**Title** - Image Generation with Generative Adversarial Networks(GANs)

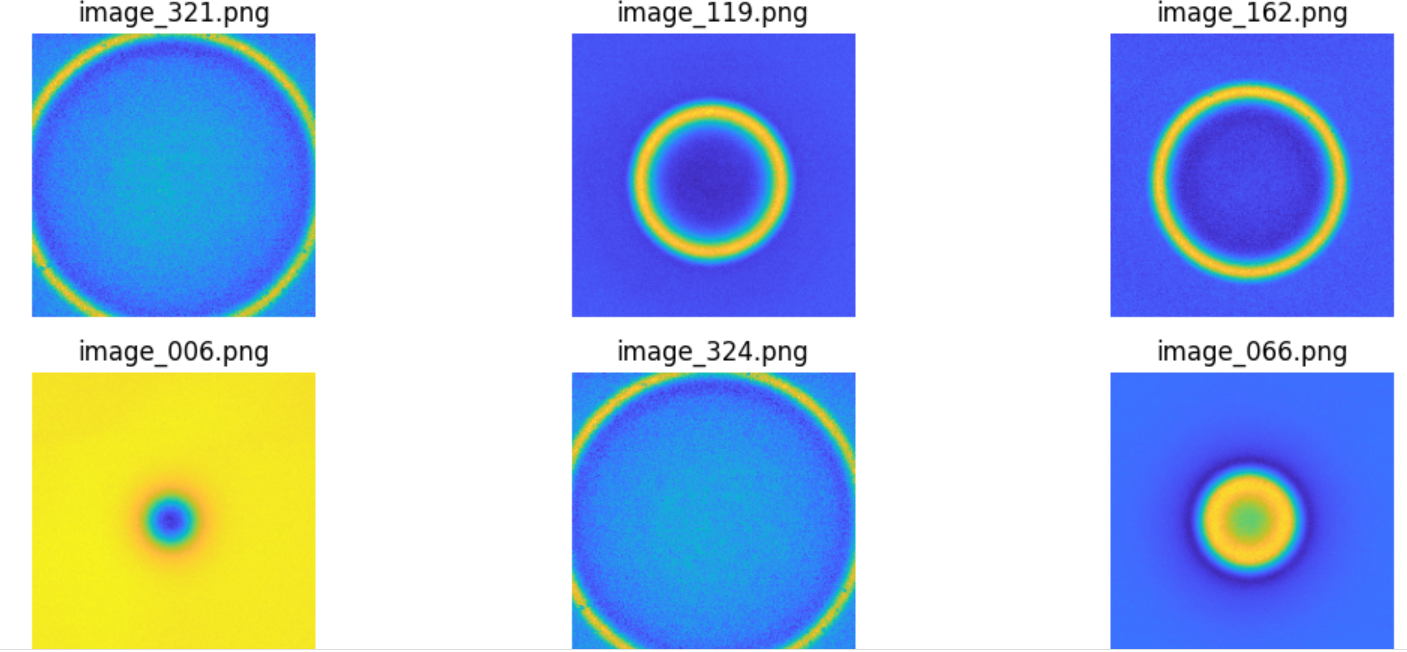
**Name** – Jayesh Sonawane (BTech 3rd Year, IIT Guwahati, India)

