GAN Introduction

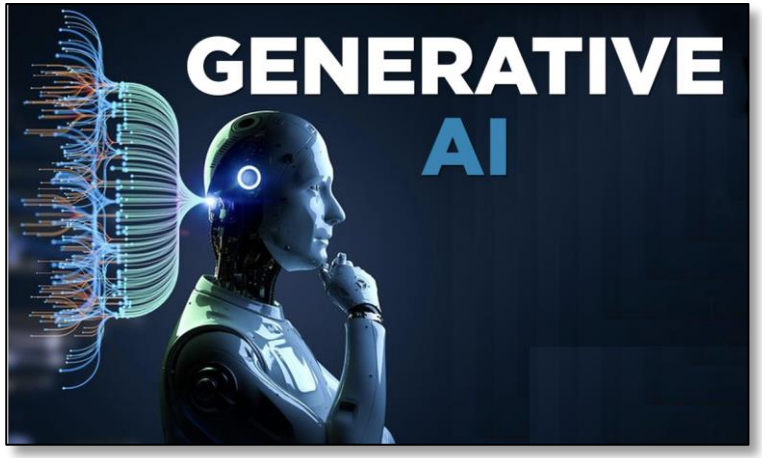
GAN Introduction

Generative Models

Until now, we have covered many different types of models, including Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory networks (LSTM). These models are designed to solve various problems, such as regression and classification, across different datasets. Some models perform better on images, while others excel with sequential data.

In this lecture, we will study a model specifically designed to address a new set of problems: Generative Adversarial Networks (GANs). These models can generate desired outputs based on our specifications

For instance, we can train GANs to create images from descriptions, write lyrics, and even produce songs. While many models are designed to tackle these types of challenges, today we will focus on the fascinating world of GANs.

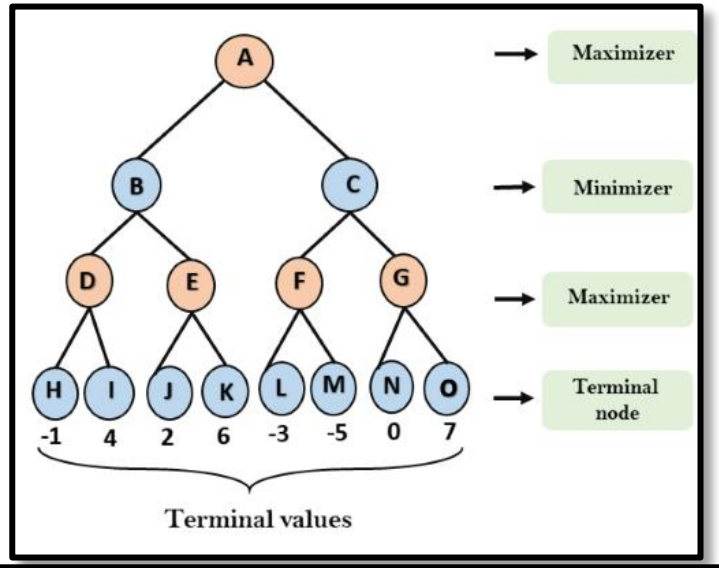


MinMax Game Theory

To understand how Generative Adversarial Networks (GANs) work, we first need to examine the concept of a minimax game. In a minimax game, two players take turns competing against each other. During each player's turn, they aim to maximize their own winning probability while minimizing their opponent's chances of winning by making optimal moves.

Imagine we are playing a game like tic-tac-toe. To determine the best move, we can think recursively. At each step, we consider all possible moves in our minds and evaluate them. If the game concludes after that move, the evaluation is straightforward.

