

Deep Generative Models

Prepared by Dr. Sherif Mostafa

AI314

Master of Science in Artificial Intelligence

Lecture 4



Generative Adversarial Network

What is Generative Adversarial Network

Types of GANs

Generative Adversarial Networks (GANs) architecture

How does a GAN work?

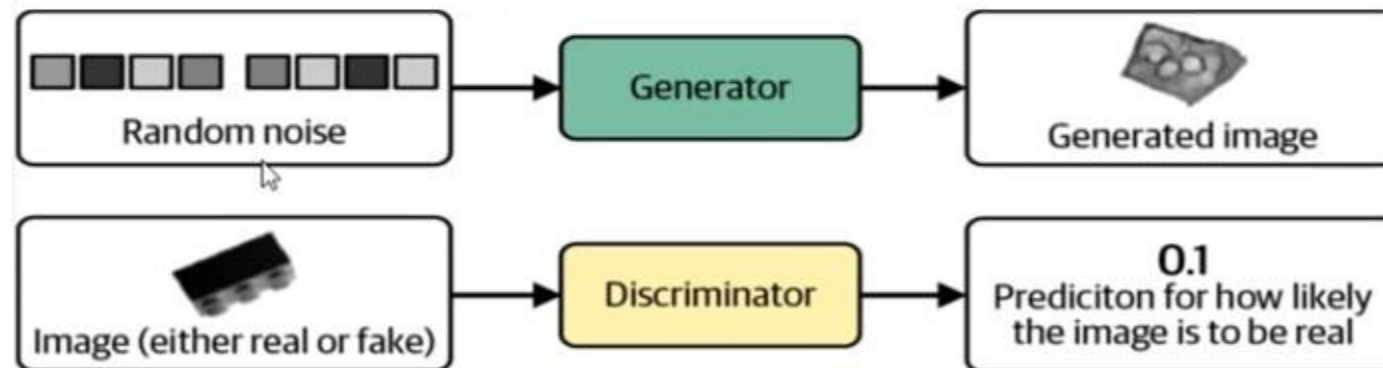
Application Of Generative Adversarial Networks (GANs)

What is a Generative Adversarial Network?

Generative Adversarial Networks (GANs) are a powerful class of machine learning that are used for an unsupervised learning. GANs are made up of two neural networks, a discriminator and a generator. They use adversarial training to produce artificial data that is identical to actual data.

What is a Generative Adversarial Network?

- The Generator attempts to fool the Discriminator, which is tasked with accurately distinguishing between produced and genuine data, by producing random noise samples.
- Realistic, high-quality samples are produced as a result of this competitive interaction, which drives both networks toward advancement.



What is a Generative Adversarial Network?

GANs are proving to be highly versatile artificial intelligence tools, as evidenced by their extensive use in image synthesis, style transfer, and text-to-image synthesis. They have also revolutionized generative modeling.

Through adversarial training, these models engage in a competitive interplay until the generator becomes adept at creating realistic samples, fooling the discriminator approximately half the time.

Generative Adversarial Networks (GANs) can be broken down into three parts:

- **Generator:** To learn a generative model, which describes how data is generated in terms of a probabilistic model.
- **Adversarial:** The word adversarial refers to setting one thing up against another. This means that, in the context of GANs, the generative result is compared with the actual images in the data set.

Generative Adversarial Networks (GANs) can be broken down into three parts:

- **A mechanism known as a discriminator is used to apply a model that attempts to distinguish between real and fake images. Discriminator is a neural network that learn to distinguish between real data and data generated by the generator. It takes as input an observations and outputs a probability of whether it is real or fake.**

Types of GANs

Vanilla GAN:

This is the simplest type of GAN. Here, the Generator and the Discriminator are simple a basic multi-layer perceptrons. In vanilla GAN, the algorithm is really simple, it tries to optimize the mathematical equation using stochastic gradient descent.