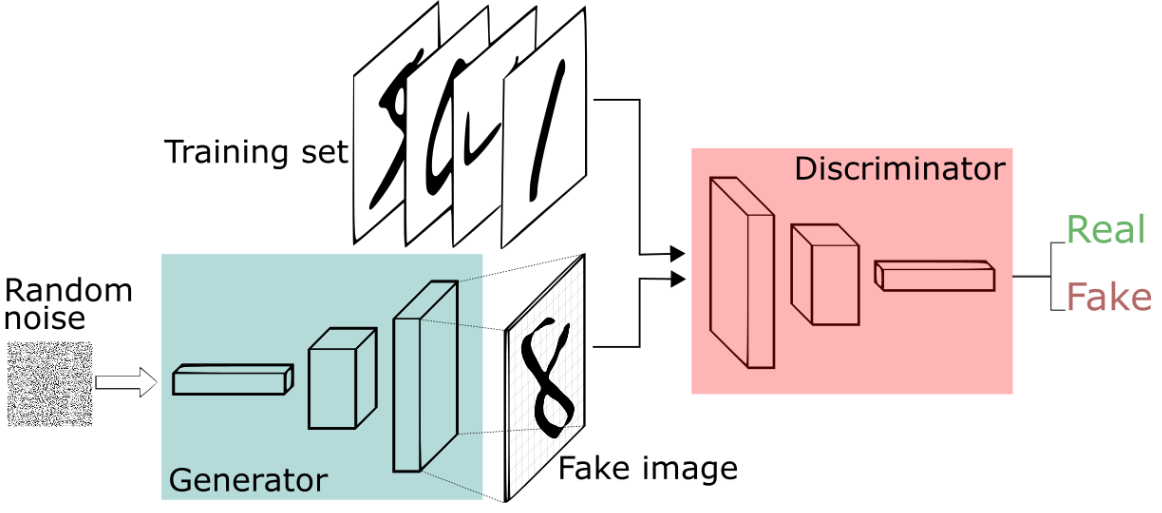
1. **Introduction**-
   1. **Background- Generative Adversarial Networks (GANs)** are a type of neural network that uses unsupervised learning to create artificial data that resembles real data. GANs are machine learning models that can generate realistic images, videos, and voice outputs.

GANs are made up of two neural networks, a discriminator and a generator. The networks compete against each other to generate new data instances. For example, GANs can create images that look like photographs of human faces, even though the faces don't belong to any real person.

* 1. **Project Objective**- The provided dataset consists of 365 Images, and the objective is to expand this dataset by creating synthetic images that seamlessly blend with the existing samples, ensuring that the generated images exhibit realistic features, textures, and structures similar to those present in the original dataset. The goal is to improve the robustness and diversity of the training data for downstream computer vision tasks.

1. **Methodology**-
   1. **Dataset Description**- The dataset contains 365 Images as follows:



-

1.2 **Training Process**-

This process involves two key neural network models: the **Generator** and the **Discriminator**, implemented as classes in **PyTorch**. The training process is iterative, taking place over a predefined number of epochs.

**Model** **Architecture**:

**Generator**: This neural network aims to generate new images from random noise. It utilizes a series of transposed convolutional layers (ConvTranspose2d) to upscale the input noise to an image. The generator progressively increases the spatial dimension while reducing the depth, ending with a Tanh activation function to output an image.

**Discriminator**: This network's goal is to distinguish between real and fake (generated) images. It consists of convolutional layers (Conv2d) that progressively downsample the input image, concluding with a sigmoid activation to output a probability indicating if the input image is real or fake.