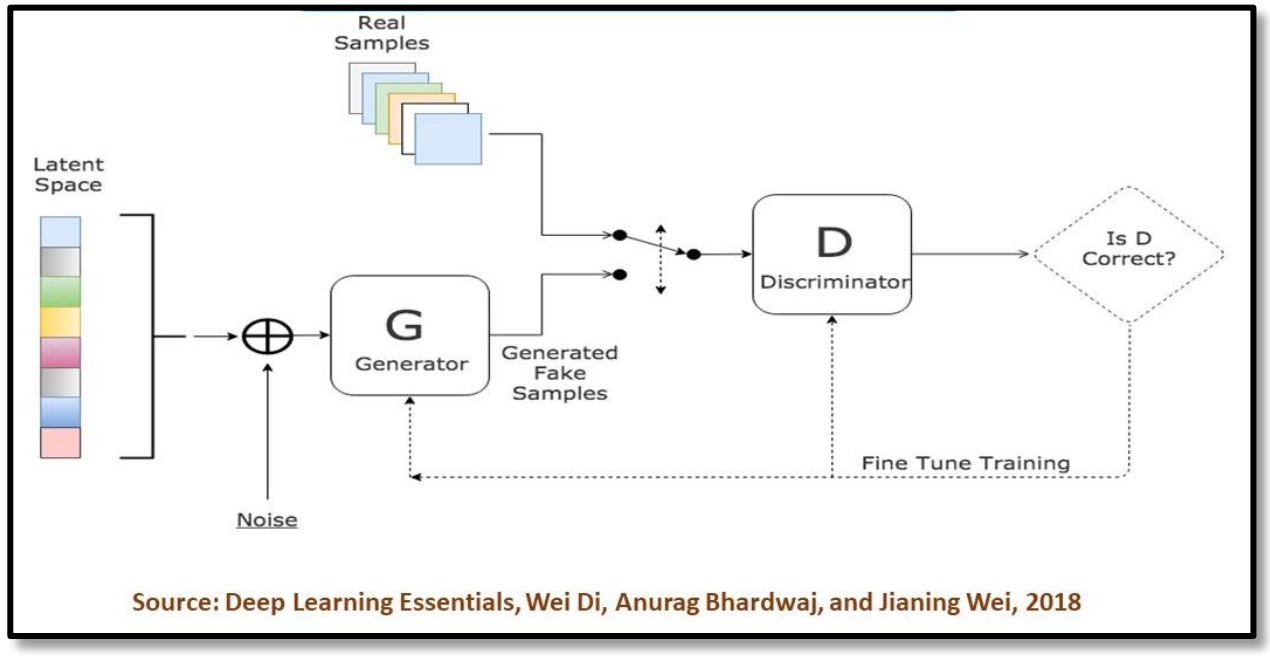
However, if the game is still ongoing, we must anticipate our opponent's moves and recursively consider the move that would minimize our winning probability, as this would be the logical choice for our opponent. In summary, we can say that the logical path of a minimax game is the one that gives us the highest minimum winning probability. This is where we find balance in the game.



Generative Adversarial Network

The Generative Adversarial Network (GAN) is a game theory-inspired neural network architecture created by Ian Goodfellow in 2014. This network comprises two components: a generator network and a discriminator network, both of which compete against each other in a minimax game. This competition allows both networks to improve simultaneously by attempting to outsmart one another.

In recent years, GANs have produced phenomenal results in various tasks, including image creation, music generation, and text generation.