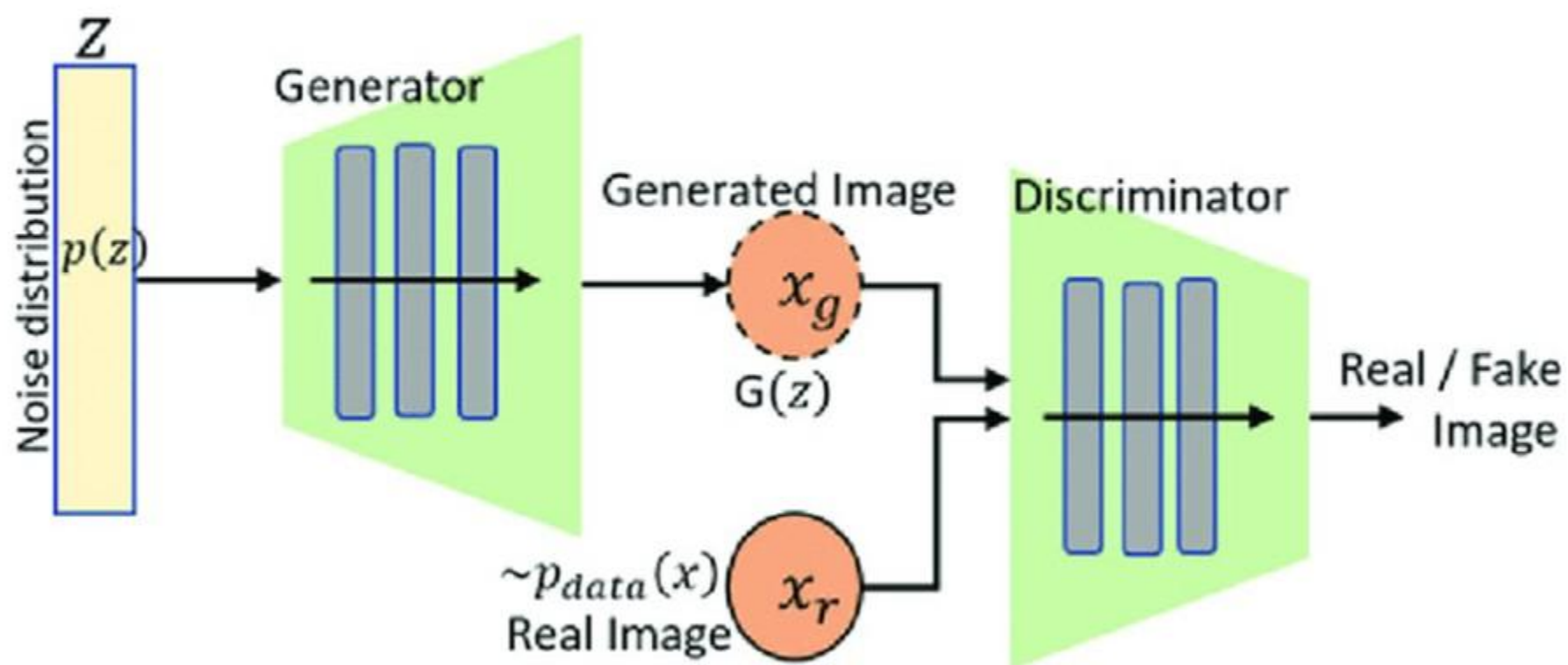
Types of GANs

Vanilla GAN:

1. The goal of the generator is to create “Fake Data” that resembles the “Real Data” using the “Noise Vectors”.
2. Whereas the goal of the discriminator is to distinguish generated “Fake Data” from the “Real Data”.

These two models competitively learned new weights. In other words, given noise vectors, the generator gradually learns to create outputs that hopefully trick the discriminator. When the discriminator is not able to give accurate predictions on whether the input is from the real dataset, the generator is recognized as a good model.