**Training** **Dynamics**:

**Initialization**: Both networks are moved to the appropriate device (**CPU** or **GPU**).

Batch\_size: **2**

Learning rate for Generator: **0.0005**

Learning rate for Discriminator: **0.00005**

Epochs: **100**

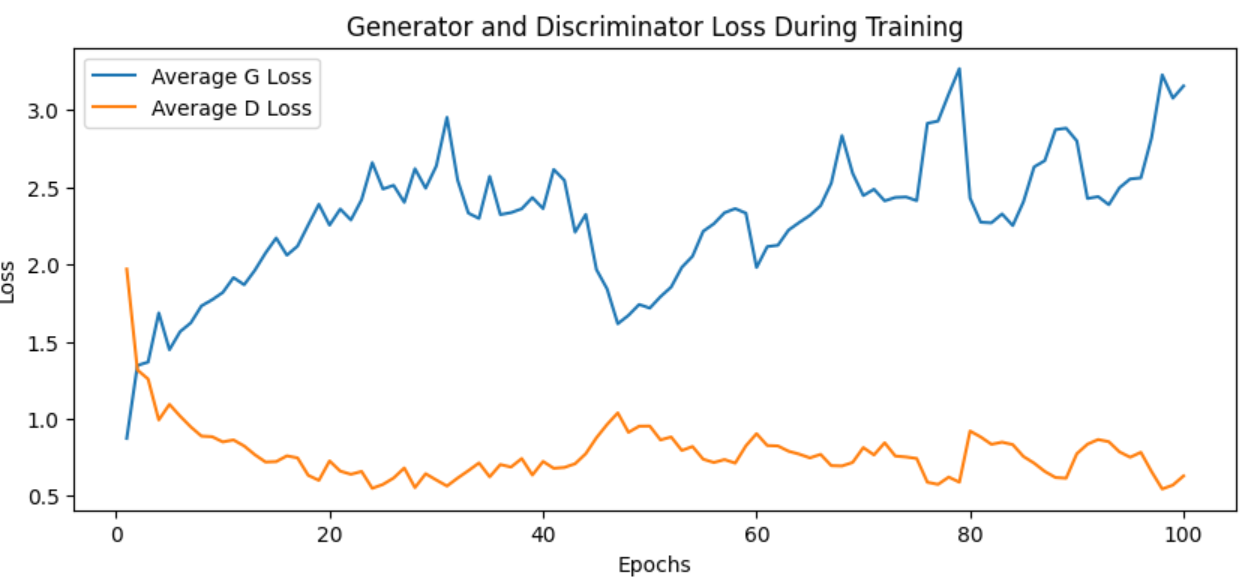
Loss function: **Binary Cross Entropy(BCELoss)**

Optimizer: **Adam** optimizer with beta1: **0.5**, beta2: **0.999**

**Discriminator Update**: For each batch, the discriminator is trained twice: first with a batch of real images and then with a batch of fake images generated by the generator. The loss is calculated using Binary Cross Entropy (BCELoss) to measure how well the discriminator can distinguish real and fake images.

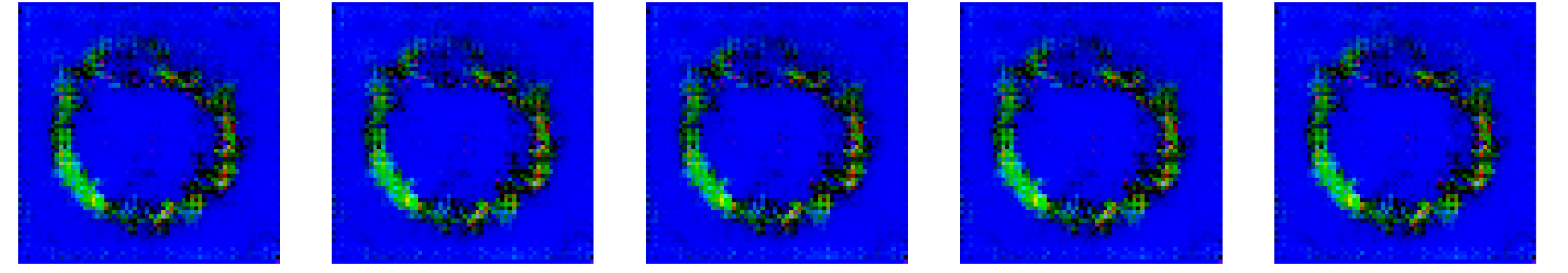
**Generator Update**: The generator is trained to fool the discriminator. It generates a batch of fake images, which are then passed to the discriminator. The generator's loss is calculated based on how well it tricks the discriminator into classifying fake images as real.

