Generative Adversarial Network (GAN)

What is GAN?

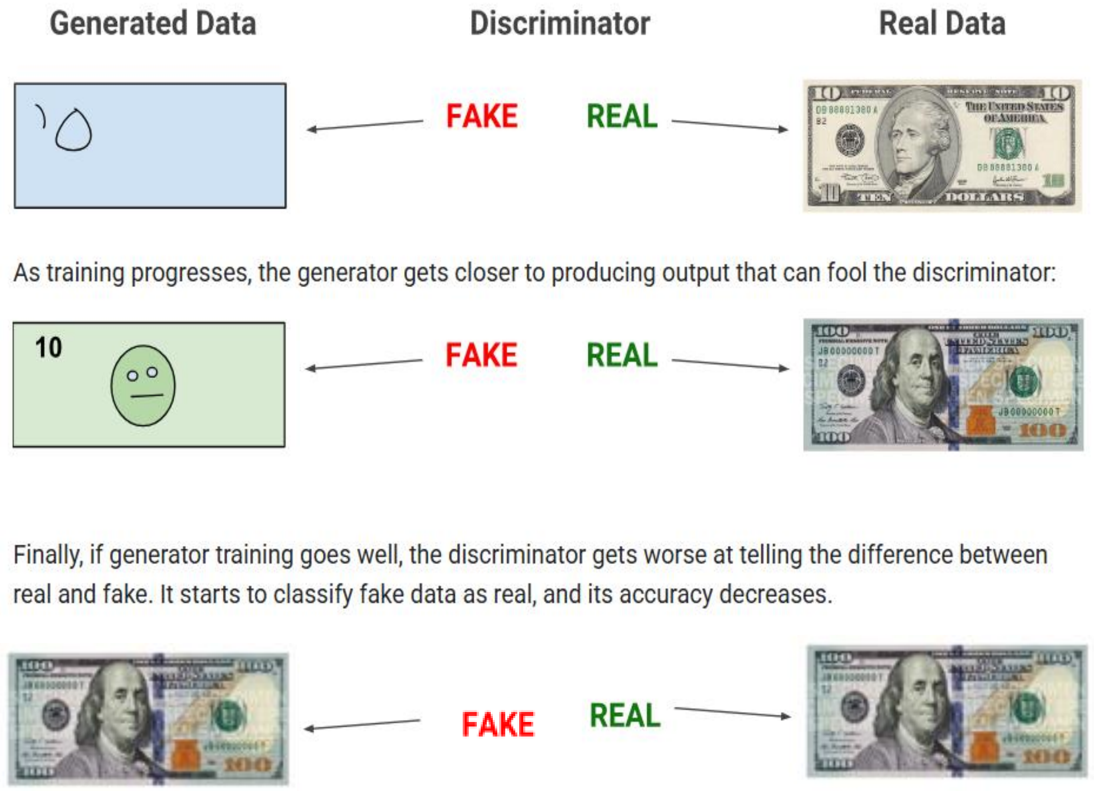
- Generative Adversarial Networks (GANs) are a powerful class of neural networks used for unsupervised learning.
- Developed and introduced by Ian J. Goodfellow in 2014.
- It is a type of machine learning that learns to create new data that is similar to existing data. It does this by learning the patterns in the existing data and then using those patterns to generate new data. The new data is not simply a copy of the existing data, but it is unique and looks like original.
- GANs can create content based on the data they are trained on, following a Learn-Generate-Improve process.

How does GAN work?

- The purpose of a GAN is to generate realistic data, typically images, that resemble a given dataset.
- GANs consist of two neural networks: a generator and a discriminator
- The generator network takes random input (typically noise) and generates samples resembling the training data, such as images, text, or audio. Its goal is to produce samples indistinguishable from real data.
- The discriminator network, on the other hand, distinguishes between real and generated samples, classifying real data as real and generated data as fake.
- The training process involves an adversarial game (**minimax game**) where the generator tries to fool the discriminator, and the discriminator aims to improve its ability to distinguish real from generated data.
  - The **generator** tries to maximize the discriminator's **error**.
  - The **discriminator** tries to minimize its own error (correctly classifying real vs. fake).

