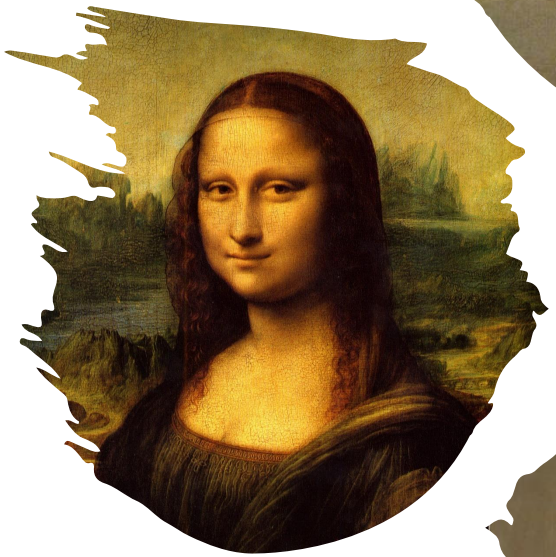
Portrait Generation Using GAN

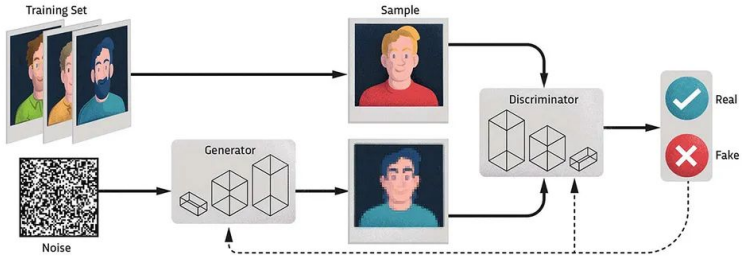
Group Members:

2029038 - Swamita sachan
2029047 - Ameesha singh
2029146 - Sarthak shrivastava
2029196 - Adarsh kumar



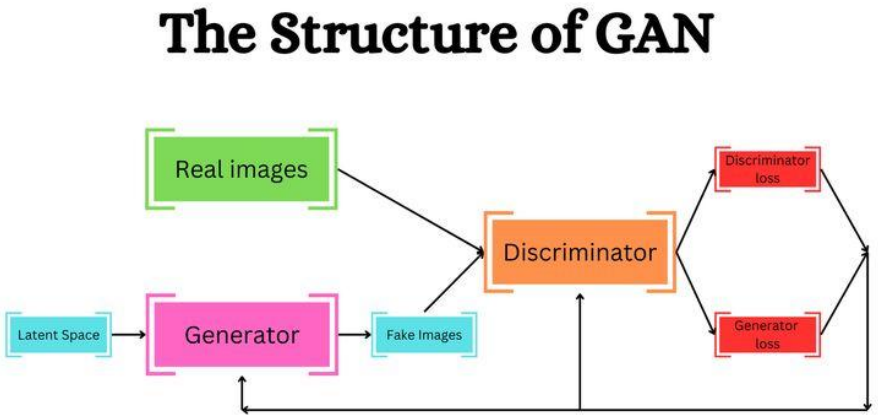
Introduction

- Introduction to Generative Adversarial Networks (GANs): Explain how GANs consist of two neural networks, the generator and the discriminator, competing against each other to generate realistic data.
- Brief overview of the project's objective: Generating portraits using GANs: Emphasize the significance of generating realistic portraits for various applications, including art, entertainment, and virtual simulations.
- Importance of generating realistic portraits: Discuss the importance of creating high-quality, diverse portraits for applications such as character design, virtual avatars, and historical reconstructions.



Architecture Diagram

- High-level overview of the GAN architecture: Visual representation of the generator and discriminator networks interconnected in an adversarial training setup.
- Schematic representation of the generator and discriminator networks: Illustrate the flow of information between the generator and discriminator during training, highlighting the role of feedback loops in improving the quality of generated portraits.



Methodology

- Data Collection
- Preprocessing
- Generator
- Discriminator
- Training

Dataset

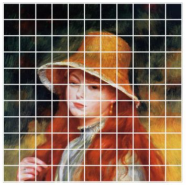
•The Link to the original Dataset Original Dataset. The Original Dataset Contains 15k+ images of Portraits. But many of them are Sketch and black & white images.

•Due to these reasons we had to go with a subset of dataset having 4k+ images.Dataset used in code Dataset.

