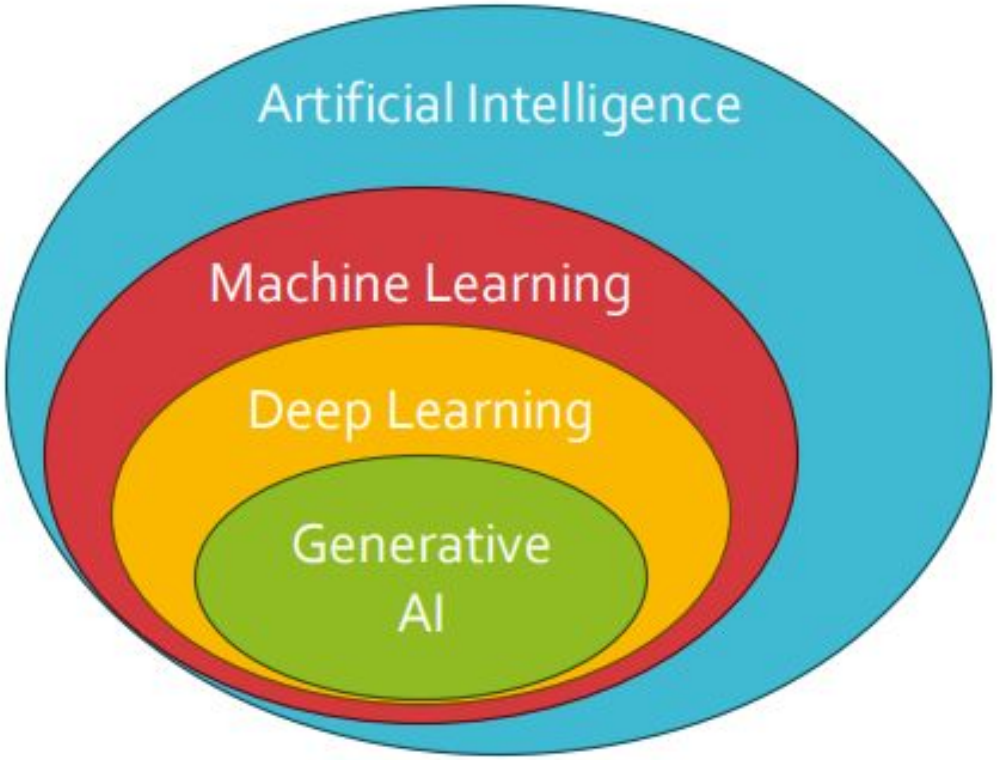
This is CS4045

GCR : dxuxugo

Generative AI


Artificial Intelligence
Machine Learning
Deep Learning
Generative
AI

What is Generative AI (GenAI)?

GenAI creates new content—text, images, code, music—based on patterns learned from data. Unlike traditional AI, it generates something new instead of analyzing existing data.

How Does GenAI Work?

GenAI models learn from vast datasets and generate new outputs using:

Neural Networks
Deep learning models for content creation.

Pattern Recognition
Predicts and generates sequences.

Pre-trained Models
Fine-tuned for specific tasks.

Applications of GenAI

Marketing
AI-generated ads & content.

Gaming
AI-generated characters & environments.

Healthcare
AI-powered drug discovery.

Education
AI tutors & content summarization.

Challenges & Ethical Concerns

Bias & Misinformation
Risk of false data.

Deepfakes & Manipulation
AI-created fake media.

Copyright Issues
Ownership of AI content.

Privacy Risks
AI models trained on personal data.

Types of Generative AI Models

Transformer Models (Text & Code)

Used for chatbots, writing, coding assistants.
📌 Examples: GPT-4, Gemini, Llama, Claude

Diffusion Models (Images & Videos)

Turn random noise into realistic visuals.
📌 Examples: Stable Diffusion, DALL·E, Midjourney

GANs (Realistic Media)

Generative Adversarial Network

Two competing AIs create lifelike images & videos.

📌 Examples: DeepFaceLab, This Person Does Not Exist

VAEs (Data Compression & Enhancement)

Variational Autoencoders

Used in creative design and data reconstruction.

📌 Examples: AI-assisted image variations

RNN & LSTM (Sequential Data)

Recurrent Neural Network

Long short-term memory

Used for music, speech, and handwriting generation.