- Through continuous training, the generator improves its ability to create realistic data, while the discriminator struggles more with differentiation.
- The ultimate goal is for the generator to create data so realistic that the discriminator cannot distinguish it from real data.

- Generative Adversarial Network (GAN) can be explained using the concept of counterfeiters and police.
- In this analogy, the generative model can be thought of as a team of counterfeiters attempting to produce fake currency that is indistinguishable from genuine currency, aiming to use it without detection.
- Conversely, the discriminative model is compared to the police, whose role is to detect the fake currency.
- The competition between these two models drives both to improve their methods until the counterfeits become indistinguishable from the real currency.
- This dynamic is central to how GANs function, with the generator (counterfeiters) learning to create more realistic data and the discriminator (police) becoming better at distinguishing between real and generated data.



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.

Generative Adversarial Networks

Generative Models

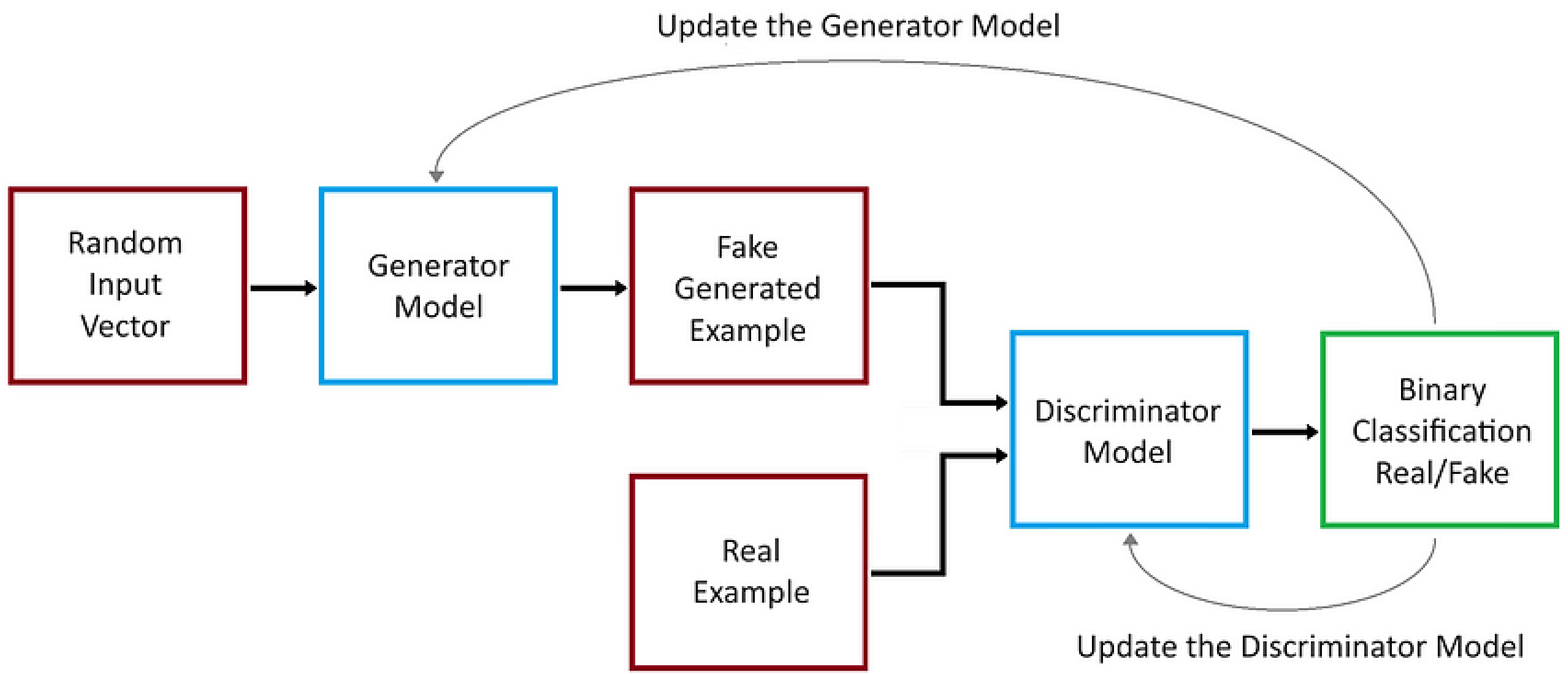
We try to learn the underlying the distribution from which our dataset comes from.
Eg: Variational AutoEncoders (VAE)