Data Preprocessing

- Image Resizing - The images in the dataset are resized to a uniform size of (64, 64) pixels
- Normalization - The pixel values of the images are normalized to the range [0, 1] by dividing each pixel value by 255.0.
- Batching - The dataset is batched into mini-batches of size 32

Resized Image

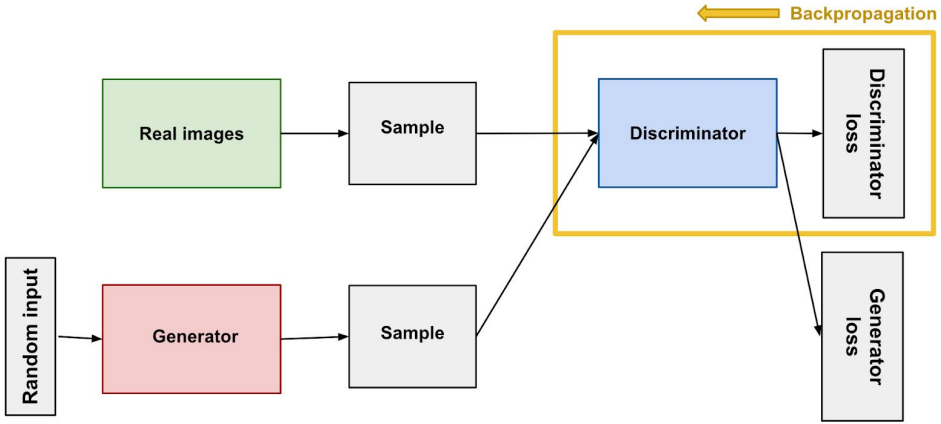


Discriminator

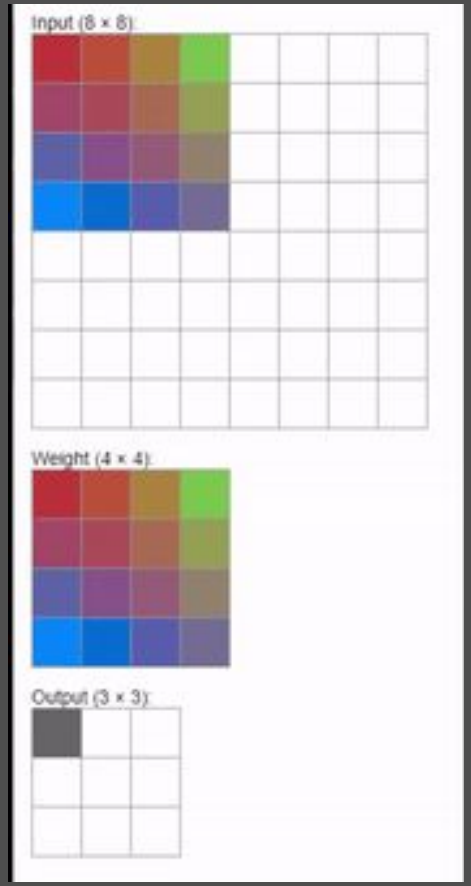
•A discriminator is a crucial component of a Generative Adversarial Network (GAN). Its primary function is to distinguish between real and fake data samples.

•The discriminator goal is to classify input data samples as either real (coming from the true data distribution) or fake (generated by the generator network).

•By learning to differentiate between real and fake samples, the discriminator provides feedback to the generator, helping it improve its ability to generate more realistic data.

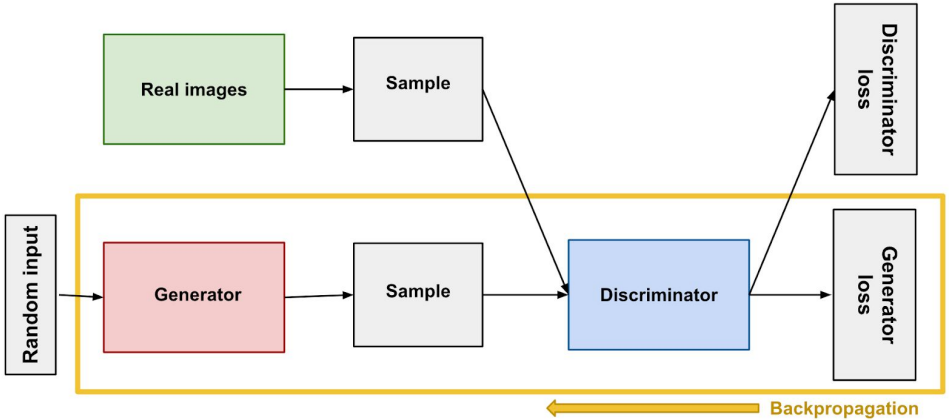


```python
discriminator = models.Sequential([
    layers.Input(shape=(64, 64, 3)),
    layers.Conv2D(64, kernel_size=4, strides=2, padding="same"),
    layers.LeakyReLU(alpha=0.2),
    layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
    layers.LeakyReLU(alpha=0.2),
    layers.Conv2D(256, kernel_size=4, strides=2, padding="same"),
    layers.LeakyReLU(alpha=0.2),
    layers.Flatten(),
    layers.Dropout(0.5),
    layers.Dense(1, activation="tanh")
], name="discriminator")
```