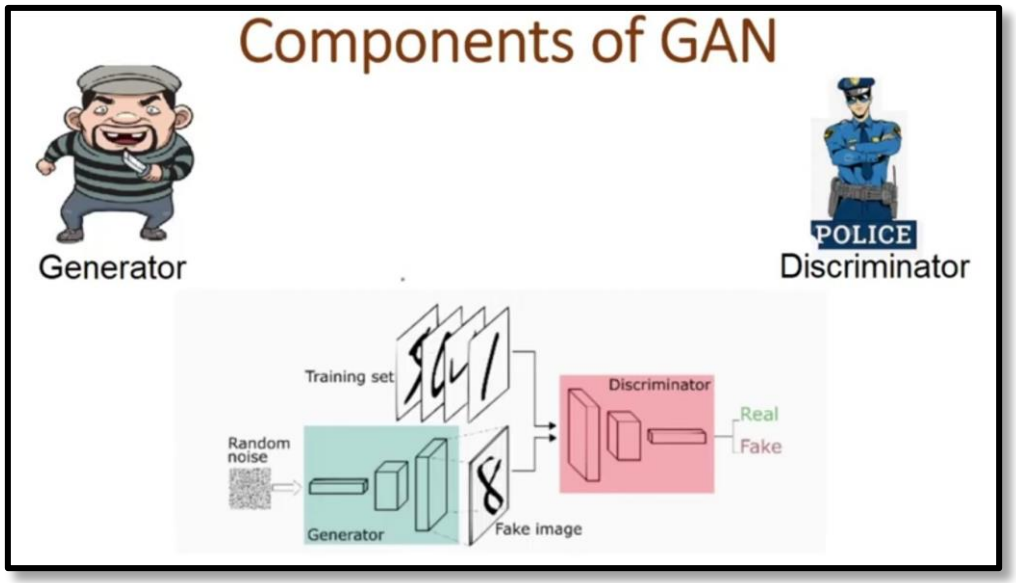


Source: Deep Learning Essentials, Wei Di, Anurag Bhardwaj, and Jianing Wei, 2018

Discriminator and Generator Network

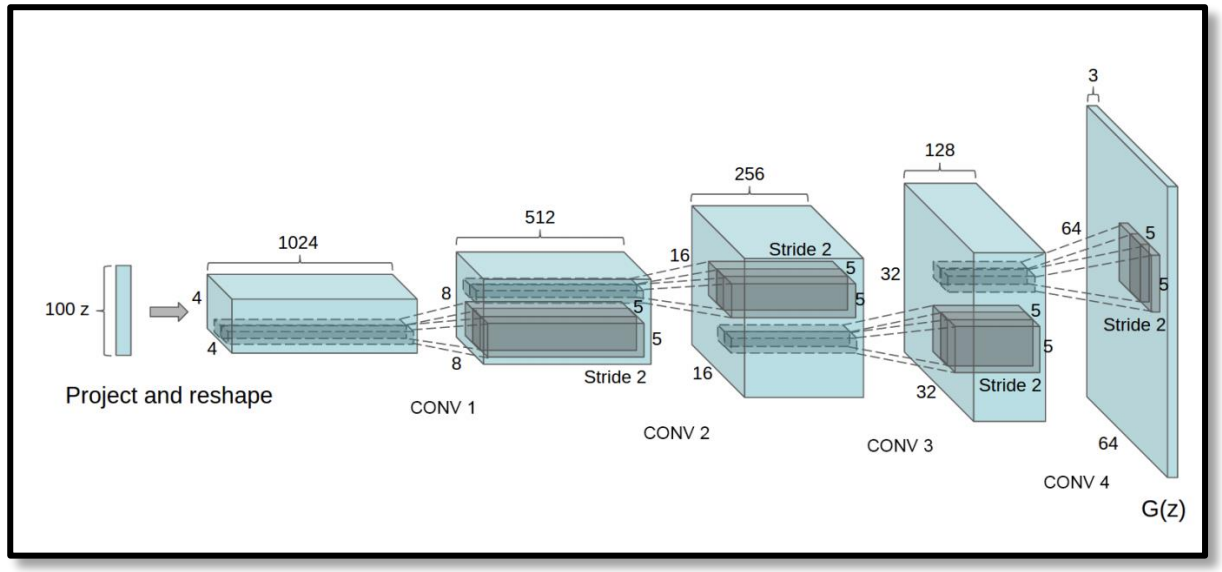
A Generative Adversarial Network (GAN) consists of two different neural networks that compete against each other. Imagine we have a dataset consisting of numerous images of dogs, and we want to train a model that can create realistic images of dogs.

The generator network is responsible for generating fake images. It receives a seed and produces a sample image. We then select some of these fake images, along with a few images from the actual dataset, and ask the discriminator network to predict which images are fake and which are real. Throughout this process, the generator network attempts to fool the discriminator into believing that the fake images are real.



