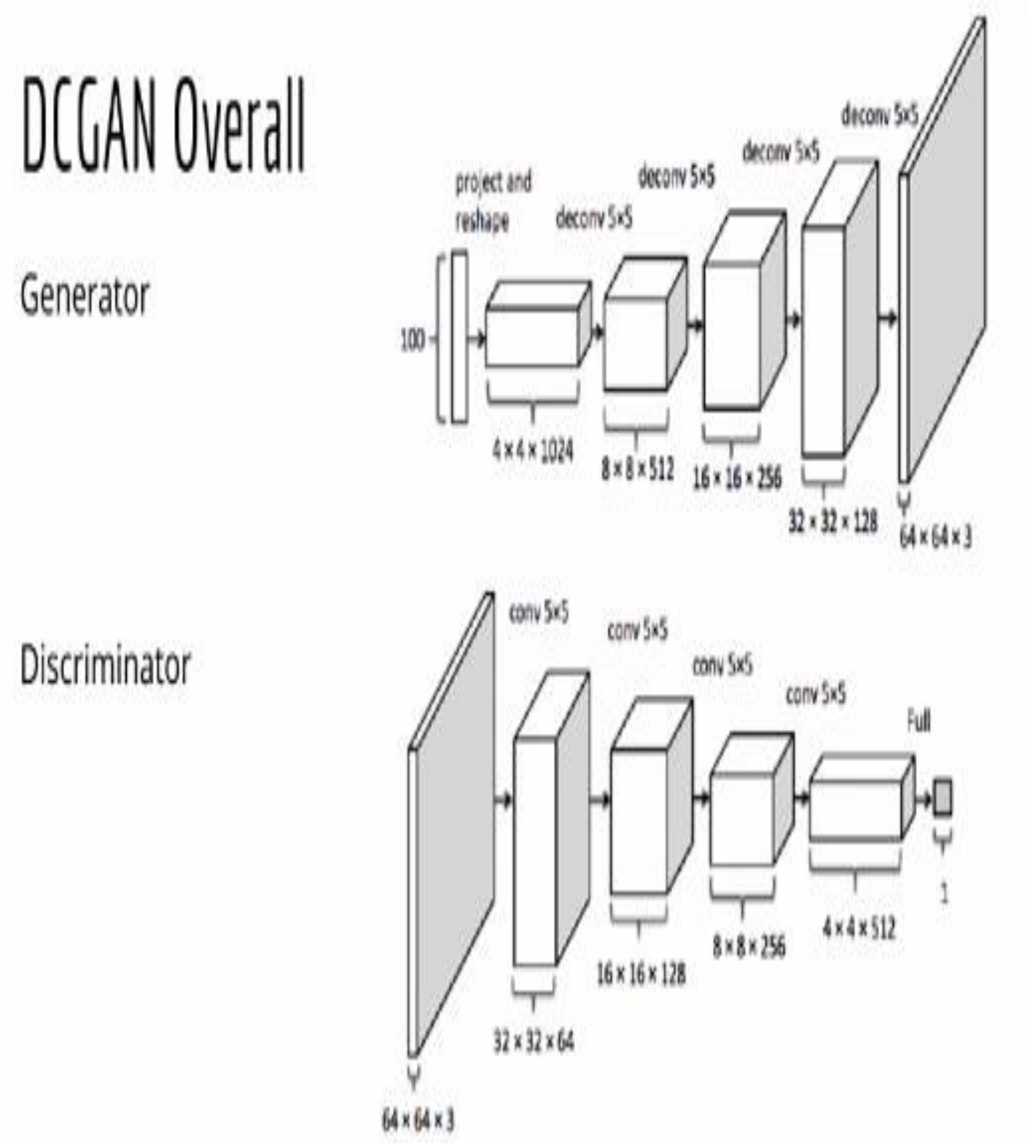
Deep Neural Networks for Face Generation

Deep neural networks, particularly convolutional neural networks (CNNs), are commonly employed in the generator and discriminator networks of GANs for face generation.