Types of GANs



Vanilla GAN

Types of GANs

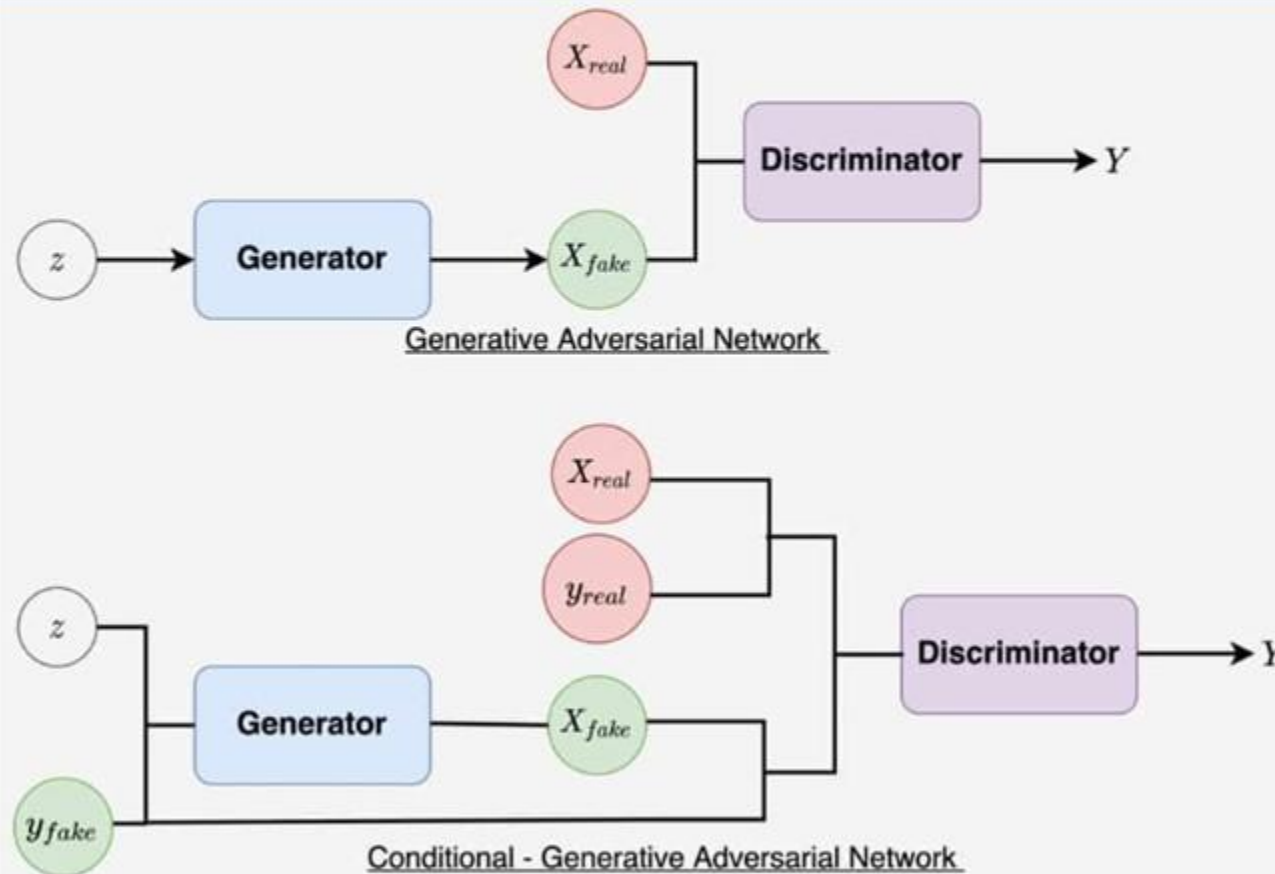
Conditional GAN (CGAN):

CGAN can be described as a deep learning method in which some conditional parameters are put into place.

In CGAN, an additional parameter 'y' is added to the Generator for generating the corresponding data.