Adversarial Training

GANS are made up of two competing networks (adversaries) that are trying beat each other.

Neural Networks

Architecture


Update the Generator Model
Random Input Vector
Generator Model
Fake Generated Example
Real Example
Discriminator Model
Binary Classification Real/Fake
Update the Discriminator Model

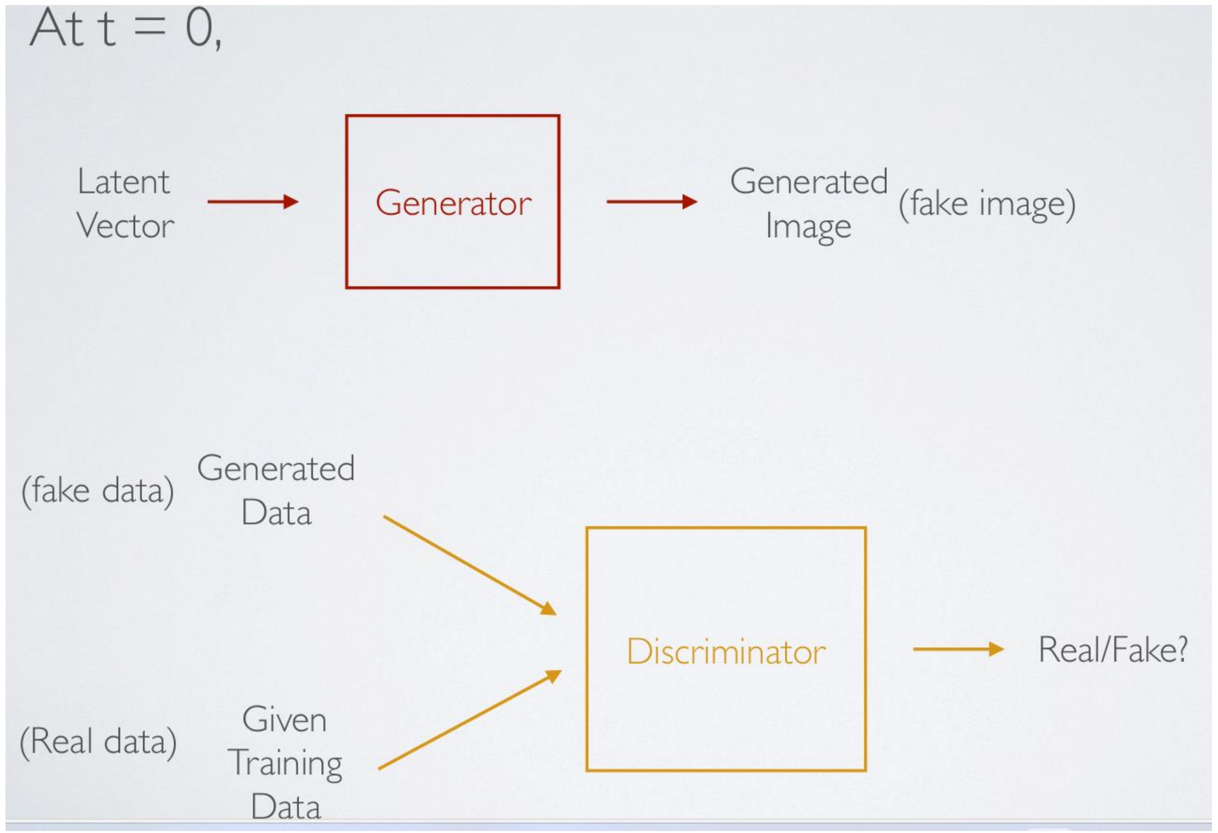
GAN Training: Adversarial Training process