📌 Examples: AI-generated poetry, song compositions

Generative Adversarial networks (GAN)

GAN

GANs are generative models: they create new data instances that resemble your training data

For example, GANs can create images that look like photographs of human faces, even though the faces don't belong to any real person

Images generated by a GAN created by NVIDIA

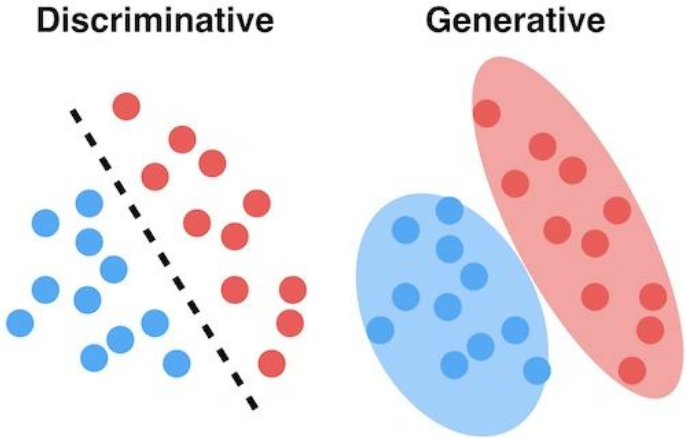
GAN

GANs achieve this level of realism by pairing a generator and discriminator

Generator learns to produce the target output, with a discriminator, which learns to distinguish true data from the output of the generator

The generator tries to fool the discriminator, and the discriminator tries to keep from being fooled

Background



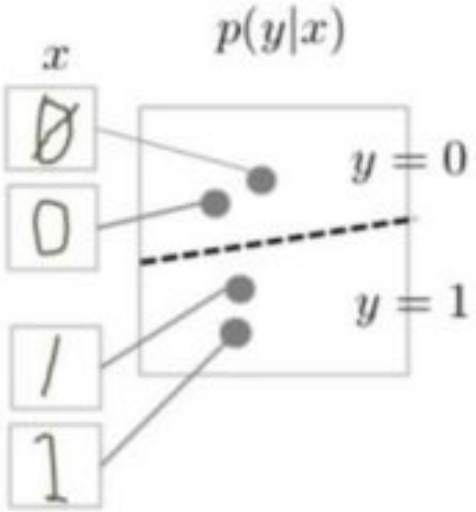
Generative models can generate new data instances

Discriminative models discriminate between different kinds of data instances

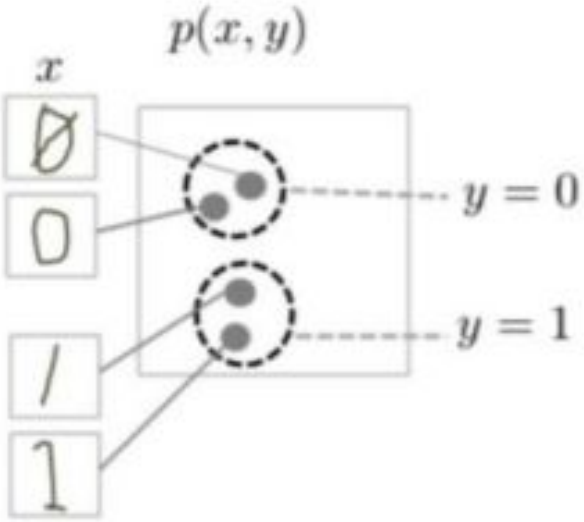
More formally, given a set of data instances X and a set of labels Y:

- Generative models capture the joint probability p(X, Y), or just p(X) if there are no labels
- Discriminative models capture the conditional probability p(Y| X)

- Discriminative Model



- Generative Model



Overview of GAN Structure

A generative adversarial network (GAN) has two parts:

The generator learns to generate plausible data. The generated instances become negative training examples for the discriminator

The discriminator learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results