Labels are also put into the input to the Discriminator in order for the Discriminator to help distinguish the real data from the fake generated data.

Types of GANs



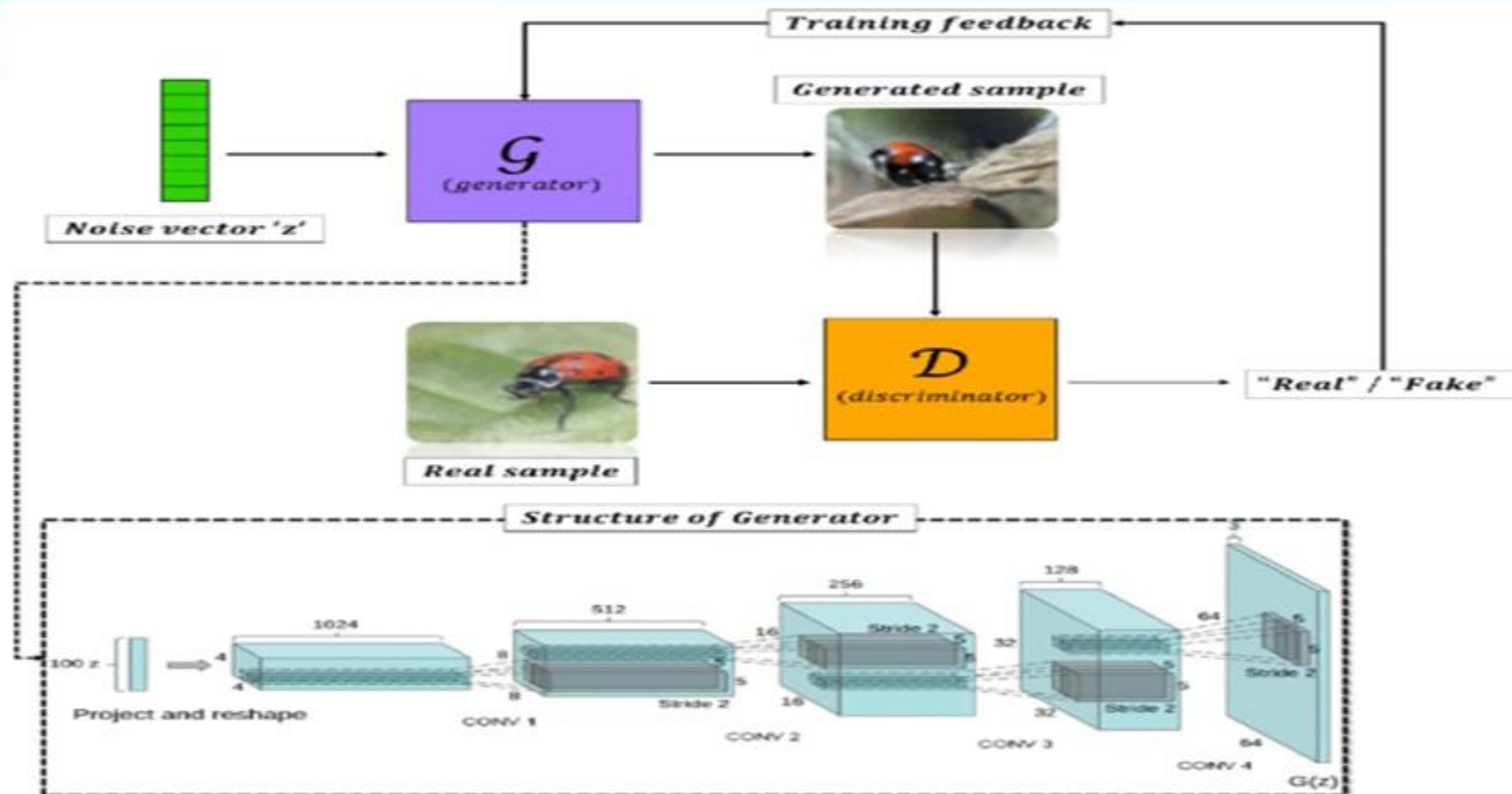
Types of GANs

Deep Convolutional GAN (DCGAN):

DCGAN is one of the most popular and also the most successful implementations of GAN. It is composed of ConvNets in place of multi-layer perceptrons.

