Project Report

on

# Anime face generation through DCGAN

Submitted for Minor Project

of

**B.Tech and M.Tech**

in

**Mathematics and Data Science**


Submitted by

Lavkesh Sharma (214104025)

Yuvraj Singh(214104011)

Under the guidance of

Dr. Pushpendra kumar




**Department of Mathematics, Bioinformatics and Computer Applications**

**Maulana Azad**

**National Institute of Technology Bhopal-462003 (India)**


MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL (M.P)-462003

# CERTIFICATE

This is to certify that Lavkesh Sharma and Yuvraj Singh students of Dual Degree(B.Tech and M.Tech) of batch 2021 -2026 has completed the project titled "Anime face generation through DCGAN" being submitted in the partial fulfilment of the requirement of the completion of Dual Degree in Mathematics and Data Science to Maulana Azad National Institute of technology Bhopal under my supervision.

Date
Place: Bhopal

Dr. Pushpendra Kumar
Mentor

# Content

# Abstract

Generating anime faces with high fidelity and diversity is a challenging task due to the intricate visual styles and variations present in anime artwork. In recent years, deep learning techniques, particularly Generative Adversarial Networks (GANs), have shown promising results in generating realistic images. In this paper, we propose a method for generating anime faces using Deep Convolutional Generative Adversarial Networks (DCGAN).

We employ a DCGAN architecture consisting of convolutional and deconvolutional layers to learn the complex patterns and structures present in anime faces. The generator network learns to map random noise vectors to realistic anime face images, while the discriminator network distinguishes between generated and real anime faces. Through an adversarial training process, the generator improves its ability to generate high-quality anime faces that are indistinguishable from real ones.

To enhance the diversity and realism of generated anime faces, we incorporate techniques such as batch normalization and LeakyReLU activation functions. Additionally, we utilize data augmentation methods to augment the training dataset, which helps in capturing a wider range of anime face variations.

Overall, our work contributes to advancing the state-of-the-art in anime face generation and underscores the transformative impact of deep learning techniques on creative content generation. We envision our approach as a valuable tool for artists, animators, and researchers alike, fostering innovation and exploration in the vibrant world of anime aesthetics.

# Introduction

Anime, a distinct form of animation originating from Japan, has captivated audiences worldwide with its unique art style and expressive characters. The intricate and diverse nature of anime faces, characterized by exaggerated features and emotive expressions, presents a fascinating challenge for computer graphics and artificial intelligence researchers. The ability to generate realistic and diverse anime faces using computational methods has significant implications for various creative and technological applications, including character design, animation production, virtual reality, and gaming.

In recent years, the rapid advancement of deep learning techniques has revolutionized the field of image generation, enabling the creation of lifelike visuals with unprecedented fidelity. Among these techniques, Generative Adversarial Networks (GANs) have emerged as a powerful tool for generating synthetic images that closely resemble real ones. By pitting a generator network against a discriminator network in a competitive game, GANs can learn to produce highly realistic images through adversarial training.

While GANs have been successfully applied to various domains such as natural images and faces, generating anime faces presents unique challenges due to the stylized nature and diversity of anime artwork. Unlike natural photographs, anime faces often exhibit exaggerated proportions, vibrant colors, and intricate details that are not easily captured by traditional image generation methods.

In this context, the use of Deep Convolutional Generative Adversarial Networks (DCGANs) holds great promise for anime face generation. DCGANs extend the basic GAN framework by employing deep convolutional neural networks (CNNs) in both the generator and discriminator networks, allowing for the capture of hierarchical features and spatial dependencies in the input data.