Generator Network

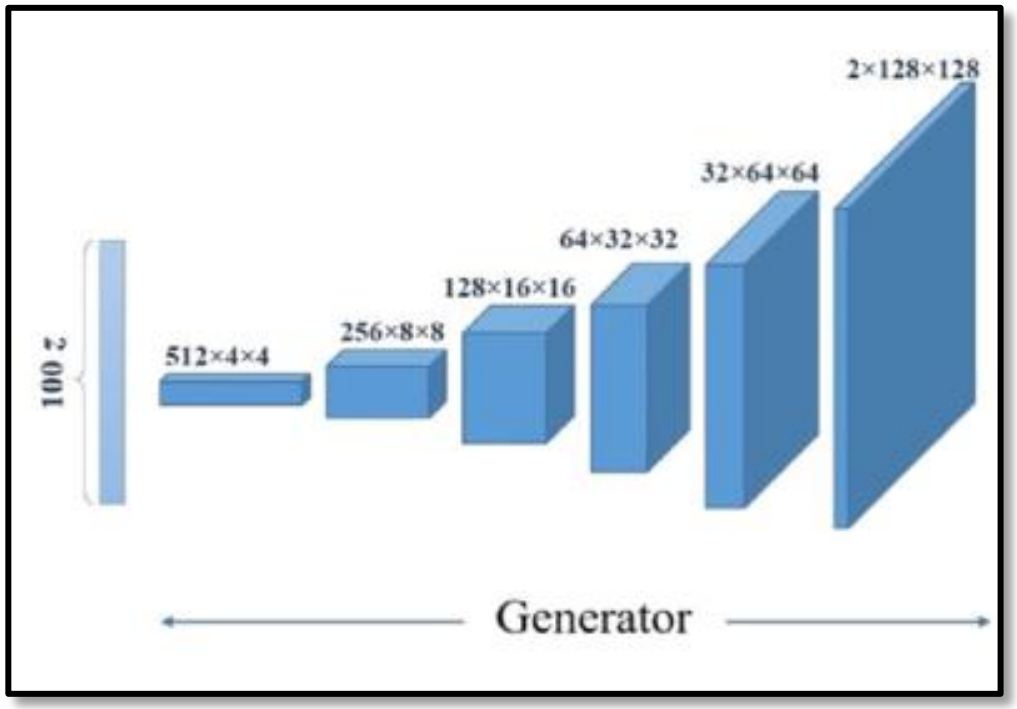
The goal of the generator network is to fool the discriminator. The generator should become so skilled at generating fake images that the discriminator cannot distinguish between fake and real images. If the discriminator fails to accurately classify fake and real images, it means that our generator network is performing exceptionally well.
When training the generator, our objective is to maximize the loss of the discriminator. To optimize and update the parameters of the generator network, we will move in the opposite direction of the gradient with respect to the generator's parameters.