- The generator and the discriminator have different training processes.
- GAN training proceeds in alternating periods:

    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.

- We keep the generator constant during the discriminator training phase. As discriminator training tries to figure out how to distinguish real data from fake, it has to learn how to recognize the generator's flaws.
- Similarly, we keep the discriminator constant during the generator training phase. Otherwise the generator would be trying to hit a moving target and might never converge.

- **Training the Discriminator (Discriminator is trained first)**
  - The Discriminator is a Binary classifier.
  - The Discriminator has two class - Real and Fake.
  - The data for Real class: THE TRAINING DATA
  - The data for Fake class? -> generate from the Generator



• Training the generator

1. Start with noise:

- Sample a random noise vector `z` from a prior distribution (e.g., a normal distribution).
- Pass it through the generator: `G(z) → fake data`.

2. Discriminator evaluates fake data:

- The discriminator assigns a probability that this fake data is **real**: `D(G(z))`.

3. Define the generator loss:

There are two common versions of the generator loss:

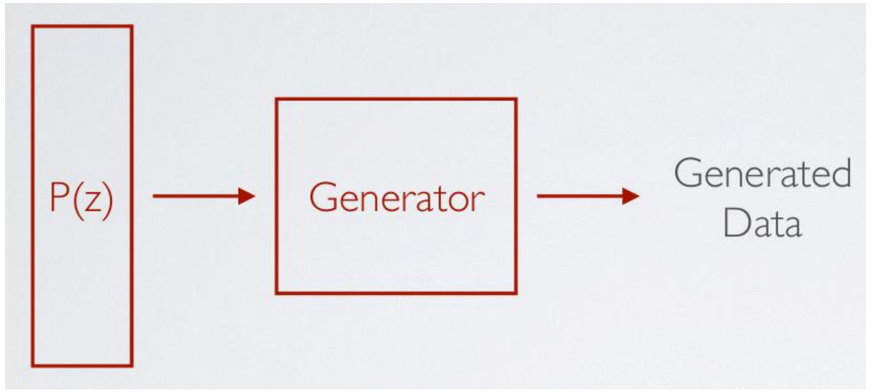
a. Non-saturating loss (more commonly used, better gradients):