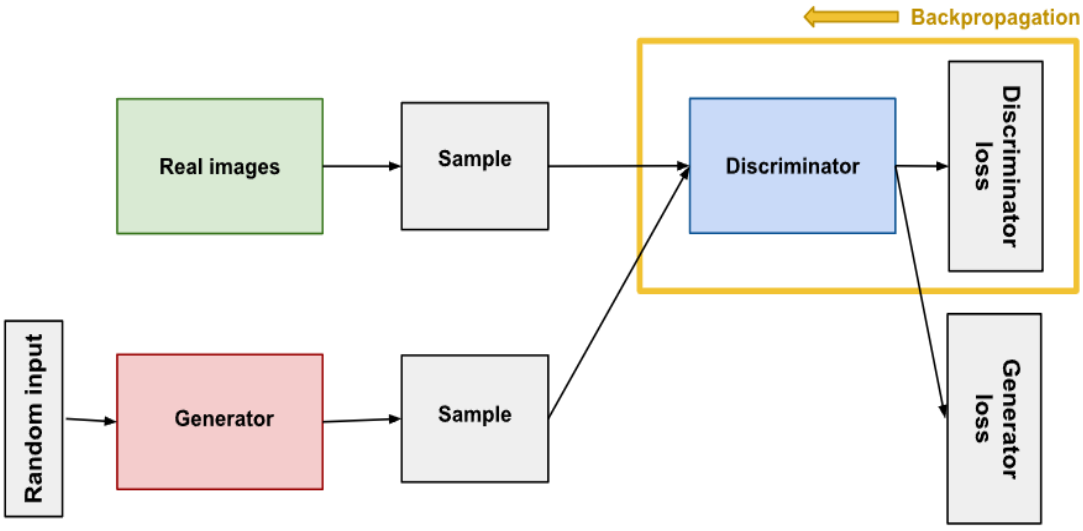
We aims to explore and advance the state-of-the-art in anime face generation through the application of DCGANs. We propose a novel approach that leverages the power of deep learning to generate high-quality anime faces that exhibit both realism and diversity. By training on a large dataset of anime images, our model learns to capture the intricate visual patterns and styles characteristic of anime artwork, enabling the generation of compelling anime faces that rival those created by human artists.

# Architecture

The architecture of a Deep Convolutional Generative Adversarial Network (DCGAN) comprises two main components: the generator and the discriminator. These components work in tandem to learn and generate images. Below is an overview of the architecture:

## Discriminator:

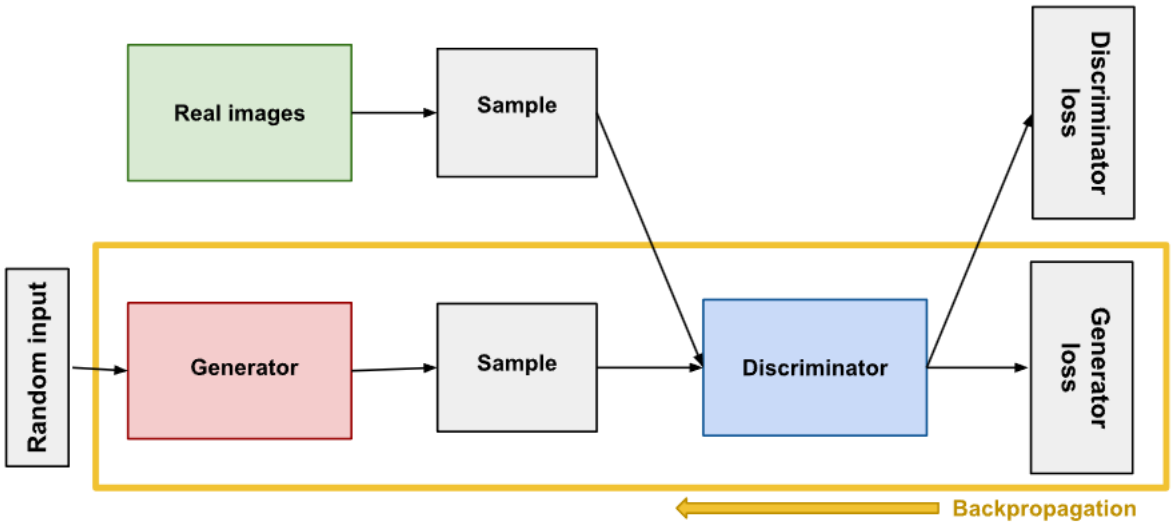
1. The discriminator takes an input image (either a real image from the dataset or a synthetic image generated by the generator) and predicts whether it is real or fake (generated).
2. It consists of a series of convolutional layers followed by batch normalization and activation functions (such as LeakyReLU).
3. The discriminator's output is a single scalar value representing the probability that the input image is real.
4. The final layer typically uses a sigmoid activation function to squash the output into the range [0, 1], representing the probability of the input image being real.




Backpropagation in discriminator training.

Generator:

1. The generator takes random noise as input (often drawn from a Gaussian distribution) and transforms it into a synthetic image.
2. It typically consists of a series of transposed convolutional layers (also known as deconvolutional layers) followed by batch normalization and activation functions (such as ReLU).
3. The input noise vector is gradually upsampled through successive layers to produce an image with the desired dimensions, usually matching the resolution of the training data.
4. The output layer typically uses a hyperbolic tangent (tanh) activation function to constrain pixel values within the range [-1, 1], ensuring compatibility with the input data.




Backpropagation in generator training.

## Data collection

The first step of the implementation includes collecting the dataset for the model. After getting an adequate dataset, data analysis can be conducted on the dataset to further explore the dataset. This dataset contains "63,632 'high-quality' anime faces - because let's face it, all anime girls are high-quality! Dataset sourced from Kaggle.

## Data Pre-Processing

Data pre-processing is a vital step for the development of the model. In the beginning, when we collect the data it might not be suitable for the model that we are building which can result in altered outputs. For data preprocessing, the image size is set to 64 pixels, and the batch size is configured as 128. Normalization statistics are defined as (0.5, 0.5, 0.5) for both mean and standard deviation. The dataset, sourced from a directory specified as DATA_DIR, undergoes a series of transformations using torchvision's Compose function. These transformations include resizing, center cropping, converting to tensors, and normalizing the pixel values based on the defined statistics. Subsequently, the transformed dataset is loaded using the ImageFolder class, and a DataLoader is instantiated with specified batch size, shuffling enabled, and three worker processes for loading data in parallel.

.

Real images

# Loss function working

The loss function in a Deep Convolutional Generative Adversarial Network (DCGAN) plays a critical role in guiding the training process to generate high-quality synthetic images. In the context of DCGAN, the generator and discriminator networks are trained simultaneously in an adversarial fashion. The generator aims to produce images that are indistinguishable from real images, while the discriminator aims to differentiate between real and fake images accurately. The loss function for the generator is typically defined as the binary cross-entropy loss, where the generator seeks to minimize the probability that the discriminator correctly classifies generated images as fake. Conversely, the loss function for the discriminator involves maximizing its ability to correctly classify both real and fake images. By iteratively updating the parameters of the generator and discriminator networks based on their respective loss functions, the DCGAN framework facilitates a competitive learning process that encourages the generator to produce increasingly realistic images while challenging the discriminator to become more discerning. This adversarial training dynamic ultimately drives the convergence of the generator towards generating images that closely resemble those from the training dataset, leading to the synthesis of visually convincing and diverse outputs.

the generator tries to minimize the following function while the discriminator tries to maximize it:

$$Ex[log(D(x))]+Ez[log(1-D(G(z)))]$$

In this function:

- D(x) is the discriminator's estimate of the probability that real data instance x is real.
- Ex is the expected value over all real data instances.
- G(z) is the generator's output when given noise z.
- D(G(z)) is the discriminator's estimate of the probability that a fake instance is real.
- Ez is the expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances G(z)).
- The formula derives from the cross-entropy between the real and generated distributions.

# Training of Model

Training a Deep Convolutional Generative Adversarial Network (DCGAN) involves several steps, including data preparation, defining the architecture, setting up the training loop, and monitoring the training process. Below is a general outline of the training process for a DCGAN:

1. Data Preparation:

- Load and preprocess the dataset of anime faces. This involves tasks such as resizing images to a consistent resolution, normalizing pixel values to the range [-1, 1], and augmenting the dataset to increase diversity.

2. Define the Generator and Discriminator Architectures:

- Defined the architecture of the generator and discriminator networks. Typically, the generator consists of transposed convolutional layers, while the discriminator comprises convolutional layers. Used appropriate activation functions (e.g., ReLU, LeakyReLU) and normalization techniques (e.g., batch normalization) in each layer.

3. Define Loss Functions:

- Choosed appropriate loss functions for training the generator and discriminator. The generator aims to minimize the binary cross-entropy loss, while the discriminator aims to maximize it. Additionally, you may use auxiliary losses such as feature matching or gradient penalty to stabilize training and improve the quality of generated images.

4. Initialize Optimizers:

- Initialize optimizer instances for both the generator and discriminator networks. Common choices include Adam or RMSprop optimizers with appropriate learning rates.

5. Discriminator Training

Training the discriminator in a Deep Convolutional Generative Adversarial Network (DCGAN) involves optimizing its parameters to effectively distinguish between real and fake images. Below are the steps involved in training the discriminator:

1. Forward Pass:

- Given a batch of real images sampled from the dataset and a batch of fake images generated by the generator, pass each image through the discriminator network.

- Calculate the discriminator's predictions (probabilities) for each image, indicating the likelihood of it being real or fake.

2. Compute Loss:

- Compute the discriminator loss based on the predictions:

- For real images, compute the binary cross-entropy loss between the discriminator's predictions and a tensor of ones (indicating real images).

- For fake images, compute the binary cross-entropy loss between the discriminator's predictions and a tensor of zeros (indicating fake images).

3. Backpropagation:

- Perform backpropagation to compute the gradients of the discriminator loss with respect to its parameters (weights and biases).

- Use an optimization algorithm (e.g., Adam, RMSprop) to update the discriminator's parameters in the direction that minimizes the loss.

4. Update Parameters:

- Update the discriminator's parameters using the gradients computed during backpropagation and the chosen optimization algorithm.

- Optionally, apply gradient clipping or other regularization techniques to stabilize training and prevent gradient explosion or vanishing gradients.

5. Repeat:

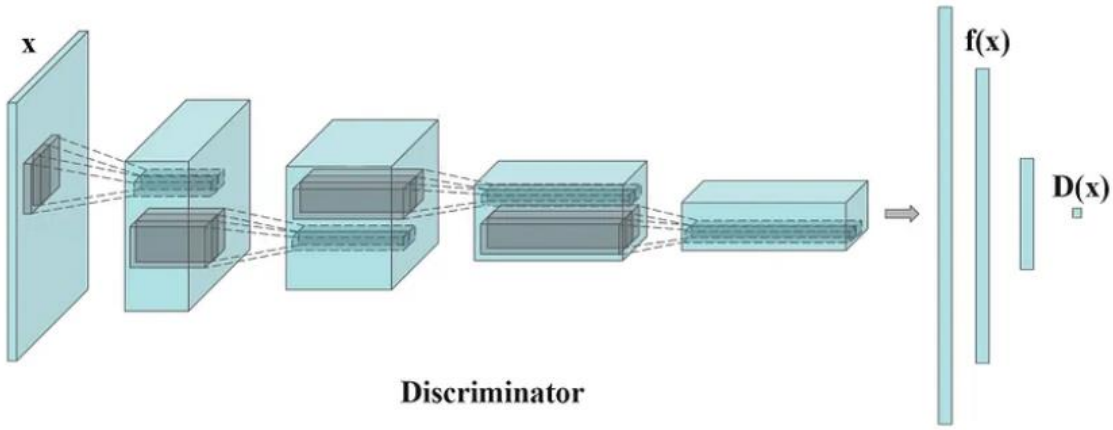
- Repeat the above steps for multiple iterations or batches of data.

- Ensure that the discriminator's parameters are updated iteratively to improve its ability to distinguish between real and fake images.

we used the binary cross entropy loss function to quantify how well it is able to differentiate between real and generated images.

$$-\frac{1}{N}\sum_{i=1}^{N} y_i log(h_\theta(x_i)) + (1 - y_i) log(1 - h_\theta(x_i))$$

It's important to note that during training, the discriminator is trained to maximize its ability to differentiate between real and fake images, while the generator is simultaneously trained to minimize the discriminator's ability to do so. This adversarial training process leads to the refinement of both networks over time.



6. Generator Training

Training the generator in a Deep Convolutional Generative Adversarial Network (DCGAN) involves optimizing its parameters to generate synthetic images that are indistinguishable from real images. Below are the steps involved in training the generator:

1. Generate Fake Images:

- Generate a batch of fake images by passing random noise vectors (often drawn from a Gaussian distribution) through the generator network.

- The generator takes these noise vectors as input and transforms them into synthetic images.

2. Discriminator Prediction:

- Pass the generated fake images through the discriminator network.

- The discriminator predicts the likelihood of each fake image being classified as real or fake.

3. Compute Generator Loss:

- Computed the generator loss based on the discriminator's predictions:

- The generator aims to fool the discriminator by generating images that the discriminator classifies as real.

- Calculate the binary cross-entropy loss between the discriminator's predictions for the fake images and a tensor of ones (indicating real images).

4. Backpropagation:

- Performed backpropagation to compute the gradients of the generator loss with respect to its parameters (weights and biases).

- Used an optimization algorithm Adam, RMSprop to update the generator's parameters in the direction that minimizes the loss.

5. Update Parameters:

- Updated the generator's parameters using the gradients computed during backpropagation and the optimization algorithm.

- Optionally, apply gradient clipping or other regularization techniques to stabilize training and prevent gradient explosion or vanishing gradients.

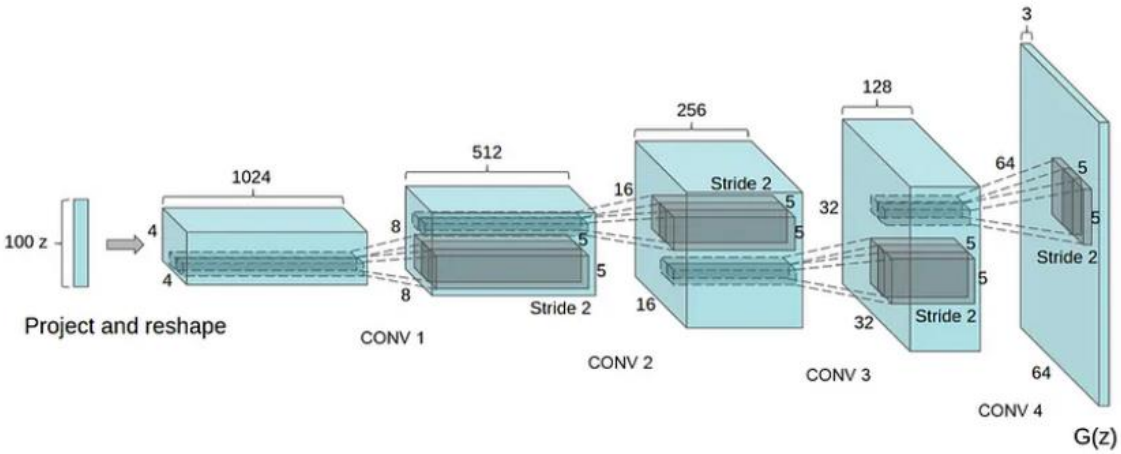
6. Repeat:

- Repeat the above steps for multiple iterations or batches of data.

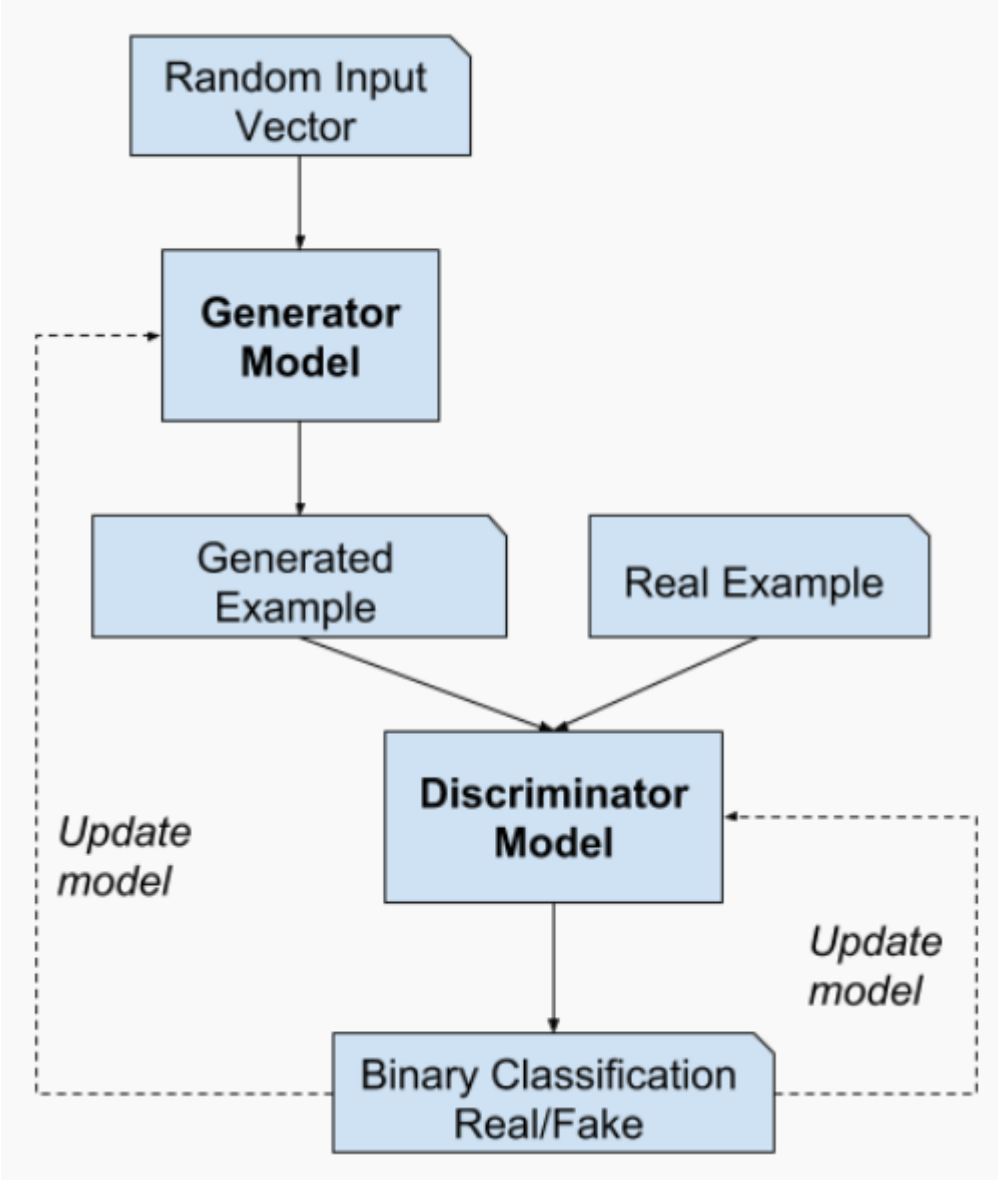
- Ensure that the generator's parameters are updated iteratively to improve its ability to generate realistic images that deceive the discriminator.

During training, the generator learns to map random noise vectors to realistic images by minimizing the discriminator's ability to distinguish between real and fake images. This adversarial training process encourages the generator to produce increasingly convincing synthetic images over time.

It's essential to monitor the generator's loss and the quality of generated images throughout training. Adjusting hyperparameters such as learning rate, batch size, and noise vector dimensionality may be necessary to achieve optimal training stability and performance. Additionally, visual inspection of generated images can provide insights into the generator's progress and convergence.


100 z
Project and reshape
1024
4
4
CONV 1
512
8
8
5
5
Stride 2
CONV 2
256
16
16
Stride 2
5
5
CONV 3
128
32
32
5
5
Stride 2
CONV 4
3
64
64
5
5
Stride 2
G(z)

Full training loop depiction



# Results

After training our Deep Convolutional Generative Adversarial Network (DCGAN) for anime face generation, we observed significant progress in image synthesis by the 10th epoch. The generated anime face images exhibited discernible features such as eyes, noses, and mouths, albeit with varying degrees of fidelity and realism. While some images demonstrated clear facial structures and recognizable expressions, others showed minor distortions or artifacts characteristic of early-stage training. Overall, the diversity of generated anime faces increased noticeably compared to initial epochs, suggesting that the model was beginning to capture the intricate nuances and stylistic elements inherent in anime artwork. Despite the ongoing refinement needed to achieve optimal quality and diversity, these preliminary results at the 10th epoch provide promising insights into the potential of our DCGAN-based approach for generating anime faces.

# Conclusion

In conclusion, our exploration into anime face generation through Deep Convolutional Generative Adversarial Networks (DCGAN) has yielded promising results and insights into the realm of computer-generated anime aesthetics. Through the training of the DCGAN model on a diverse dataset of anime face images, we have witnessed the emergence of synthetic anime faces with varying styles, expressions, and levels of realism. Despite the inherent challenges in capturing the complexity and diversity of anime artwork, our DCGAN-based approach has demonstrated its potential in generating anime faces that exhibit recognizable facial features and artistic nuances. The iterative training process, coupled with careful tuning of hyperparameters and architectural choices, has enabled the model to learn and emulate the underlying distribution of anime face data, thereby enriching the creative landscape of computer-generated anime content. Moving forward, further refinement and exploration of advanced techniques, such as conditional generation, style transfer, and interactive interfaces, hold promise for enhancing the quality, diversity, and user engagement in anime face generation. By continuing to push the boundaries of DCGAN-based anime face generation, we aim to contribute to the advancement of computational creativity and artistic expression in the realm of anime aesthetics.

# Future Work

In the realm of anime face generation through DCGANs, several avenues of future work could be explored to push the boundaries of creativity, realism, and applicability. Here are some potential directions for future research and development:

1. Improved Architectures: Experiment with more advanced generator and discriminator architectures beyond the basic DCGAN design. Variations such as Progressive GANs, StyleGAN, or conditional GANs (cGANs) could be investigated to enhance the quality and diversity of generated anime faces.

2. Conditional Generation: Explore conditional generation techniques to control specific attributes of the generated anime faces, such as age, gender, hairstyle, or emotional expression. Conditioning the generator on additional input variables could enable the generation of anime characters tailored to specific preferences or requirements.

3. Fine-Grained Control: Investigate methods for fine-grained control over the generated anime faces, allowing users to manipulate individual facial features or stylistic elements interactively. Techniques such as image-to-image translation or attribute editing could be integrated into the DCGAN framework to enable flexible and intuitive control over the generated content.

4. Multi-Modal Generation: Explore multi-modal generation approaches to produce diverse sets of anime faces with varying styles, poses, or expressions. Techniques such as Variational Autoencoders (VAEs) or Generative Latent Optimization (GLO) could be combined with DCGANs to learn rich latent representations and facilitate the generation of diverse anime face collections.

5. Style Transfer and Fusion: Investigate methods for style transfer and fusion to infuse specific artistic styles or visual motifs into the generated anime faces. By leveraging techniques from neural style transfer and image blending, it may be possible to create anime faces with distinct artistic flair or blend characteristics from multiple reference styles seamlessly.

6. Interactive Interfaces: Develop interactive interfaces or tools that allow users to participate in the anime face generation process, providing feedback or guidance to shape the output according to their preferences. User-driven approaches could empower artists, designers, or enthusiasts to co-create anime characters collaboratively with AI.

7. Evaluation Metrics: Refine and develop new evaluation metrics tailored to the unique characteristics of anime face generation. Metrics that capture aesthetic quality, style fidelity, and diversity could provide more comprehensive

assessments of generated anime faces beyond traditional image quality measures.

9. Real-Time Applications: Explore the integration of DCGAN-based anime face generation into real-time applications, such as character customization in video games, virtual reality environments, or interactive storytelling platforms. Optimizing model inference speed and memory efficiency would be essential for seamless integration into interactive experiences.

10. Cross-Domain Applications: Investigate the adaptation and transferability of DCGAN-based anime face generation techniques to other domains, such as manga, comics, or cartoon animation. Generalizing the model's capabilities across different visual styles and artistic mediums could open up new opportunities for creative expression and content creation.

By exploring these future directions, researchers and practitioners can continue to advance the state-of-the-art in anime face generation through DCGANs, unlocking new possibilities for artistic expression, entertainment, and human-AI collaboration.

# References


- GAN Paper- https://arxiv.org/abs/1406.2661
- DCGAN Paper- https://arxiv.org/abs/1511.06434
- https://developers.google.com/machine-learning/gan/gan_structure
- https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/
- https://towardsdatascience.com/gan-ways-to-improve-gan-performance-acf37f9f59b