The ConvNets are implemented without max pooling, which is in fact replaced by convolutional stride. Also, the layers are not fully connected

Types of GANs



Deep Convolutional GAN (DCGAN)

Types of GANs

Laplacian Pyramid GAN (LAPGAN): The Laplacian pyramid is a linear invertible image representation consisting of a set of band-pass images, spaced an octave apart, plus a low-frequency residual.

This approach uses multiple numbers of Generator and Discriminator networks and different levels of the Laplacian Pyramid.

This approach is mainly used because it produces very high-quality images. The image is down-sampled at first at each layer of the pyramid and then it is again up-scaled at each layer in a backward pass where the image acquires some noise from the Conditional GAN at these layers until it reaches its original size.

Types of GANs

Super Resolution GAN (SRGAN):

SRGAN as the name suggests is a way of designing a GAN in which a deep neural network is used along with an adversarial network in order to produce higher-resolution images. This type of GAN is particularly useful in optimally up-scaling native low-resolution images to enhance their details minimizing errors while doing so

Architecture of GANs

Generator Model

- A key element responsible for creating fresh, accurate data in a Generative Adversarial Network (GAN) is the generator model.
- The generator takes random noise as input and converts it into complex data samples, such text or images. It is commonly depicted as a deep neural network.
- The generator continuously learns and improves its ability to generate realistic data through iterative training.

Architecture of GANs

Generator Model

The training data's underlying distribution is captured by layers of learnable parameters in its design through training. The generator adjusts its output to produce samples that closely mimic real data as it is being trained by using backpropagation to fine-tune its parameters.

The generator's ability to generate high-quality, varied samples that can fool the discriminator is what makes it successful.

Architecture of GANs

Generator Loss

The objective of the generator in a GAN is to produce synthetic samples that are realistic enough to fool the discriminator. The generator achieves this by minimizing its loss function J_G . The loss is minimized when the log probability is maximized, i.e., when the discriminator is highly likely to classify the generated samples as real.

Architecture of GANs

The following equation is given below:

$$J_G = -\frac{1}{m} \sum_{i=1}^m \log D(G(Z_i))$$

Where, J_G measure how well the generator is fooling the discriminator.

- $\log D(G(Z_i))$ represents log probability of the discriminator being correct for generated samples.
- The generator aims to minimize this loss, encouraging the production of samples that the discriminator classifies as real $\log D(G(Z_i))$ close to 1

Architecture of GANs

Discriminator Model

An artificial neural network called a discriminator model is used in Generative Adversarial Networks (GANs) to differentiate between generated and actual input. By evaluating input samples and allocating probability of authenticity, the discriminator functions as a binary classifier.

Over time, the discriminator learns to differentiate between genuine data from the dataset and artificial samples created by the generator. This allows it to progressively hone its parameters and increase its level of proficiency.

Architecture of GANs

Convolutional layers

usually used in the architecture when dealing with picture data. Maximizing the discriminator's capacity to accurately identify generated samples as fraudulent and real samples as authentic is the aim of the adversarial training procedure. The discriminator grows increasingly discriminating as a result of the generator and discriminator's interaction, which helps the GAN produce extremely realistic-looking synthetic data overall.