Overview of GAN Structure

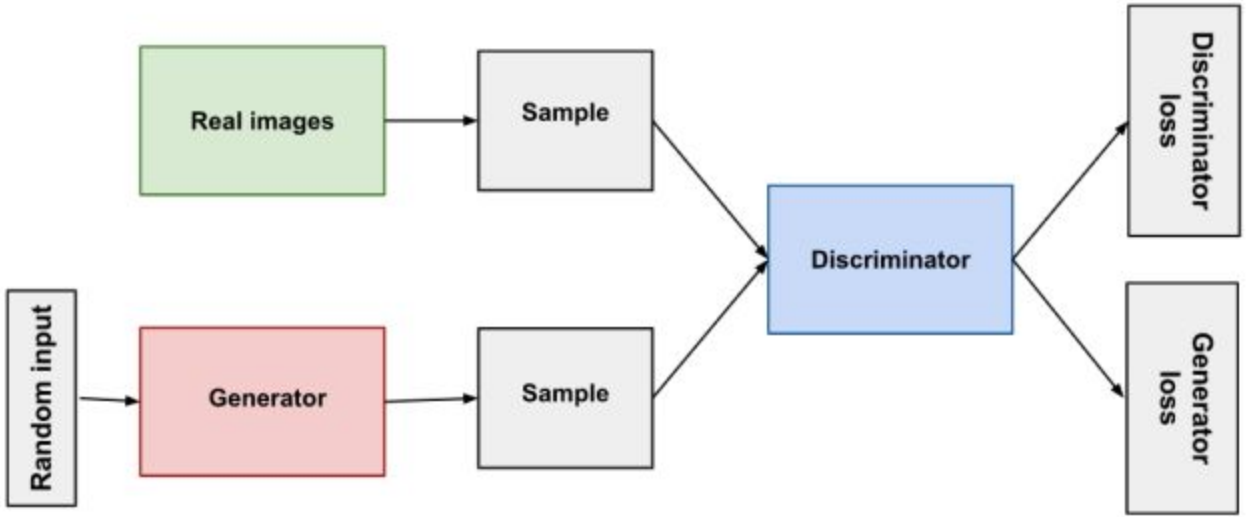
When training begins, the generator produces obviously fake data, and the discriminator quickly learns to tell that it's fake:

Overview of GAN Structure

As training progresses, the generator gets closer to producing output that can fool the discriminator:

Finally, if generator training goes well, the discriminator gets worse at telling the difference between real and fake. It starts to classify fake data as real, and its accuracy decreases

GAN Block Diagram



Overview of GAN Structure

Both the generator and the discriminator are neural networks. The generator output G(z) is connected directly to the discriminator input D(G(z))

Through backpropagation, the discriminator's classification provides a signal that the generator uses to update its weights

Discriminator

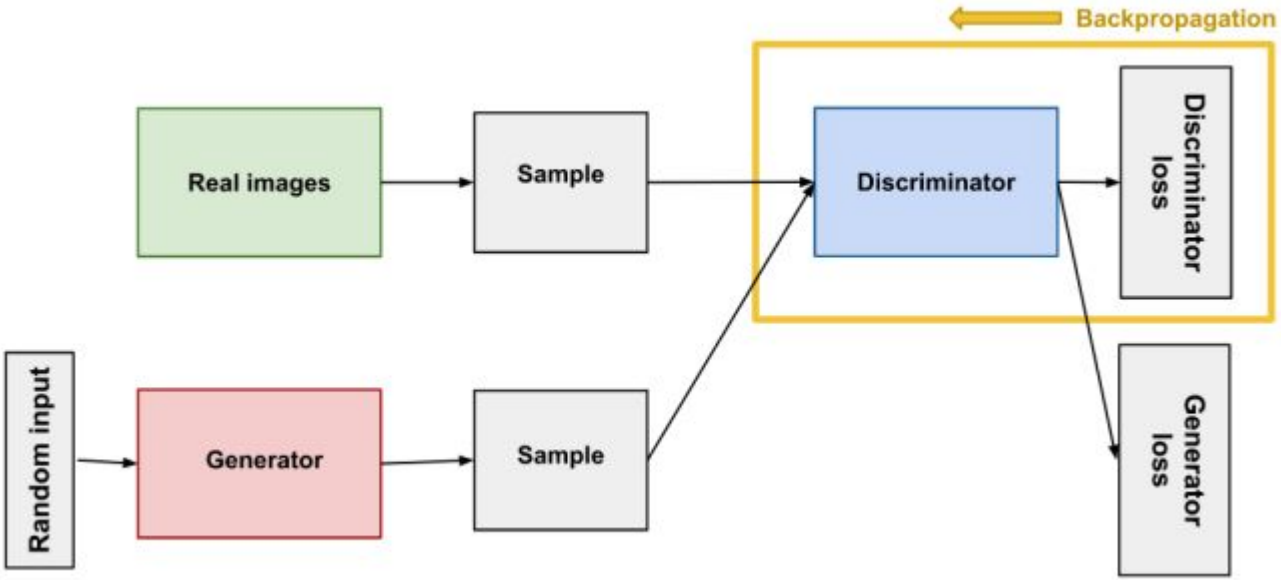
The discriminator in a GAN is simply a classifier