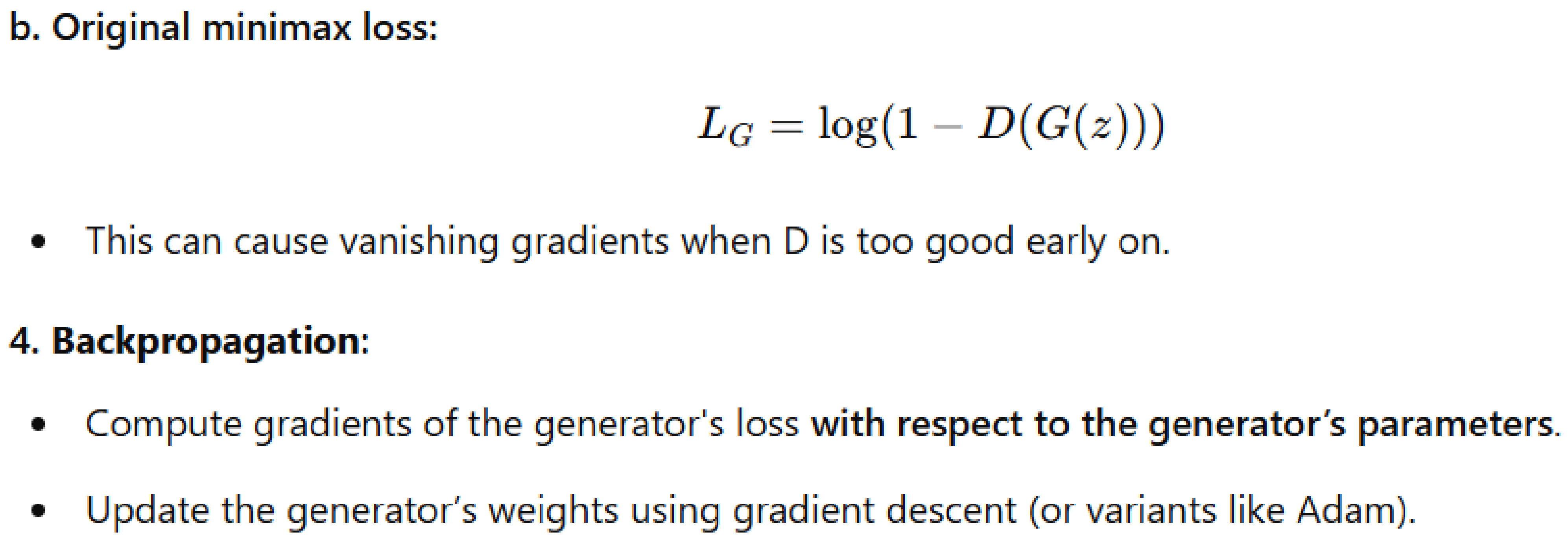
$$L_G = -\log(D(G(z)))$$

- The generator wants the discriminator to output a high value for fake data (i.e., treat it as real).

🎲 Input: Random Noise Vector z

- Notation: $z \sim p_z(z)$
- Common distributions:
  - Normal (Gaussian): $z \sim \mathcal{N}(0, I)$
  - Uniform: $z \sim \mathcal{U}(-1, 1)$





b. Original minimax loss:

$$L_G = \log(1 - D(G(z)))$$

- This can cause vanishing gradients when D is too good early on.

4. Backpropagation:

- Compute gradients of the generator's loss **with respect to the generator's parameters**.
- Update the generator's weights using gradient descent (or variants like Adam).

- If the discriminator easily spots the fakes, it means the generator is doing a poor job.
- So, the generator updates its parameters to **generate more realistic samples**, thus increasing the discriminator's confusion.