Architecture of GANs

Discriminator Loss

The discriminator reduces the negative log likelihood of correctly classifying both produced and real samples. This loss incentivizes the discriminator to accurately categorize generated samples as fake and real samples with the following equation:

$$J_D = -\frac{1}{m} \sum_{i=1}^m \log D(x_i) - \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(Z_i)))$$

Architecture of GANs

$$J_D = -\frac{1}{m} \sum_{i=1}^m \log D(x_i) - \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(Z_i)))$$

- J_D assesses the discriminator's ability to discern between produced and actual samples.
- The log likelihood that the discriminator will accurately categorize real data is represented by $\log D(x_i)$
- The log chance that the discriminator would correctly categorize generated samples as fake is represented by $\log(1 - D(G(Z_i)))$
- The discriminator aims to reduce this loss by accurately identifying artificial and real samples.

Architecture of GANs

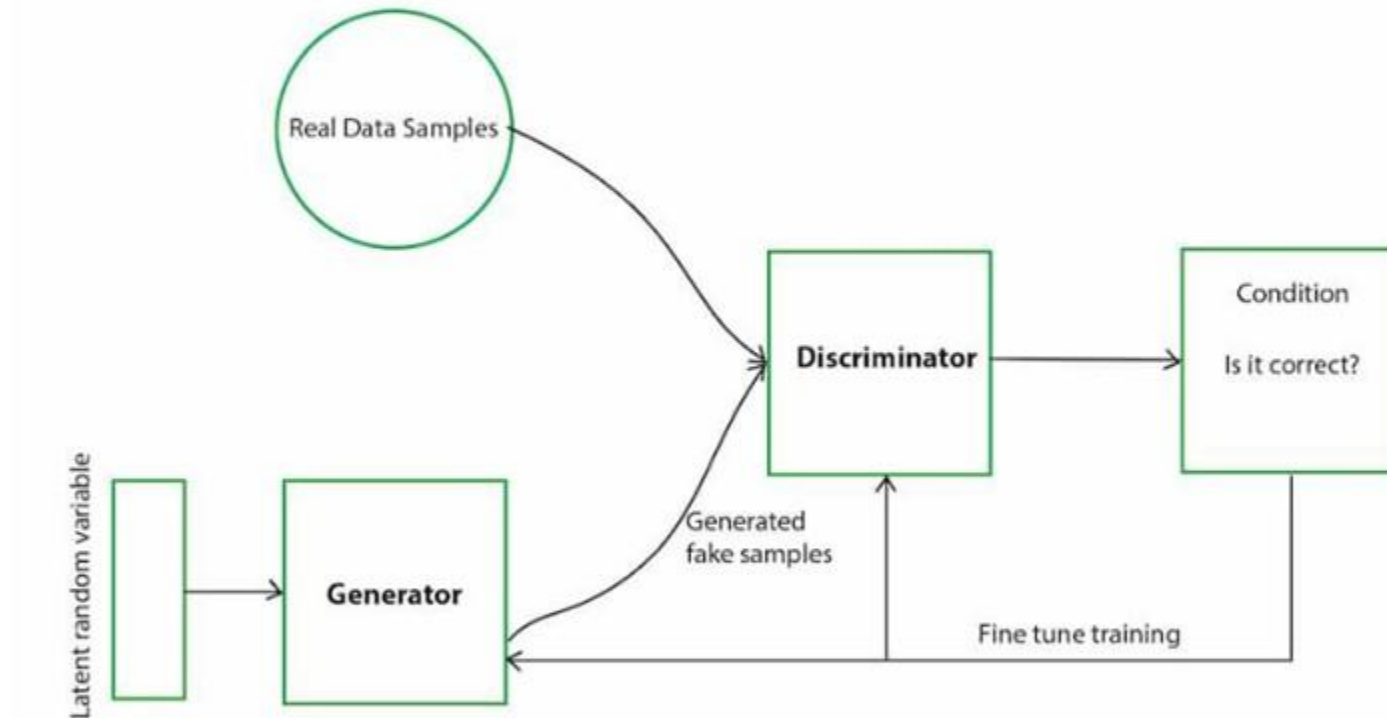
In a Generative Adversarial Network (GAN), the minimax loss formula is provided by:

$$\min_G \max_D (G, D) = [\mathbb{E}_{x \sim P_{data}} [\log D(x)] + \mathbb{E}_{Z \sim p_Z(Z)} [\log(1 - D(g(Z)))]$$