It tries to distinguish real data from the data created by the generator. It could use any network architecture appropriate to the type of data it's classifying

The discriminator's training data comes from two sources:

- Real data instances,x, such as real pictures of people. The discriminator uses these instances as positive examples during training
- Fake data instances created by the generator, G(z). The discriminator uses these instances as negative examples during training

Discriminator



Discriminator Training

The discriminator connects to two loss functions

During discriminator training, the discriminator ignores the generator loss and just uses the discriminator loss.

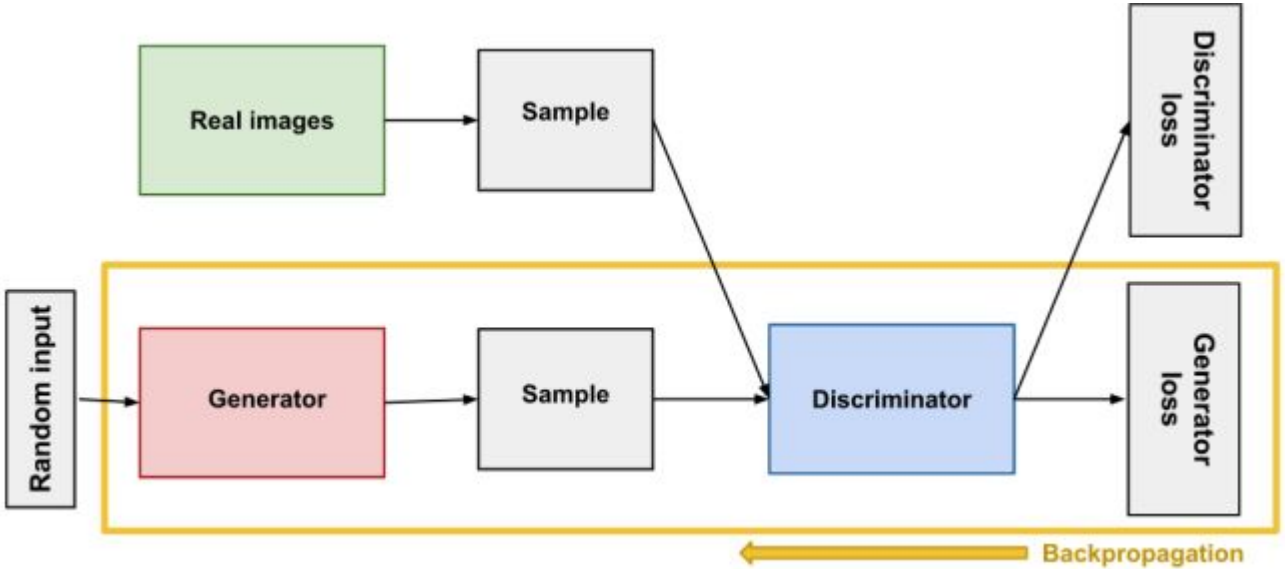
During discriminator training:

- The discriminator classifies both real data and fake data from the generator
- The discriminator loss penalizes the discriminator for misclassifying a real instance as fake or a fake instance as real
- The discriminator updates its weights through backpropagation from the discriminator loss through the discriminator network

GENERATOR

The generator part of a GAN learns to create fake data by incorporating feedback from the discriminator