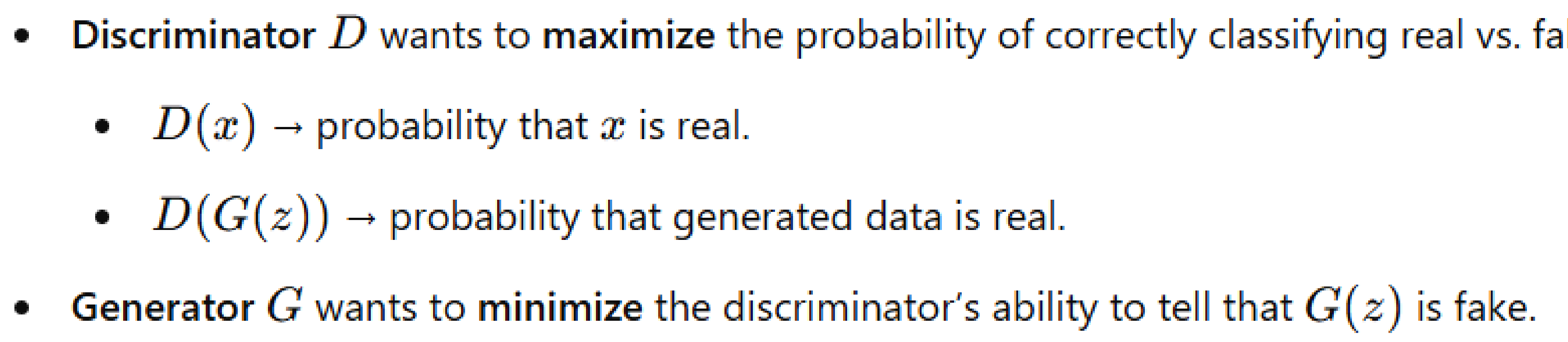
🔄 Training Loop Summary

1. Train the discriminator:
    - On real data → label 1.
    - On fake data from generator → label 0.
2. Train the generator:
    - Generate fake data.
    - Pass fake data to discriminator.
    - Calculate generator loss (using D’s output).
    - Update G to increase D’s error.

This adversarial dynamic continues until an equilibrium is (ideally) reached — where generated data is indistinguishable from real data.

Loss function

- A GAN can have two loss functions: one for generator training and one for discriminator training. The two loss functions work together to reflect a distance measure between probability distributions
- Generator loss measures how well the generator fools the discriminator, typically aiming to minimize the discriminator’s ability to detect fake data.
- Discriminator loss measures how well the discriminator differentiates between real and fake data, aiming to maximize correct classifications of real and fake samples.



- **Discriminator** D wants to **maximize** the probability of correctly classifying real vs. fake:
  - $D(x)$ → probability that x is real.
  - $D(G(z))$ → probability that generated data is real.
- **Generator** G wants to **minimize** the discriminator's ability to tell that $G(z)$ is fake.