Where,

- G is generator network and D is the discriminator network
- Actual data samples obtained from the true data distribution $P_{data}(x)$ are represented by x .
- Random noise sampled from a previous distribution $p_Z(Z)$ (usually a normal or uniform distribution) is represented by Z .
- $D(x)$ represents the discriminator's likelihood of correctly identifying actual data as real.
- $D(g(Z))$ is the likelihood that the discriminator will identify generated data coming from the generator as authentic.

Architecture of GANs



new samples that are similar in structure and characteristics to the input data.

How does a GAN work?

The steps involved in how a GAN works:

Initialization: Two neural networks are created: a Generator (G) and a Discriminator (D).

- G is tasked with creating new data, like images or text, that closely resembles real data.
- D acts as a critic, trying to distinguish between real data (from a training dataset) and the data generated by G.

How does a GAN work?

Generator's First Move: G takes a random noise vector as input. This noise vector contains random values and acts as the starting point for G's creation process. Using its internal layers and learned patterns, G transforms the noise vector into a new data sample, like a generated image

Discriminator's Turn: D receives two kinds of inputs:

- Real data samples from the training dataset.
- The data samples generated by G in the previous step. D's job is to analyze each input and determine whether it's real data or something G cooked up. It outputs a probability score between 0 and 1. A score of 1 indicates the data is likely real, and 0 suggests it's fake.

How does a GAN work? (The Learning Process)

Now, the adversarial part comes in:

- If D correctly identifies real data as real (score close to 1) and generated data as fake (score close to 0), both G and D are rewarded to a small degree. This is because they're both doing their jobs well.
- However, the key is to continuously improve. If D consistently identifies everything correctly, it won't learn much. So, the goal is for G to eventually trick D.

How does a GAN work?

Generator's Improvement:

- When D mistakenly labels G's creation as real (score close to 1), it's a sign that G is on the right track. In this case, G receives a significant positive update, while D receives a penalty for being fooled.
- This feedback helps G improve its generation process to create more realistic data

How does a GAN work?

Discriminator's Adaptation:

- Conversely, if D correctly identifies G's fake data (score close to 0), but G receives no reward, D is further strengthened in its discrimination abilities.
- This ongoing duel between G and D refines both networks over time.

As training progresses, G gets better at generating realistic data, making it harder for D to tell the difference. Ideally, G becomes so adept that D can't reliably distinguish real from fake data. At this point, G is considered well-trained and can be used to generate new, realistic data samples.

Application Of Generative Adversarial Networks (GANs)

Generative Adversarial Networks, have many uses in many different fields. Here are some of the widely recognized uses of GANs:

Image Synthesis and Generation :

GANs are often used for picture synthesis and generation tasks, They may create fresh, lifelike pictures that mimic training data by learning the distribution that explains the dataset. The development of lifelike avatars, high-resolution photographs, and fresh artwork have all been facilitated by these types of generative networks.

Application Of Generative Adversarial Networks (GANs)

Image-to-Image Translation :

GANs may be used for problems involving image-to-image translation, where the objective is to convert an input picture from one domain to another while maintaining its key features. GANs may be used, for instance, to change pictures from day to night, transform drawings into realistic images, or change the creative style of an image

Application Of Generative Adversarial Networks (GANs)

Text-to-Image Synthesis :

GANs have been used to create visuals from descriptions in text. GANs may produce pictures that translate to a description given a text input, such as a phrase or a caption. This application might have an impact on how realistic visual material is produced using text-based instructions.

Application Of Generative Adversarial Networks (GANs)

Data Augmentation :

GANs can augment present data and increase the robustness and generalizability of machine-learning models by creating synthetic data samples.