Generator Network

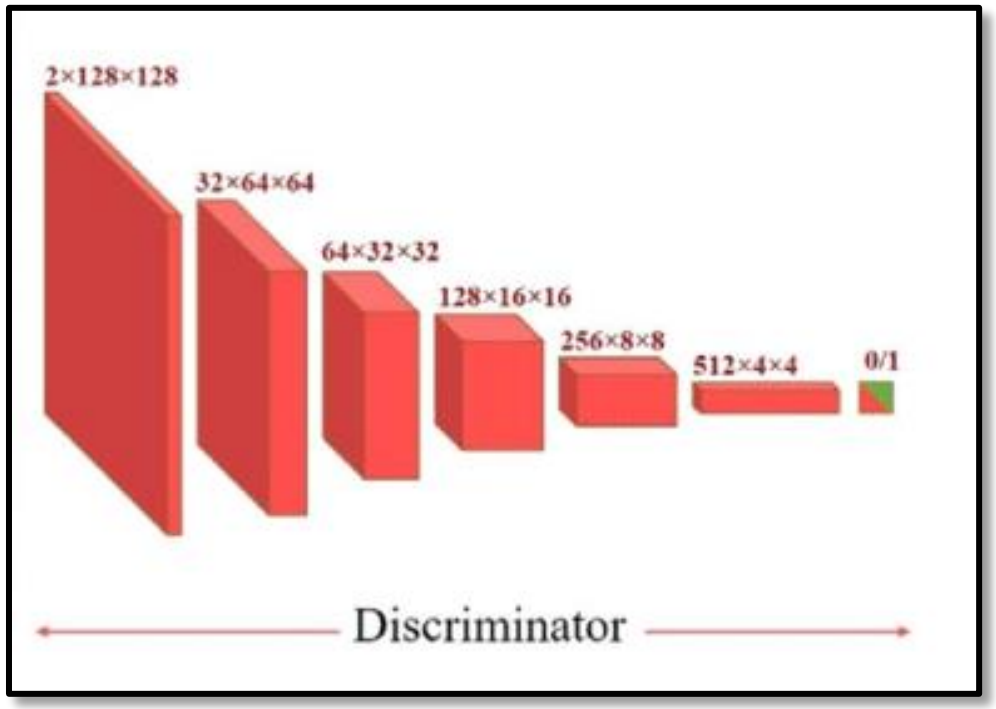
The structure of the generator differs from the models we have studied previously. This network takes a vector of numbers as its input and outputs an image.

The layers used in this network are transposed convolutional layers. These layers function as the opposite of convolutional layers found in a Convolutional Neural Network (CNN). The transposed convolution layer increases the spatial dimensions of the input feature maps. Unlike simple upsampling methods that merely increase dimensions without learning, transposed convolutions learn to fill in details during training. These layers are usually more challenging to train compared to other layers.



Discriminator Network

The discriminator is a simple classifier whose goal is to perform binary classification. This network aims to differentiate between the fake images generated by the generator and the real images from the actual dataset.
Initially, when the generator has not learned much, the discriminator can easily distinguish between the fake images and the real ones. However, as the generator improves and produces better fake images, the discriminator may struggle increasingly to classify the images correctly. The goal of the discriminator is to minimize its loss, and we update its parameters in the same direction as the gradient.



Training the Model

In Generative Adversarial Networks (GANs), the losses for the generator and discriminator are calculated using distinct functions that reflect their respective objectives in the adversarial training process.
The discriminator's loss aims to maximize its ability to correctly classify real and fake samples, which is typically expressed as binary cross-entropy loss. This loss function evaluates the probability that a given input is real and penalizes the discriminator for misclassifying real instances as fake and vice versa.

$$\min_G \max_D L(G,D) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}[\log(1 - D(G(z)))]$$

Standard GAN loss function (min-max GAN loss)

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$$

Discriminator loss and gradient

Training the Model

On the other hand, the generator's loss is designed to encourage it to produce samples that can fool the discriminator. This loss is calculated based on the discriminator's output for the generated samples, aiming to maximize the probability that these samples are classified as real. Essentially, the generator seeks to minimize the discriminator's ability to distinguish between real and fake samples. This formulation creates a minimax game where the generator seeks to minimize its loss while the discriminator seeks to maximize its own, leading to a dynamic interplay that drives both networks to improve over time.