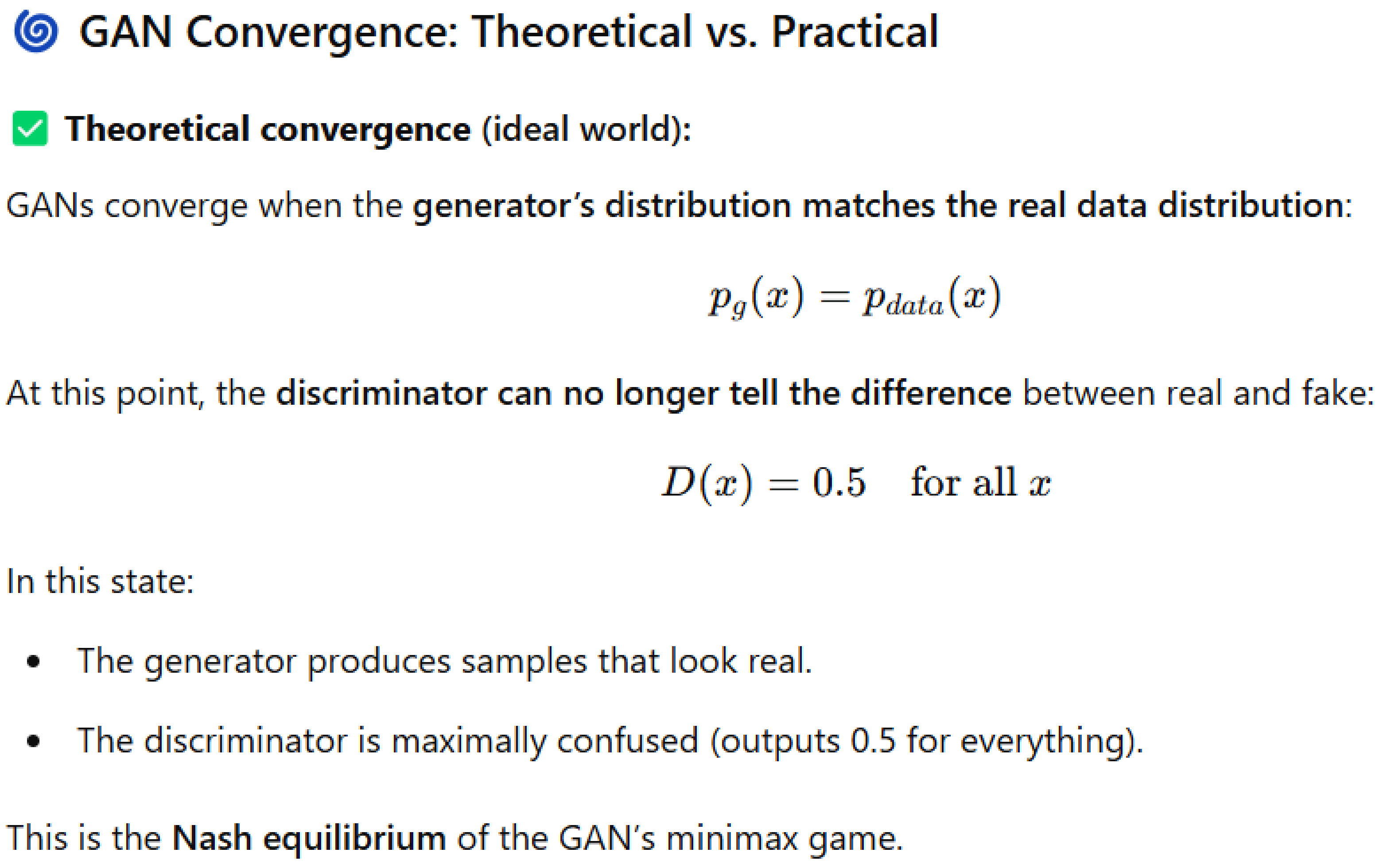
🧮 Minimax Loss Function

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

- The **discriminator** maximizes this:
  - It wants to give **high scores** to real data ($D(x) \to 1$)
  - And **low scores** to fake data ($D(G(z)) \to 0$)
- The **generator** minimizes it:
  - It wants $D(G(z)) \to 1$, so that the discriminator is **fooled**.

Convergence

- As the generator improves with training, the discriminator performance gets worse because the discriminator can't easily tell the difference between real and fake.



🌀 GAN Convergence: Theoretical vs. Practical

✅ **Theoretical convergence** (ideal world):

GANs converge when the **generator's distribution matches the real data distribution**:

$$p_g(x) = p_{data}(x)$$

At this point, the **discriminator can no longer tell the difference** between real and fake:

$$D(x) = 0.5 \quad \text{for all } x$$

In this state:

- The generator produces samples that look real.
- The discriminator is maximally confused (outputs 0.5 for everything).

This is the **Nash equilibrium** of the GAN's minimax game.

Convergence is hard to define or detect. Why?

1. **No clear loss metric**:
   - Unlike supervised learning (where loss decreases over time), GAN losses **don’t directly correlate** with sample quality.
   - The generator loss can oscillate or even increase while sample quality improves.
2. **Mode collapse**:
   - The generator might get stuck producing a limited variety of outputs (e.g., the same image over and over).
3. **Discriminator too strong**:
   - If the discriminator is perfect, gradients vanish → the generator can’t learn.
4. **Oscillations**:
   - G and D keep trying to outsmart each other, leading to unstable dynamics.

🔍 So when *do* we say a GAN has converged?

Usually by **visual inspection** or **metrics like**:

- **FID (Fréchet Inception Distance)**: Lower = better match to real data.
- **IS (Inception Score)**: Measures both image quality and diversity.
- **Precision/Recall for GANs**: Measures how well G covers the true data distribution.

🧪 In practice:

- You train until **samples look good and diverse**.
- You often save **checkpoints** and pick the best model **manually or via metrics**.

- https://www.analyticsvidhya.com/blog/2021/10/an-end-to-end-introduction-to-generative-adversarial-networksgans/

- https://deeplearning.cs.cmu.edu/F20/document/recitation/GANS-ppt.pdf