It learns to make the discriminator classify its output as real

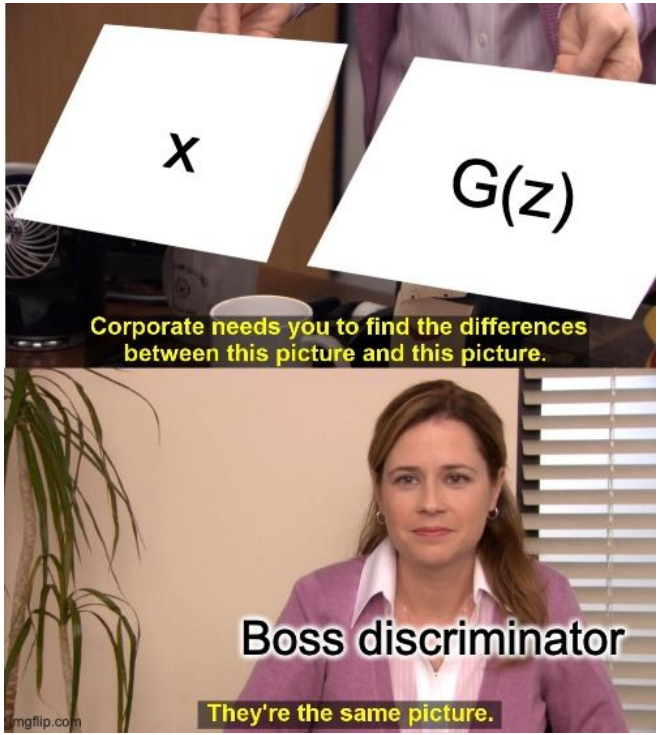


Generator Training

- Sample random noise
- Produce generator output from sampled random noise
- Get discriminator "Real" or "Fake" classification for generator output
- Calculate loss from discriminator classification
- Backpropagate through both the discriminator and generator to obtain gradients
- Use gradients to change only the generator weights

GAN Training

1. The discriminator trains for one or more epochs
2. The generator trains for one or more epochs
3. Repeat steps 1 and 2 to continue to train the generator and discriminator networks
4. Careful with Convergence as not to overtrain the generator otherwise generator will start training with less meaningful discriminator



Loss Function

Original GAN: minimax Loss

$$E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$

- 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

The generator can't directly affect the log(D(x)) term in the function, so, for the generator, minimizing the loss is equivalent to minimizing log(1 - D(G(z)))

Modified MiniMax Loss

$$E_x[log(D(x))] + E_z[log(1 - D(G(z)))]$$

The original GAN paper notes that the above minimax loss function can cause the GAN to get stuck in the early stages of GAN training when the discriminator's job is very easy

The paper therefore suggests modifying the generator loss so that the generator tries to maximize log D(G(z))

Component	Original GAN	WGAN
Discriminator	Classifies real (1) vs fake (0)	Called a **Critic** — outputs **a score** (no sigmoid)
Output Range	[0, 1] (probability)	$\mathbb{R}$ (real numbers; no restriction)
Loss Type	Binary Cross-Entropy	Wasserstein (EM distance approximation)
Stability	Can be unstable (saturates, vanishes)	More stable gradients

Wasserstein Loss

Here no classification for discriminator.

Discriminator becomes “Critic”: `Critic Loss: D(x) - D(G(z))`

The discriminator tries to maximize this function

In other words, it tries to maximize the difference between its output on real instances and its output on fake instances

`Generator Loss: D(G(z))`

The generator tries to maximize this function. In other words, It tries to maximize the discriminator's output for its fake instances

Advantages of Wasserstein Loss

Wasserstein GANs are less vulnerable to getting stuck than minimax based GANs and avoid problems with vanishing gradients

The earth mover distance also has the advantage of being a true metric: a measure of distance in a space of probability distributions

Cross-entropy is not a metric in this sense.

Common problems in GAN

Problem #1: Vanishing Gradient: Research has suggested that if your discriminator is too good, then generator training can fail due to vanishing gradients

In effect, an optimal discriminator doesn't provide enough information for the generator to make progress

Sol: Wasserstein loss and Modified minimax loss

Common problems in GAN

Problem #2: Mode Collapse: When the generators rotate through a small set of output types, this form of GAN failure is called Mode Collapse

In effect, an optimal discriminator doesn't provide enough information for the generator to make progress

Sol: Wasserstein loss and Modified minimax loss

Common problems in GAN

Problem #3: Failure To Converge: Often problem in GAN

Sol:

(i) Adding noise to discriminator inputs

(ii) Penalizing discriminator weights

References

- https://media.licdn.com/dms/image/v2/D4E22AQFi4YW6Uchugw/feedshare-shrink_800/B4EZV_6KoZHUAg-/0/1741607710656?e=2147483647&v=beta&t=wJgeUcKADM9xp6vf48AoG_EcVVc8XYCH8UkPqz0KtHM
- https://developers.google.com/machine-learning/gan

Resources Of GenAI

https://community.deeplearning.ai/t/generative-ai-for-everyone-lecture-notes/481740