**Loss Tracking and Visualization**:

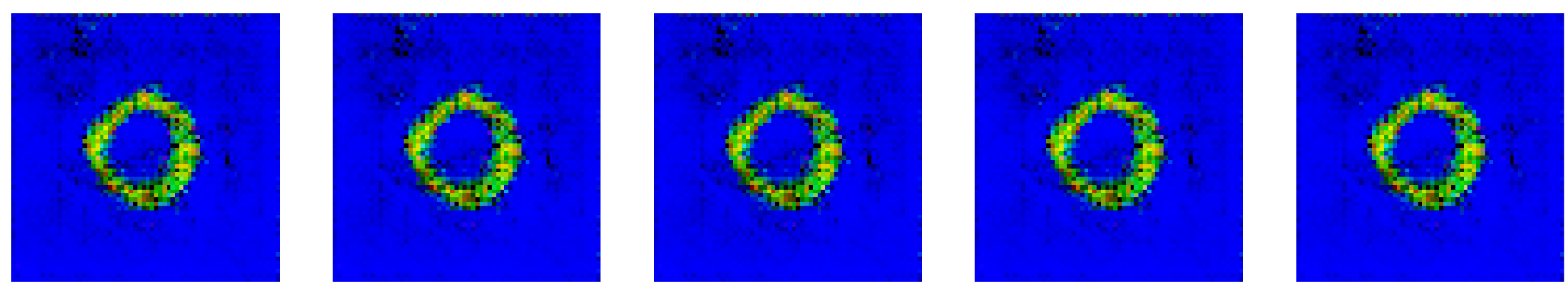


**Progress of Image Generation with Epochs**:

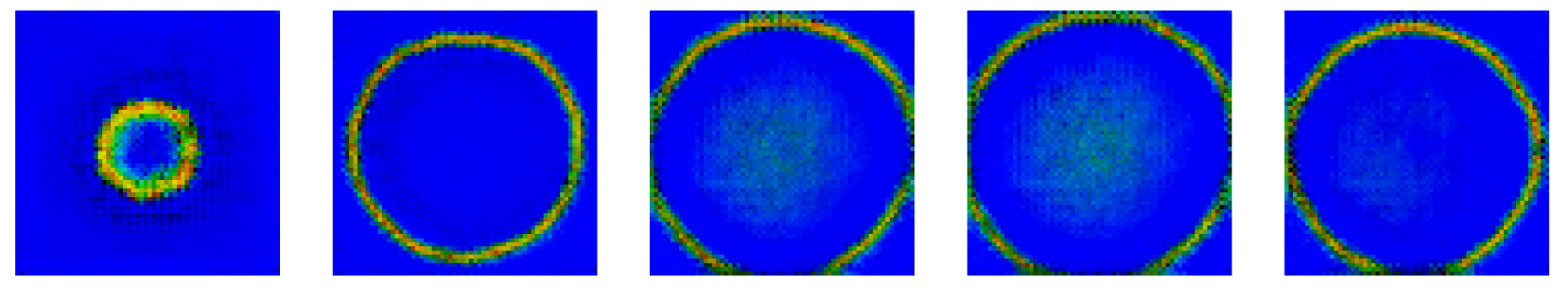
Epoch 11–



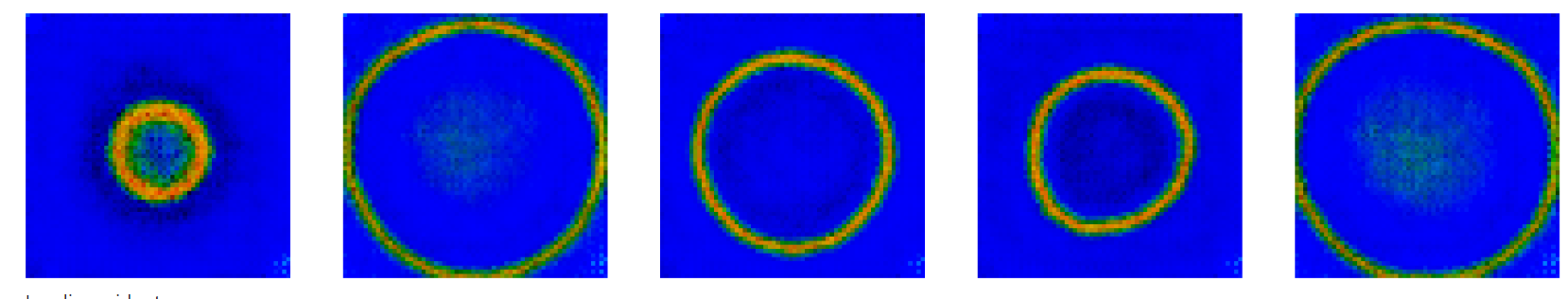
Epoch 31 –



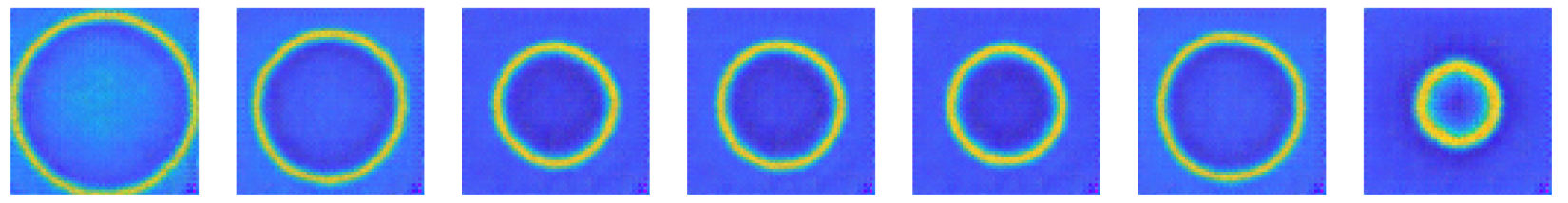
Epoch 51 –



Epoch 71 –



After training at 100 Epochs and unnormalizing the images –