Data Generation for Training :

GANs can enhance the resolution and quality of low-resolution images. By training on pairs of low-resolution and high-resolution images, GANs can generate high-resolution images from low-resolution inputs, enabling improved image quality in various applications such as medical imaging, satellite imaging, and video enhancement.

Advantages of GAN

The advantages of the GANs are as follows:

Synthetic data generation:

GANs can generate new, synthetic data that resembles some known data distribution, which can be useful for data augmentation, anomaly detection, or creative applications.

High-quality results:

GANs can produce high-quality, photorealistic results in image synthesis, video synthesis, music synthesis, and other tasks.

Advantages of GAN

Unsupervised learning:

GANs can be trained without labeled data, making them suitable for unsupervised learning tasks, where labeled data is scarce or difficult to obtain.

Versatility:

GANs can be applied to a wide range of tasks, including image synthesis, text-to-image synthesis, image-to-image translation, anomaly detection, data augmentation, and others.

Disadvantages of GAN

Training Instability:

GANs can be difficult to train, with the risk of instability, mode collapse, or failure to converge.

Computational Cost:

GANs can require a lot of computational resources and can be slow to train, especially for high-resolution images or large datasets.

Overfitting:

GANs can overfit the training data, producing synthetic data that is too similar to the training data and lacking diversity.

Disadvantages of GAN

Bias and Fairness:

GANs can reflect the biases and unfairness present in the training data, leading to discriminatory or biased synthetic data.

Interpretability and Accountability:

GANs can be opaque and difficult to interpret or explain, making it challenging to ensure accountability, transparency, or fairness in their applications.

Challenges in training Generative Adversarial Networks

Generative Adversarial Networks (GANs) have emerged as a powerful class of deep learning models, capable of generating highly realistic diverse synthetic data . However, training GANs comes with its own unique set of challenges that must be addressed to ensure stable and effective model performance.

Challenges in training Generative Adversarial Networks

Uninformative loss function

Lack of correlation:

The loss function of generator may not be a reliable indicator of image quality, as the generator is only evaluated against the current discriminator, which is constantly improving. This can lead to the generator loss increasing over time even as the generator image improve.

Challenges in training Generative Adversarial Networks

Uninformative loss function

Difficulty in monitoring:

The disconnect between generator loss and image quality make it challenging to monitor the progress of GAN training, hampering the quality to make informed adjustments to the model.

Challenges in training Generative Adversarial Networks

Uninformative loss function

Educated trail and error:

Finding the right set of hyperparameters for a GAN model often requires extensive experiments as GANs are highly sensitive to even minor changes in various parameters.

Challenges in training Generative Adversarial Networks

Hyperparameter sensitivity

GANs have many hyperparameters to tune including:

- The architecture of the discriminator and generator
- Batch normalization (normalizes the activations of each layer, helping to prevent gradients from exploding or vanishing)
- Learning rate
- Convolution filter and kernel size
- Latent space size

What challenges do GAN face?

GANs encounter difficulties such training instability, mode collapse (when the generator generates a limited range of samples), and striking the correct balance between the discriminator and generator. It's frequently necessary to carefully build the model architecture and tune the hyperparameters.

How are GAN evaluated?

The produced samples' quality, diversity, and resemblance to real data are the main criteria used to assess GANs. For quantitative assessment, metrics like the Fréchet Inception Distance (FID) and Inception Score are frequently employed.

Can GAN be used for tasks other than image generation?

Yes, different tasks can be assigned to GANs. Text, music, 3D models, and other things have all been generated with them. The usefulness of conditional GANs is expanded by enabling the creation of specific content under certain input conditions.

What are some famous architectures of GANs?

A few well-known GAN architectures are Progressive GAN (PGAN), Wasserstein GAN (WGAN), Conditional GAN (CGAN), Deep Convolutional GAN (DCGAN), and Vanilla GAN. Each has special qualities and works best with particular kinds of data and tasks.



Thank You