Human Face Generation with DCGAN

The main goal of this project is to explore and implement a DCGAN-based system capable of generating realistic human faces. To achieve this, the project focuses on the following key objectives:

1. Understand the Architecture of DCGAN
   Study how Generative Adversarial Networks work, with a focus on the deep convolutional variant (DCGAN), including the roles of the Generator and Discriminator.
2. Collect and Prepare a Suitable Dataset
   Use a dataset of real human face images (e.g., CelebA) and preprocess it to a format suitable for training the model (resizing, normalization, etc.).
3. Build and Train the DCGAN Model
   Implement a DCGAN using a deep learning framework (such as PyTorch or TensorFlow) and train it to generate high-quality face images.
4. Evaluate the Quality of Generated Faces
   Analyze the outputs using visual inspection and qualitative metrics to assess how realistic and diverse the generated faces are.
5. Overcome Common GAN Training Challenges
   Address potential issues like mode collapse, unstable training, or vanishing gradients to improve the performance of the model.
6. Document and Present Findings
   Summarize the process, results, challenges, and learnings in a clear and well-structured format for academic or practical presentation.

MODELS USED

In this project, a Deep Convolutional Generative Adversarial Network (DCGAN) was used to generate synthetic face images. DCGAN consists of two neural networks:

- The Generator, which takes random noise as input and produces fake images.
- The Discriminator, which distinguishes between real and fake images.

Both networks are trained together in a game-like setup where the generator tries to fool the discriminator. The model was trained on 64×64 resolution images to reduce GPU load, while still capturing key facial features. DCGAN was chosen for its simplicity and strong performance in image generation tasks.

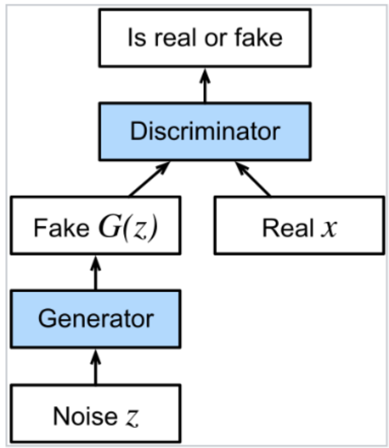


Figure. Illustration of how GANs work

DCGAN Loss Functions

The objective is based on the minimax game:

minGmaxDV(D,G)=$E_{x\sim pdata(x)}$[logD(x)]+$E_{z\sim pz(z)}$[log(1−D(G(z)))]

- D(x): Probability that input xx is real.
- G(z): Generated image from noise zz.
- The generator is trained to minimize the loss log(1−D(G(z)), or sometimes −log(D(G(z)) for better gradients.

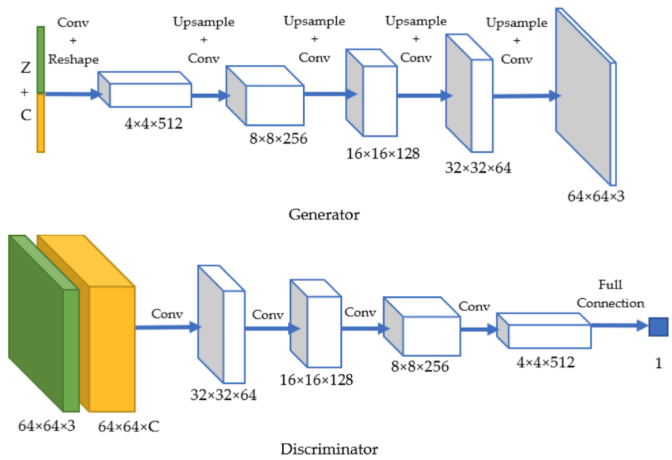
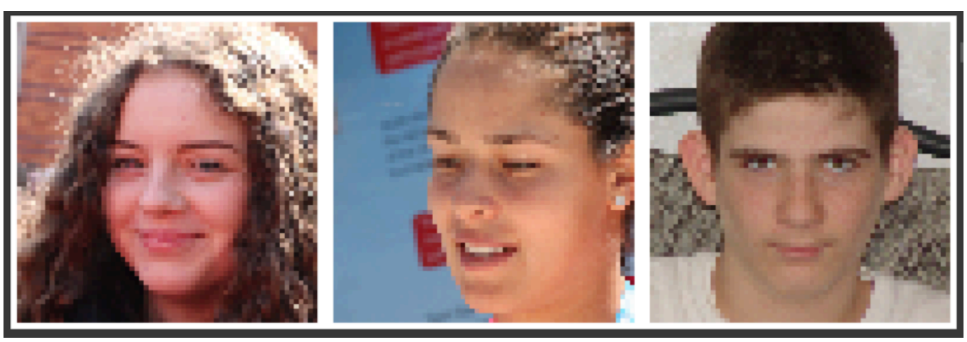


Figure.DCGAN Models

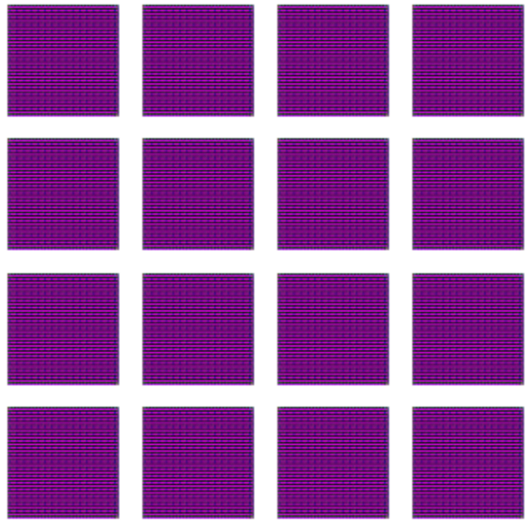
Dataset can be downloaded from
https://www.kaggle.com/datasets/matheuseduardo/flickr-faces-dataset-resized
This dataset contains the same images as the FFHQ dataset, downscaled to 256x256, 128x128, and 64x64 pixels, to make them easier to use in smaller generative models.
I have used the 64x64 one.

Sample Input Images:



Output:

Images at 1st epoch:



Images at 60th epoch: