# Image Generation from Generative Adversarial Networks

**Ahmed Albishri**
*School of Computing and Engineering*
*University of Missouri Kansas City*
Kansas City, MO, US
aa8w2@umsystem.edu

**Saeed Alqarni**
*School of Computing and Engineering*
*University of Missouri Kansas City*
Kansas City, MO, US
saacfb@umsystem.edu

**Vinay Reddy Kalluri**
*School of Computing and Engineering*
*University of Missouri Kansas City*
Kansas City, MO, US
vk9ry@umsystem.edu

**Gayathri Garikapati**
*School of Computing and Engineering*
*University of Missouri Kansas City*
Kansas City, MO, US
ggr9d@umsystem.edu

**Karthik Yanagandula**
*School of Computing and Engineering*
*University of Missouri Kansas City*
Kansas City, MO, US
kyb8k@umsystem.edu

**Muni Kanaka Sri Shalini Chintam**
*School of Computing and Engineering*
*University of Missouri Kansas City*
Kansas City, MO, US
mcnhy@umsystem.edu

***Abstract*— Image generation is a critical step for Artificial Intelligence technologies in understanding image properties at the pixel level concerning context and environment. Generative Adversarial networks along with transformers played a crucial role in generating the image content with specified parameters. The main idea presented in this paper is to develop deep learning models to take the celebrity dataset related to face images and generate similar images related to technically fake faces. Two models are designed to handle the solution for deep face image generation and text to image using VQ-GAN and CLIP. Realistic than the original images with and without context. Taking input from the user like image type and context related to the image and generating the images at a pixel level. Both are different models as they need separate training and development. These generated images are used in a wide variety of multi-media, Non-Fungible Token (NFT) developers, UI/UX designers, developers across the industries of animation and internet companies, and research groups for generating data.**

***Keywords*— *Generative Adversarial Networks, VQ-GAN, CLIP, NFT, UI/UX, Image Generation, Deep Learning, Artificial Intelligence.***

## I. Introduction

By merely adding a tiny amount of noise to the original image data, most typical neural networks like CNN can be easily misled into misclassifying objects. Without classification of the objects, it will be tough to re-generate the same image after the addition of noise. Surprisingly, the model's confidence in the inaccurate prediction and generation is higher after noise is introduced than when it predicts correctly. The reason for this problem is that most machine learning and deep learning models only learn from a small amount of data, which is a big disadvantage because overfitting occurs. The mapping between the input and output is also roughly linear. Although the boundaries between the various feature classes are linear, they are made up of a lot of linearities, and even a small deviation in a point in the feature space might cause a change in the feature class. A similar issue is present in Image generation with GANs. Generating an image at a pixel level with and without context with a smaller number of training records will not result in generating a quality image. This paper aims to address the problems in generating an image and the application to generate the images.

## II. Problem Statement

To create images like art from generative adversarial networks by training a deep learning model with a celebrity dataset without a context on defined hardware to produce the most realistic image possible in the stipulated period at commercial grade is a challenging task. Another model uses VQ-GAN and CLIP to produce images from the context provided by the user with various keywords. Image generation is a collection of approaches aiming at creating the most realistic image feasible given the limitations of available computer hardware, time, financing, and skill set. Not long ago, generative arts and NFT were all the rage. With AI-powered Generative Adversarial Networks, more accurate images can be generated. Unsupervised learning with GANs can generate images at scale. This will be achieved with a lot of training on popular image celebrity datasets and creating generative models.

## III. Related Work

Many applications attempted to solve the image generation using different techniques by static image hosting, label-based image extraction, and Deep image generator. All are using pre-trained machine learning algorithms for labeling the existing images and whenever a user requests a specific keyword. They are fetched based on the labels attached to the images. Some of the keywords used by the users make the system configuration fails to recognize the correct label and fetches the wrong image concerning the context. This is tested in the existing images.

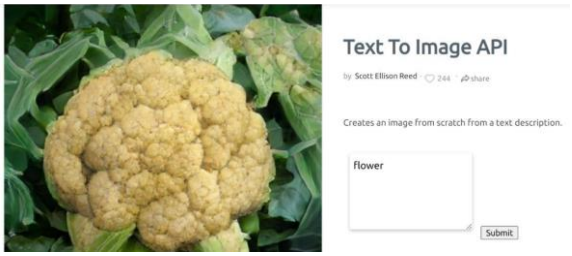



Fig. 1. Deep AI Text to Image API Failed Response to simple keyword

## IV. Solutions

*A. Deep Fake Face Image Generator using DeepImagify-GAN.*

*B. Context-based Image Generator using VQ-GAN and CLIP.*

Both solutions solve the problem with and without context-based image generation at pixel levels.

## V. Dataset

CelebFaces Attributes (CelebA) Dataset. Face recognition is a major component of computer vision and deep learning, with applications ranging from login into your phone with your face to looking through surveillance photographs for a specific suspect.

This dataset is ideal for training and testing face detection models, especially for identifying facial characteristics like people with brown hair, smiling, or wearing spectacles. Images span a wide range of stance variations, backdrop clutter, and a diverse range of people, with a significant number of images and detailed annotations.

MMLAB Researchers at the Chinese University of Hong Kong, originally gathered this information.

This dataset is used to understand the face characteristics and generate the fake image near to the original by adding some noise to the original one that will be attempted in this application.

A total of 202,599 celebrity profile pictures taken were identified. There are 10,177 unique identities, but no names are provided. 5 feature locations, 40 quantitative characteristic labels per image.

Cropped and aligned photos of everyone's face.


A short glimpse of the sample images from the dataset for understanding the image properties.

Fig. 2. Sample Images from CeleFaces Attributes (CelebA) Dataset

## VI. Preprocessing

For preprocessing the CelebFaces Attributes (CelebA) Dataset, Google Colab is used to develop the model. Data loaded to the drive and mounted to Google Colab. As the images are available in compressed formatted. Loaded the images after extracting them to a temporary folder and re-defined the properties of the image for training purposes.

The properties and preprocessing are as follows: Image count is locked at 10,000 for training as the GPU in Google Colab will not support the entire 200,000 images for training. The original image height and width are 208 * 178 * 3 dimensions. The difference between the dimensions is calculated by subtracting the original height from the original width and performing floor division on the difference. The height and width of the images are adjusted to 128 * 128 * 3 dimensions cropping based on the floor division result. All images are converted to UINT8 format from the NumPy library which is an 8-bit unassigned integer ranging from 0-255 decimals values.

In the final step, all the images using the NumPy array are divided by the float value 255.0 for converting to the least possible value for mathematical calculations.

Dividing by 255 expresses a 0-1 representation because 255 is the maximum value. Because each channel (Red, Green, and Blue) is 8 bits long, they are each restricted to 256 characters, in this case, 255 because 0 is included. When using floating point values, systems commonly use values between 0 and 1.

For VQ-GAN and CLIP model, it requires no pre-processing from the model perspective as it is more of transfer learning without datasets. The application will be using the pre-stored weights in deriving the images from user input.

## VII. SOLUTION

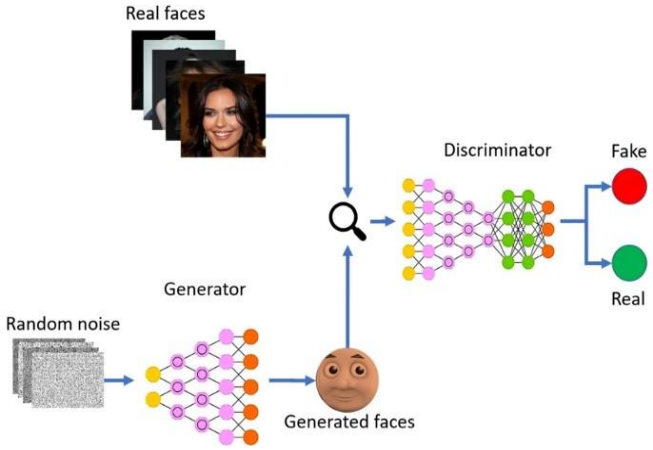
### A. *Deep Fake Face Image Generator using DeepImagify-GAN.*

To develop Deep Fake Face Image Generation, the system is trained to generate images based on the training from celebrities' image data set using Generative Adversarial Networks. A custom build GAN is used to do the task. GAN is named *Deep-Imagify GAN*. It is built on encoder-decoder-based architecture.

The images generated by the system can be used by various users like:

- Designers
- UI/UX Designer
- Artists
- NFT Developers
- Advertisement Agencies

In a nutshell, we'll tell the generator to make faces without providing any extra information. Simultaneously, we'll provide the discriminator of the existing faces in the dataset and ask it to determine whether the images generated by the Generator are authentic. Initially, the Generator will produce poor photos that the Discriminator will immediately flag as wrong with a lower confidence score along with deviation.
As a result of the decreased deviation from the true photos, the Generator will learn to fool the Discriminator after receiving enough data from it. As a result, we'll have a really good generative model that can produce very realistic results.




Technically, by competing for two neural networks against each other, GANs learn a probability distribution of the dataset. The Generator generates new data instances, while the Discriminator assesses them for authenticity; that is, the discriminator determines whether each instance of data it examines corresponds to the actual training dataset or not. In the meantime, the generator creates new synthetic/fake images, which it sends to the discriminator. It does so in the hopes of being recognized as genuine, even though they are not. The fake image is created using the inverse of convolution, called transposed convolution, to create a 100-dimensional noise (uniform distribution between -1.0 and 1.0).

This model developed on a face image set can help understand the face properties at a pixel. This GAN will attempt to develop the fake image from the existing faces after specific compilation and training.

### B. *Context-based Image Generator using VQ-GAN and CLIP.*

The Deep Learning Community has gone crazy about text-to-image synthesis. After NFT grabbed the market by storm, AI-generated artworks and photographs received a lot of attention. All of this is achievable because of the VQGAN-CLIP. VQGAN's generative skills (Esser et al, 2021) and CLIP's discriminative capacity (Radford et al, 2021) are used to create stunning images of artwork. These are existing models which are combined as applications for the solution to the problem defined in the problem.

CLIP is for Contrastive Image-Language Pretraining, while VQGAN refers to Vector Quantized Generative Adversarial Network. The interaction between these two networks is referred to as VQGAN-CLIP. They're two different models that work together.

VQGAN generates the images, whereas CLIP assesses how well an image matches our text question. Our generator is guided by this interaction to produce more accurate images:

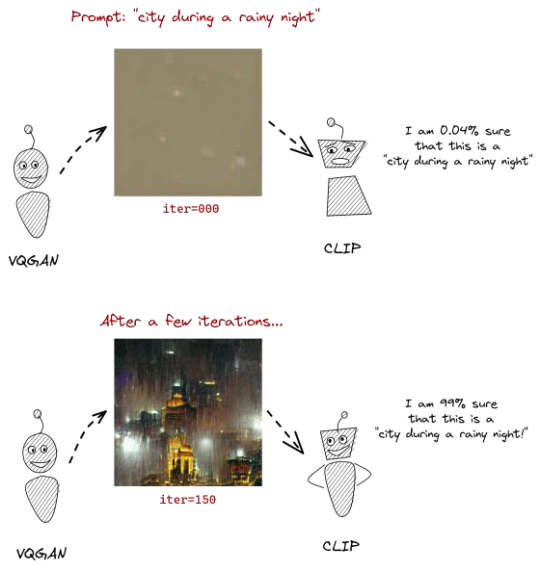



Fig. 3. VQ-GAN and CLIP In a Nutshell

VQ-GAN is capable of not only learning the features of the images but also the relationships between the visual parts of the image.

It is the combination of three components mainly on a high level:

- Convolutional neural networks: it is used to read all the images and take the load of the weights and bias by understanding the features among the images.
- A transformer network that learns long-range interactions from a sequence.
- A codebook for storing the visual information for generating the new images.

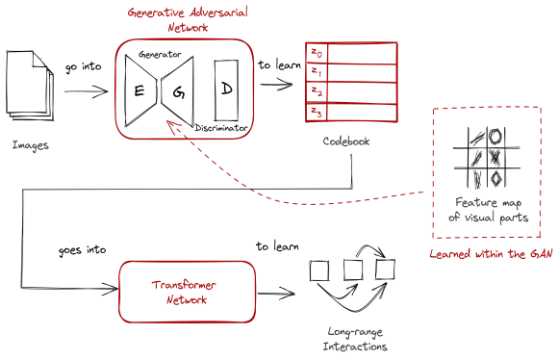



Fig. 4. VQ-GAN Architecture [10]

### CLIP: Contrastive Image-Language Pretraining

CLIP is another neural network that can determine how well a caption (or prompt) matches an image.

It is trained and developed by the Open AI community with ImageNet dataset and natural language processing tags images and allowed to predict the context with image.

CLIP draws on previous research in zero-shot transfer, natural language supervision, and multimodal learning.

The concept of zero-data learning has been around for a while, although it was largely investigated in computer vision until recently as a technique of generalizing to unseen object categories.

To facilitate generalization and transfer, researchers used natural language as a flexible prediction space. Richer Socher and co-authors [2] at Stanford demonstrated the concept in 2013 by training a model using CIFAR-10 to make predictions in a word vector embedding space and demonstrating that the model could predict two previously undiscovered classes.

DeVISE [3] scaled this strategy the following year, demonstrating that it was possible to fine-tune an ImageNet model so that it could properly forecast [5].

Using both VQ-GAN and CLIP together, artist-based images can be generated at scale.

## VIII. Model

*Solution [A]:* ***Deep Fake Face Image Generator using DeepImagify-GAN.***

After preprocessing the face image dataset, the model will contain a generator and a discriminator.

The generator reverses the situation: it is attempting to deceive the discriminator. There are eight convolutional layers in this network. First, we take our gen input and feed it into our first convolutional layer. Each convolutional layer first executes a convolution, followed by batch normalization and a leaky ReLU. The tanh activation function is then returned.

**Generator Model Summary:**

| | | | |
|---|---|---|---|
| input_1 InputLayer | input: output: | [(None, 32)] | [(None, 32)] |
| dense Dense | input: output: | (None, 32) | (None, 32768) |
| leaky_re_lu LeakyReLU | input: output: | (None, 32768) | (None, 32768) |
| reshape Reshape | input: output: | (None, 32768) | (None, 16, 16, 128) |
| conv2d Conv2D | input: output: | (None, 16, 16, 128) | (None, 16, 16, 256) |
| leaky_re_lu_1 LeakyReLU | input: output: | (None, 16, 16, 256) | (None, 16, 16, 256) |
| conv2d_transpose Conv2DTranspose | input: output: | (None, 16, 16, 256) | (None, 32, 32, 256) |
| leaky_re_lu_2 LeakyReLU | input: output: | (None, 32, 32, 256) | (None, 32, 32, 256) |
| conv2d_transpose_1 Conv2DTranspose | input: output: | (None, 32, 32, 256) | (None, 64, 64, 256) |
| leaky_re_lu_3 LeakyReLU | input: output: | (None, 64, 64, 256) | (None, 64, 64, 256) |
| conv2d_transpose_2 Conv2DTranspose | input: output: | (None, 64, 64, 256) | (None, 128, 128, 256) |
| leaky_re_lu_4 LeakyReLU | input: output: | (None, 128, 128, 256) | (None, 128, 128, 256) |
| conv2d_1 Conv2D | input: output: | (None, 128, 128, 256) | (None, 128, 128, 512) |
| leaky_re_lu_5 LeakyReLU | input: output: | (None, 128, 128, 512) | (None, 128, 128, 512) |
| conv2d_2 Conv2D | input: output: | (None, 128, 128, 512) | (None, 128, 128, 512) |
| leaky_re_lu_6 LeakyReLU | input: output: | (None, 128, 128, 512) | (None, 128, 128, 512) |
| conv2d_3 Conv2D | input: output: | (None, 128, 128, 512) | (None, 128, 128, 3) |

Fig 5: Generator Model Summary

The discriminator network, like the generator, is made up of convolutional layers. Convolution will be applied to each layer of the network, followed by batch normalization to make the network quicker and more accurate, and lastly a Leaky ReLU.

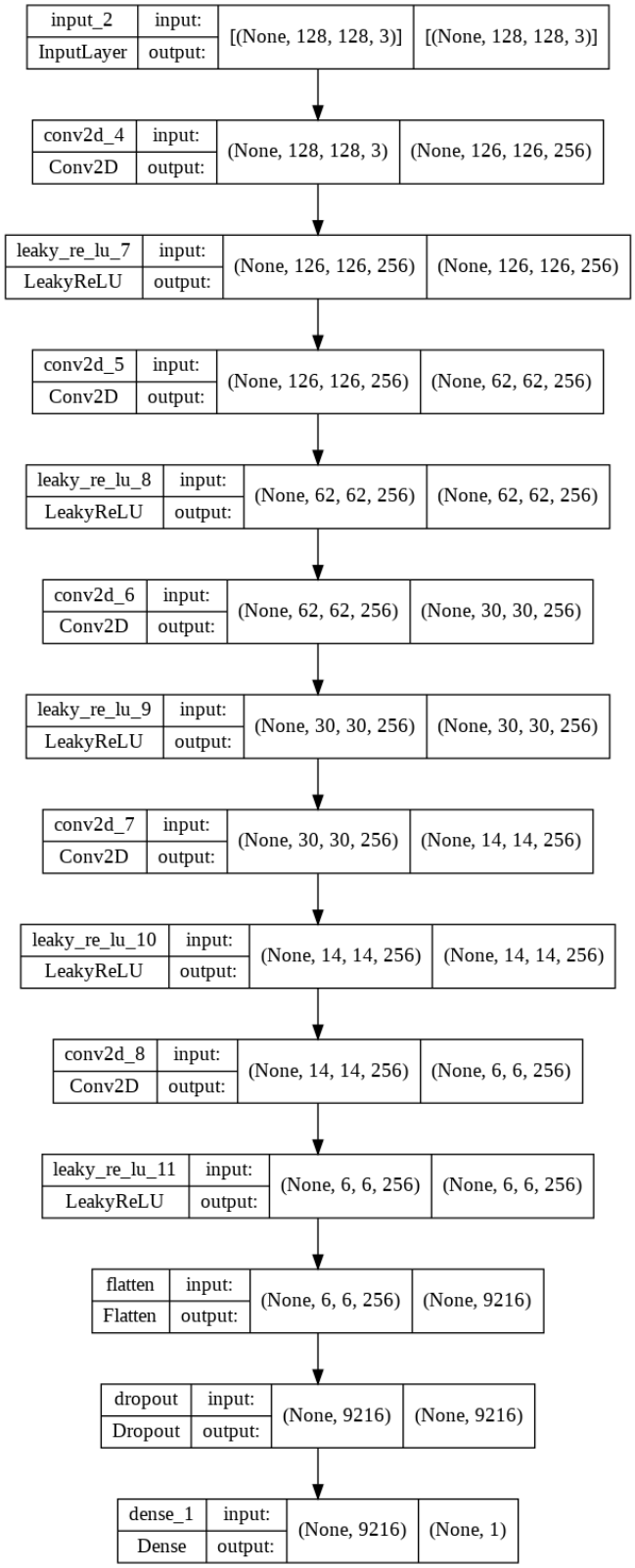



Fig 6: Discriminator Model Summary

The hardest component is training, and because a GAN has two separately trained networks, its training algorithm must deal with two issues:

GANs must manage two types of training (generator and discriminator).

## IX. Post Processing

Because the discriminator can't identify the difference between real and false, as the generator improves with training, the discriminator's performance deteriorates.

The discriminator has a 50% accuracy if the generator succeeds flawlessly.

To make its forecast, the discriminator essentially flips a coin.

This evolution causes difficulty for the GAN's overall convergence: the discriminator feedback becomes less useful over time.

If the GAN is trained after the discriminator has given random feedback, the generator will begin to train on trash feedback, and its quality will deteriorate.

The output of the Deep Face Image Generator after 3200 Epochs.

Fig 6: Deep-Imagify GANs Generated Images Last 8 images are selected

TABLE I. DEEP FACE IMAGE GENERATOR

| Deep Fake Face | Image loss calculation (Discriminator) | | |
|---|---|---|---|
| | *Image count* | *Starting Loss* | *Ending Loss* |
| 1. | Model Image Generation | 1.701 | 0.631 |

TABLE II. VQ-GAN & CLIP LOSS

| Image Generation | Image loss calculation GAN loss | | |
|---|---|---|---|
| | *Image count* | *Starting Loss* | *Ending Loss* |
| 2. | Underwater city image | 1.85 | 1.42 |
| 3. | Foot Ball Image | 1.80 | 1.58 |
| 4. | Mid Night City with Art Station | 1.85 | 1.66 |
| 5. | Mid Night City with Art Station (Portrait) | 3.54 | 3.28 |
| 6. | Global Warming Image | 1.84 | 1.67 |

## X. APPLICATION

VQ-GAN and CLIP Application built using Gradio and Angular frontend framework.

It is based on GPU CUDA, an optimizer using ADAM, it has text prompts from the user like what should be an image about. The weights for the model are loaded from vqgan_imagenet.

Users can select the quality of the image like draft, normal, and better. Type of the image like image, painting, pixel art.

Users also can select the size of the image like square, widescreen, or portrait.