CNNs are powerful models for image processing, capable of learning intricate patterns and features from visual data.

The generator network often utilizes a series of transposed convolutions (also known as deconvolutions) to upsample the random noise and transform it into a realistic facial image.

The discriminator network uses convolutional layers to extract meaningful features from both real and generated images, enabling it to distinguish between the two.



METHODOLOGY

Principal Component Analysis (PCA):

- PCA is a statistical technique used for dimensionality reduction and feature extraction.
- In face generation, PCA can be applied to capture the main variations in a dataset of facial images.
- The eigenvectors (principal components) obtained from PCA can be used to generate new face samples by manipulating their corresponding eigenvalues.

Covariance Matrix:

Calculate the covariance matrix C of the preprocessed data. The covariance between two variables x and y can be computed as: $\mathrm{Cov}(x, y) = (1/n) * \Sigma[(x_i - \mu_x) * (y_i - \mu_y)]$

Variational Autoencoders (VAEs):

- VAEs are generative models that combine deep neural networks with probabilistic modeling.
- They aim to learn a low-dimensional representation (latent space) of facial images and generate new samples from that space.
- The VAE objective involves maximizing the evidence lower bound (ELBO), which balances reconstruction accuracy and latent space regularization.

```python
import numpy as np
import pandas as pd
import os

PIC_DIR = f'drive/My Drive/celebdata/img_align_celeba/img_align_celeba/'

from tqdm import tqdm
from PIL import Image

IMAGES_COUNT = 10000

ORIG_WIDTH = 178
ORIG_HEIGHT = 208
diff = (ORIG_HEIGHT - ORIG_WIDTH) // 2

WIDTH = 128
HEIGHT = 128

crop_rect = (0, diff, ORIG_WIDTH, ORIG_HEIGHT - diff)

images = []
for pic_file in tqdm(os.listdir(PIC_DIR)[:IMAGES_COUNT]):
    pic = Image.open(PIC_DIR + pic_file).crop(crop_rect)
    pic.thumbnail((WIDTH, HEIGHT), Image.ANTIALIAS)
    images.append(np.uint8(pic))
```

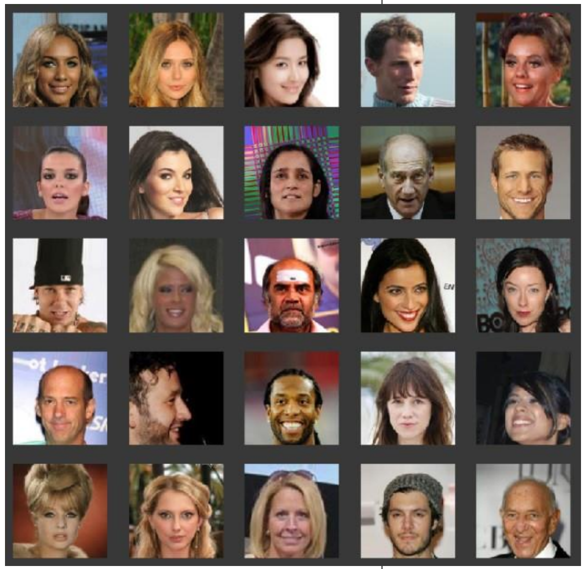

In [14]:

```python
images = np.array(images) / 255
print(images.shape)


from matplotlib import pyplot as plt
```

(10000, 128, 128, 3)

In [16]:

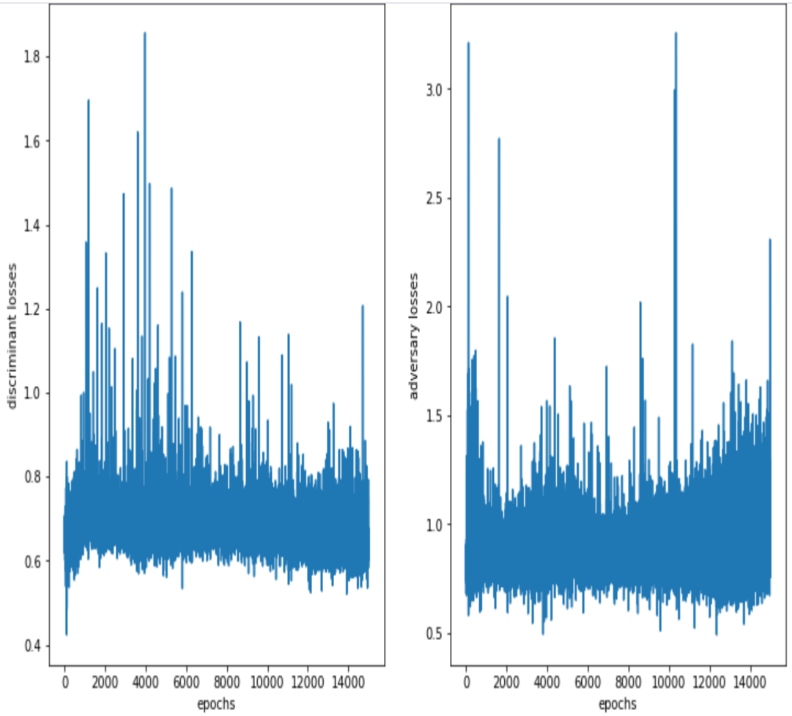
```
plt.figure(1, figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i+1)
    plt.imshow(images[i])
    plt.axis('off')
plt.show()
```



```
    combined_images = np.concatenate([generated, real])

    labels = np.concatenate([np.ones((batch_size, 1)),
np.zeros((batch_size, 1))])
    labels += .05 * np.random.random(labels.shape)

    d_loss = discriminator.train_on_batch(combined_images, labels)
    d_losses.append(d_loss)

    latent_vectors = np.random.normal(size=(batch_size, LATENT_DIM))
    misleading_targets = np.zeros((batch_size, 1))

    a_loss = gan.train_on_batch(latent_vectors, misleading_targets)
    a_losses.append(a_loss)

    start += batch_size
```

```
iters = 20000
batch_size = 16RES_DIR = 'res2'
FILE_PATH = '%s/generated_%d.png'
if not os.path.isdir(RES_DIR):
    os.mkdir(RES_DIR)CONTROL_SIZE_SQRT = 6
control_vectors = np.random.normal(size=(CONTROL_SIZE_SQRT**2,
LATENT_DIM)) / 2start = 0
d_losses = []
a_losses = []
images_saved = 0
for step in range(iters):
    start_time = time.time()
    latent_vectors = np.random.normal(size=(batch_size, LATENT_DIM))
    generated = generator.predict(latent_vectors)

    real = images[start:start + batch_size]
```

```python
CONTROL_SIZE_SQRT = 6
control_vectors = np.random.normal(size=(CONTROL_SIZE_SQRT**2, LATENT_DIM)) / 2

start = 0
d_losses = []
a_losses = []
images_saved = 0
for step in range(iters):
    start_time = time.time()
    latent_vectors = np.random.normal(size=(batch_size, LATENT_DIM))
    generated = generator.predict(latent_vectors)

    real = images[start:start + batch_size]
    combined_images = np.concatenate([generated, real])

    labels = np.concatenate([np.ones((batch_size, 1)), np.zeros((batch_size, 1))])
    labels += .05 * np.random.random(labels.shape)

    d_loss = discriminator.train_on_batch(combined_images, labels)
    d_losses.append(d_loss)

    latent_vectors = np.random.normal(size=(batch_size, LATENT_DIM))
    misleading_targets = np.zeros((batch_size, 1))

    a_loss = gan.train_on_batch(latent_vectors, misleading_targets)
    a_losses.append(a_loss)
```



```
import imageio
import shutilimages_to_gif = []
for filename in os.listdir(RES_DIR):
    images_to_gif.append(imageio.imread(RES_DIR + '/' + filename))
imageio.mimsave('trainnig_visual.gif', images_to_gif)
shutil.rmtree(RES_DIR)
```

```
    if start > images.shape[0] - batch_size:
        start = 0

    if step % 50 == 49:
        gan.save_weights('gan.h5')

        print('%d/%d: d_loss: %.4f,  a_loss: %.4f.  (%.1f sec)' % (step
+ 1, iters, d_loss, a_loss, time.time() - start_time))

        control_image = np.zeros((WIDTH * CONTROL_SIZE_SQRT, HEIGHT *
CONTROL_SIZE_SQRT, CHANNELS))
        control_generated = generator.predict(control_vectors)
        for i in range(CONTROL_SIZE_SQRT ** 2):
            x_off = i % CONTROL_SIZE_SQRT
            y_off = i // CONTROL_SIZE_SQRT
            control_image[x_off * WIDTH:(x_off + 1) * WIDTH, y_off *
```

```
HEIGHT:(y_off + 1) * HEIGHT, :] = control_generated[i, :, :, :]
        im = Image.fromarray(np.uint8(control_image * 255))
        im.save(FILE_PATH % (RES_DIR, images_saved))
        images_saved += 1
```

- **Training and Fine-tuning:**
  Creating the network
  The concept behind GAN is that it has two networks called **Generator Discriminator**. The job of the Generator is to generate realistic-looking images from the noise and to fool the discriminator. On the other hand, the job of the discriminator is to discriminate between real and fake images. These both networks are trained separately.

- Discriminator Network:
- The discriminator network aims to distinguish between real facial images and synthetic images generated by the generator.
- Design the discriminator network using CNNs, similar to the generator, to extract meaningful features from facial images.
- The architecture usually involves a series of convolutional layers followed by fully connected layers for classification.
- Incorporate activation functions and normalization techniques like batch normalization to improve the discriminative power of the network.
- Implement downsampling layers (e.g., max-pooling or strided convolutions) to reduce the spatial dimensions of the input.

Recent Advances and Applications

Recent Advances and Applications:
Highlight recent advancements in face generation, such as the use of self-attention mechanisms, progressive growing, and image-to-image translation.
Discuss practical applications of face generation, including character customization in video games, movie special effects, and virtual reality avatars

CONCLUSION

GANs and generative models general are very fun and perplexing. They encapsulate another step towards a world where we depend more and more on artificial intelligence. GANs have a huge number of applications in cases such as ***Generating examples for Image Datasets, Generating Realistic Photographs, Image-to-Image Translation, Text-to-Image Translation, Semantic-Image-to-Photo Translation, Face Frontal View Generation, Generate New Human Poses, Face Aging, Video Prediction, 3D Object Generation, etc.***

MukthShabd Journal

ISSN NO: 2347-3150
Scientific Journal Impact Factor – 4.6

ACCEPTANCE LETTER TO AUTHOR

Dear Author,

With reference to your paper submitted "AI Human Face Generator Using The Concept Of Deep Learning Technique" we are pleased to accept the same for publication in MSJ Volume XII, Issue 7, JULY 2023.

Manuscript ID: MSJ7339

Please send the scanned Copyright form and Registration form along with bank receipt of an online maintenance/processing fee of **2000 INR** Per paper. Please note that the amount we are charging is very nominal & only an online maintenance and processing fee.

The Fee includes:
Online maintenance and processing charge.
No limitation of number of pages.
Editorial fee.
Taxes.

Note:
Paper will be published online with in 24 hours after receiving the fee.
*Fee Paid for the Publication of the paper does not refund under any circumstances
*In case of any query please do not hesitate to contact us at submitmsj@gmail.com Early reply is appreciated.

DATE
8-JULY-2023

Sincerely,
Best regards,
M. ASHITOSH MEHATA
http://shabdbooks.com/

Sumit Ganguly
Editor-In-Chief
MSJ
www.shabdbooks.com

DOI : 10.37896/MSJ

Our Journal Project Was Accepted...

ANY QUERIES??

THANK YOU.....