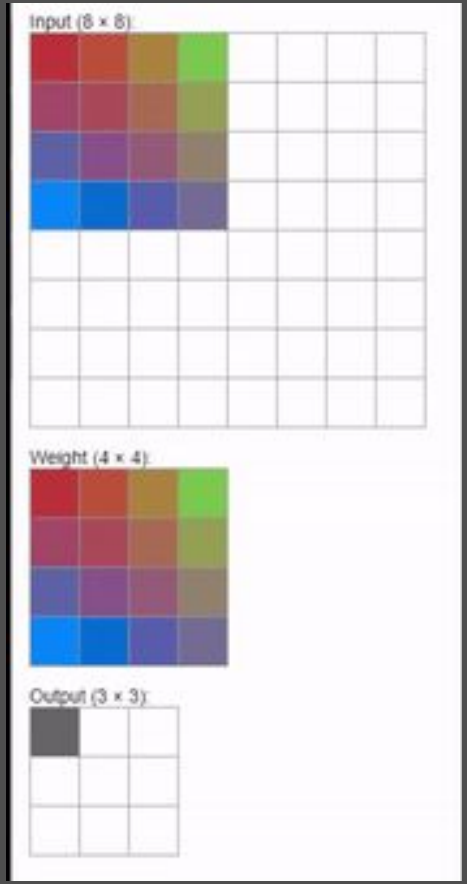


Generator

•In a Generative Adversarial Network (GAN), the generator is one of the two neural networks involved in the training process, alongside the discriminator. The generator's main objective is to create realistic data samples, such as images, audio clips, or text, from random noise or a latent space.