**Deep-Imagify Dashboard:**

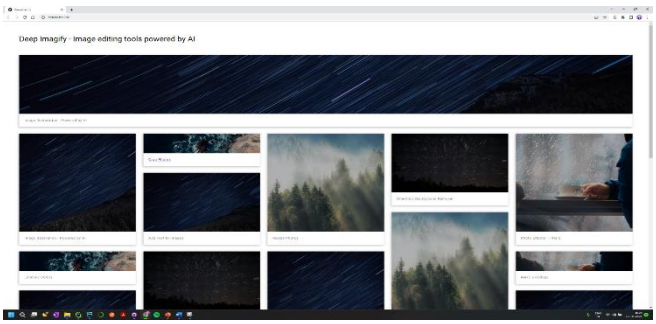



Fig 7: Deep-Imagify main dashboard

**Gradio UI:**

As part of Image Generation which is critical in image processing in many fields. An attempt to understand the GANs and their usage in image generation is implemented in this project. The project achieved less loss percentage which is a good sign for improved image generation. Due to the limited computational power usage of GPUs. The training process is implemented in limited Epochs.
Further increase in epochs will significantly improve the model as well as the predicted image.

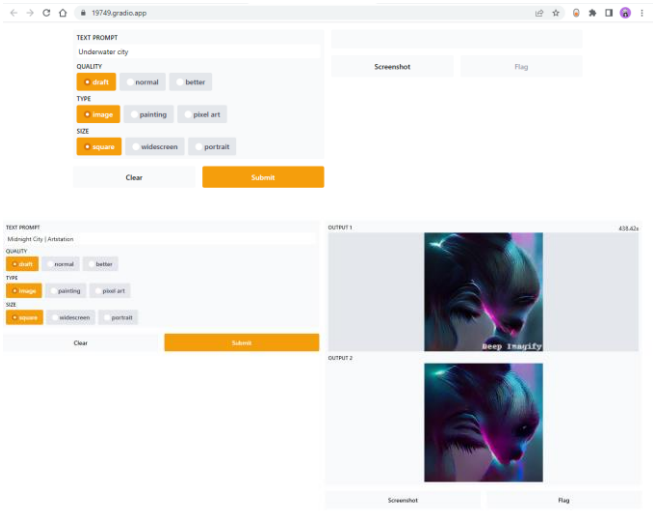



Fig 9: Deep-Imagify VQ-GAN and CLIP based application UI

**Images generated by VQ-GAN and CLIP:**

Fig 10: Deep-Imagify Generated Images after minimum of 300 epochs

## XI. Future Work

Deep Imagify To implement several other features like image generation from text, Image Style neural transfer, background Remover, Removing Text from the images, Sharpening the images, Image to Animated photo, and Enhancing color and image resolution will be primary development tasks as part of this Deep Imagify project.

## XII. References


*[1]. S. Yang, P. Luo, C. C. Loy, and X. Tang, "From Facial Parts Responses to Face Detection: A Deep Learning Approach", in IEEE International Conference on Computer Vision (ICCV), 2015.*

*[2]. Socher, R., Ganjoo, M., Manning, C. D., & Ng, A. (2013). "Zero-shot learning through cross-modal transfer." In NeurIPS 2013.*

*[3]. Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Ranzato, M. A., & Mikolov, T. (2013). "Devise A deep visual-semantic embedding model." In NeurIPS 2013.*

*[4]. Li, A., Jabri, A., Joulin, A., & van der Maaten, L. (2017). "Learning visual n-grams from web data." In Proceedings of the IEEE International Conference on Computer Vision 2017.*

*[5]. Doersch, C., Gupta, A., & Efros, A. A. (2015). "Unsupervised visual representation learning by context prediction." In ICCV 2015.*

*[6]. Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Networks.*

*[7]. Liu, Z., Luo, P., Wang, X., & Tang, X. (2014). Deep Learning Face Attributes in the Wild.*

*[8]. Zhang, C., & Peng, Y. (2018). Stacking VAE and GAN for Context-aware Text-to-Image Generation. 2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM), 1–5. https://doi.org/10.1109/BigMM.2018.8499439*

*[9]. Zhang, C., & Peng, Y. (2018). Stacking VAE and GAN for Context-aware Text-to-Image Generation. 2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM), 1–5. https://doi.org/10.1109/BigMM.2018.8499439*

*[10]. Im, D.-H., & Seo, Y.-S. (2021). Generating Face Images Using VQGAN and Sparse Transformer. 2021 International Conference on Information and Communication Technology Convergence (ICTC), 1642–1644. https://doi.org/10.1109/ICTC52510.2021.9621202*