**End of Training**:

After all epochs are completed, the training process ends, resulting in a trained generator capable of producing realistic images and a discriminator skilled in distinguishing between real and generated images.

1. **Results and Discussion**:

To improve the quality of generated images, images are sharpened using some basic **OpenCV** techniques.

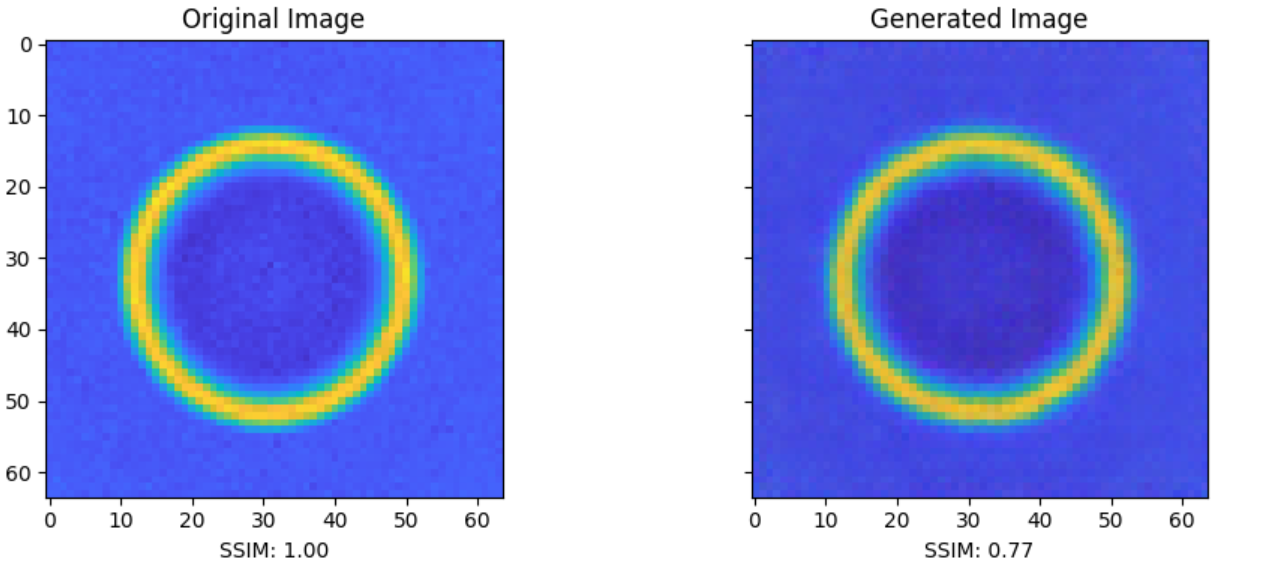
**Generated Images**:

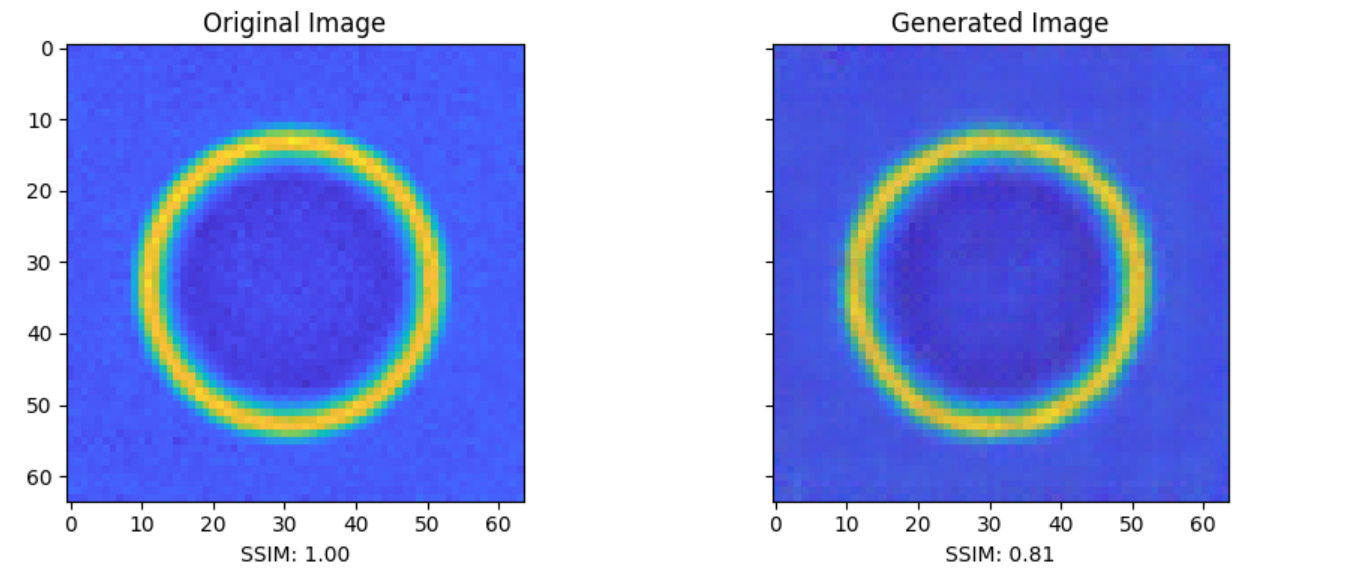


**Performance Metrics**:

Present performance metrics like loss curves, PSNR, SSIM, etc.

Average SSIM value on all generated images: **0.70**





Most of the images have an SSIM greater than **0.75** but some images are there whose quality is not that good have SSIM below **0.60**.

Average PSNR value on all generated images: **29.7**

**Scope of Improvement**: The quality of generated images can be further improved by introducing more images in the training dataset. The model is unable to generate some variants of images (for example, the images that are in yellow) due to less number of training images.

1. **Conclusion**: Implemented a GAN-based approach to augment the dataset, and successfully generated images that exhibit realistic features, textures, and structures similar to those present in the original dataset.