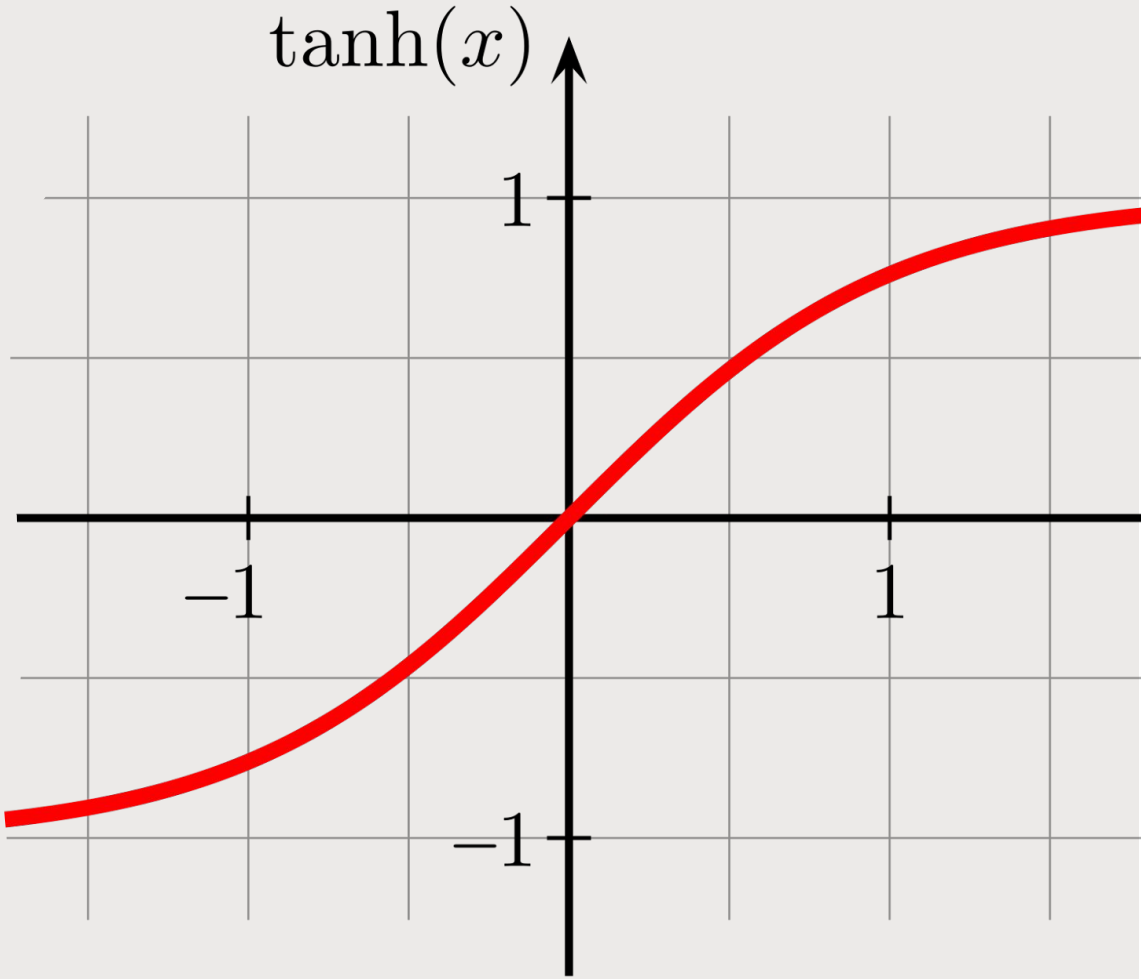


```python
latent_dim = 128
generator = models.Sequential([
    layers.Input(shape=(latent_dim,)),
    layers.Dense(8 * 8 * 128),
    layers.Reshape((8, 8, 128)),

    layers.Conv2DTranspose(128, kernel_size=4, strides=2, padding="same"),
    layers.BatchNormalization(),
    layers.LeakyReLU(alpha=0.2),

    layers.Conv2DTranspose(256, kernel_size=4, strides=2, padding="same"),
    layers.BatchNormalization(),
    layers.LeakyReLU(alpha=0.2),

    layers.Conv2DTranspose(512, kernel_size=4, strides=2, padding="same"),
    layers.BatchNormalization(),
    layers.LeakyReLU(alpha=0.2),

    layers.Conv2D(3, kernel_size=5, padding="same", activation="tanh")
], name="generator")
```



Tanh

- Tanh function has a range of values between (-1,1) and has a S-shaped form
- Benefit of tanh function is to map the -ve inputs as firmly negative.
- It is primarily used in classification of two classes.



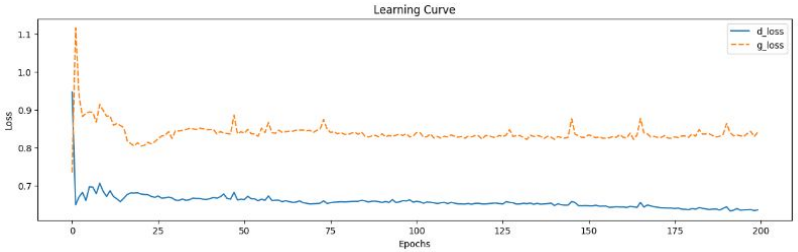
Training a GAN Model

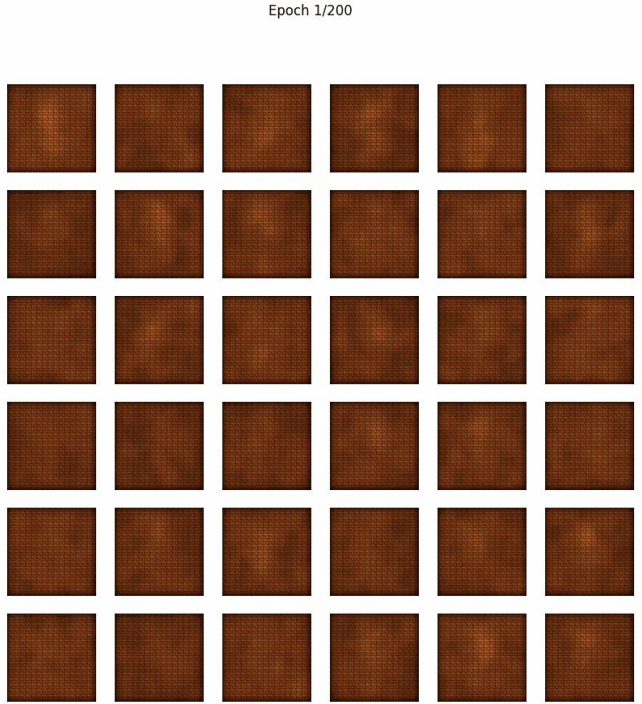
- Sample random points in the latent space.
- Decode the random latent vectors into fake images using the generator.
- Combine the generated images with real images from the dataset.
- Create labels for distinguishing between real and fake images.
- Add random noise to the labels (important trick for training stability).
- Train the discriminator using the combined images and labels.
- Sample random points in the latent space again.
- Assemble labels indicating all images are real (misleading labels).
- Train the generator using the misleading labels and generated images.
- Update metrics (loss) for both discriminator and generator.
- Repeat the training process for multiple epochs to optimize the GAN model.