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$$

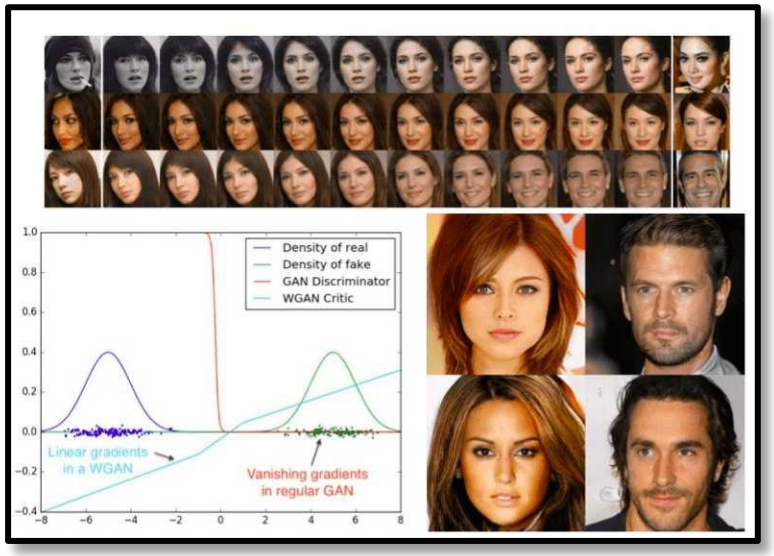
Generator loss and gradient

Challenges of Training GANs

Training Generative Adversarial Networks (GANs) is known to be very challenging because we are training two networks simultaneously. In particular, the convergence of the two networks is not guaranteed, as the gradient descent of either model does not directly impact the other, leading to oscillation and destabilization of model parameters.

Another significant issue is mode collapse, which arises from improper convergence. This phenomenon occurs when the generator outputs only a limited number of generated samples that it knows will successfully trick the discriminator.

Lastly, the discriminator could become so proficient that the gradient of the generator vanishes, preventing it from learning anything at all.

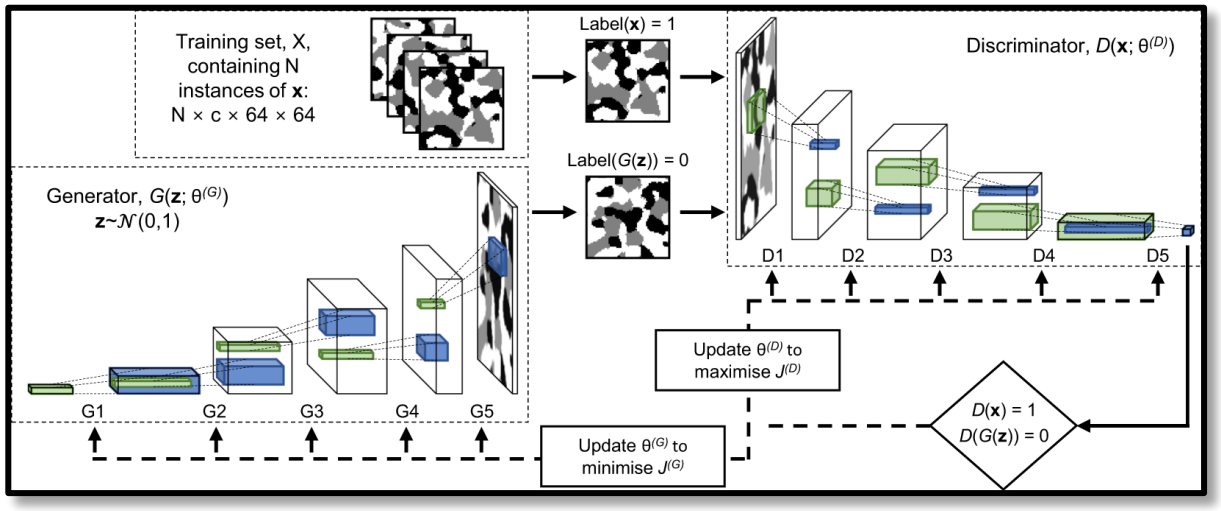


Summary

In this presentation, we covered the fundamentals of Generative Adversarial Networks (GANs). These networks are designed to generate images, text, and even music. They consist of two main components: the generator and the discriminator. We studied the design and objectives of each part.

We then discussed the training process for both the generator and discriminator networks. Finally, we mentioned some of the challenges encountered when training these models. There are many open areas for research in GANs.

In the next presentation, we will demonstrate how to build a GAN using Python.



Source: www.nature.com