Results

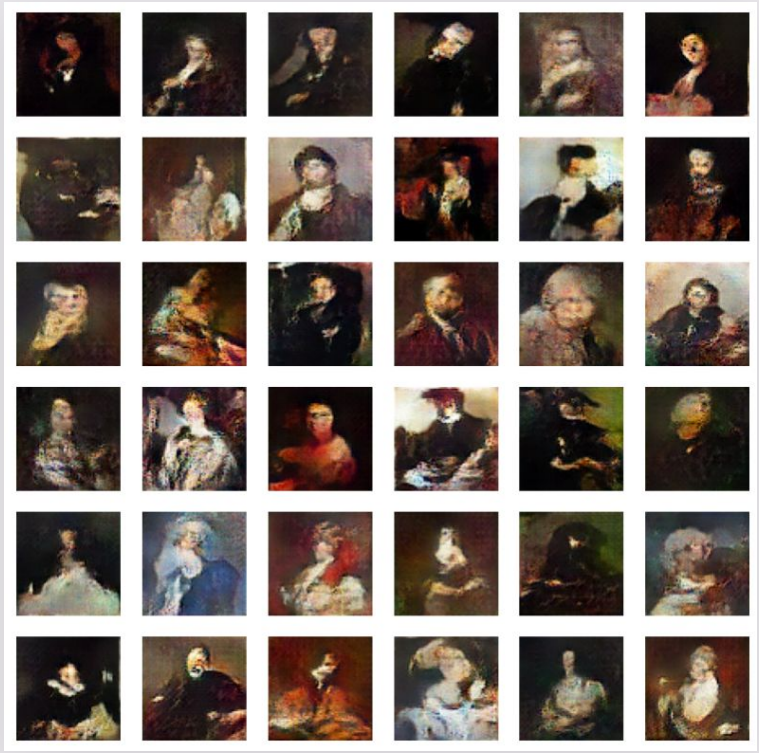
No of Epochs = 200
Batch Size = 32
Learning Rate = 0.0001
Optimizer = RMS
Activation Function = LeakyReLu
Loss Function = Binary Cross Entropy

Binary Cross Entropy / Loss Fn = $-\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\big(p(y_i)\big) + (1-y_i) \cdot \log\big(1-p(y_i)\big)$





Generated



Original

Conclusion

The presented GAN model demonstrates the capability to generate realistic portraits based on the provided dataset.

Through the adversarial training process, the generator learns to produce high-quality images that closely resemble portraits in the dataset.

The discriminator effectively distinguishes between real and synthetic images, contributing to the training stability and overall performance of the GAN model.

Despite challenges in training, the model shows promising results, indicating the potential of GANs in generating diverse and lifelike portraits.

Future Scope

Explore advanced GAN architectures: Investigate novel GAN architectures, such as Progressive GANs or StyleGAN, to further enhance the quality and diversity of generated portraits.

Incorporate additional datasets: Expand the training dataset to include a wider variety of portrait styles, poses, and expressions, thereby enriching the diversity of generated images.

Fine-tune hyperparameters: Experiment with different hyperparameters, loss functions, and optimization techniques to improve the training efficiency and convergence speed of the GAN model.

Apply style transfer techniques: Investigate the integration of style transfer techniques to enable users to control specific attributes or artistic styles of generated portraits.

Real-time applications: Explore the potential of deploying the trained GAN model in real-time applications, such as interactive portrait generation tools or virtual character customization systems.

Thank You!