Generative Adversarial Networks (GAN)

Seminar Intelligent Systems (SS 25)

Ullin Noe Yanou

University of Duisburg-Essen
Faculty of Computer Science
Department of Software Engineering
Chair of Intelligent Systems

09. July 2025

Overview

1. **Motivation & Problem Statement**
2. **Generative Adversarial Networks (GANs)**
3. **Wasserstein GANs (WGANs)**
4. **Conditional GANs (Brief Overview)**
5. **Evaluation & Comparison**
6. **Outlook & Conclusion**

What if AI is more powerful than you think?

Machines no longer just recognize – they **create**.

- AI can generate realistic faces, art, music, text...
- Deepfakes, AI art, synthetic medical data are reality.
- Behind many of these breakthroughs: **GANs**.

But are these people real? Look for yourself:

AI Can Create Faces Like These




These people don't exist.

Generated by GANs (thispersondoesnotexist.com).

Problem Statement: Why Improving GANs is Hard

GANs are powerful, but difficult to control.

- **Instability:** Training often fails or oscillates.
- **Mode Collapse:** Generator produces only few, repetitive outputs.
- **Unreliable Loss:** Loss curves don't clearly reflect progress.

Why? It's a fragile two-player game:

- If the **Discriminator** is too strong — Generator stops learning.
- If the **Generator** gets ahead — Discriminator can't detect fakes.

Goal: Make GANs more stable, interpretable, and reliable.

GANs: Foundation & Structure (2/6)

Two networks in competition — the foundation of GANs.

In this part:

- What are GANs?
- The Minimax Game
- GAN Training Algorithm (Goodfellow et al., 2014)
- Training Challenges
- Practical Example: MNIST
- Live Example

What is a GAN?

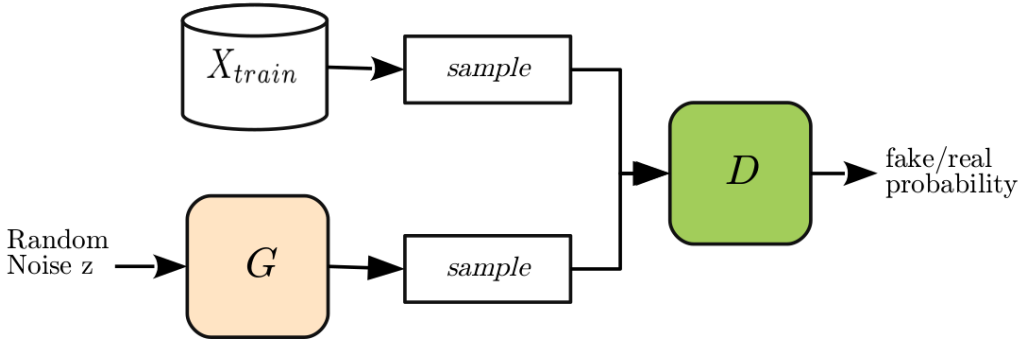
GAN = Generative Adversarial Network

Two neural networks, playing a game:

- **Generator (G):** Creates synthetic data (e.g., fake images)
- **Discriminator (D):** Judges if data is real or generated

Goal: The Generator gets so good that even the Discriminator is fooled.

GAN Fundamentals: Visual Illustration



Simplified architecture of a GAN: The Generator and Discriminator compete in a learning loop.

GANs: The Minimax Game

Training as a mathematical competition:

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

Intuition:

- D: Maximizes ability to detect fakes.
- G: Minimizes chance of getting caught.

A continuous battle — both networks improve step by step.

Minimax Game: Understanding the Formula

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

What the symbols mean:

- $\mathbb{E}$: Expected value — average over all possible examples
- x : Real data example (e.g., real image)
- $p_{\text{data}}(x)$: Distribution of real data
- z : Random noise input for generator
- $G(z)$: Generated (fake) data
- $D(x)$: Discriminator's estimate that x is real (value between 0 and 1)

Minimax Game: Who Plays Which Role?

Two networks, two opposite goals:

- **Discriminator (D)** wants:
  - $D(x) \rightarrow 1$ for real data
  - $D(G(z)) \rightarrow 0$ for fake data
- **Generator (G)** wants:
  - Generate fake data $G(z)$ so realistic that $D(G(z)) \rightarrow 1$

Why This Formula? Why Logarithms?

The log **functions make learning stable:**

- $\log D(x)$: High reward when D is confident and correct.
- $\log(1 - D(G(z)))$: High reward when D catches fake data.

Without log:

- Gradients (learning signals) would vanish or explode.
- Training becomes unstable.

With log:

- Smooth learning.
- Penalizes confident wrong answers strongly.

Goodfellow's trick to make GANs learn reliably.

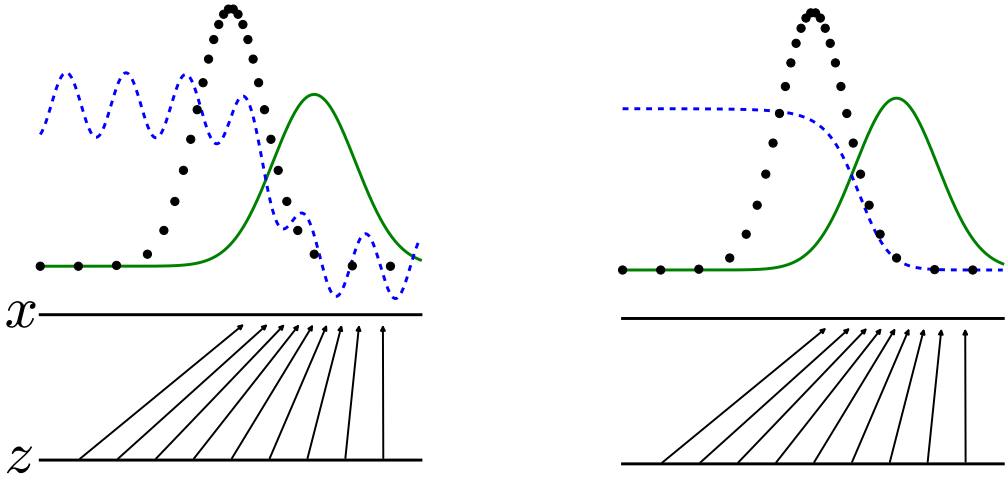
GAN Training Algorithm

Algorithm 1: GANs training algorithm (adapted from Goodfellow et al.2014)

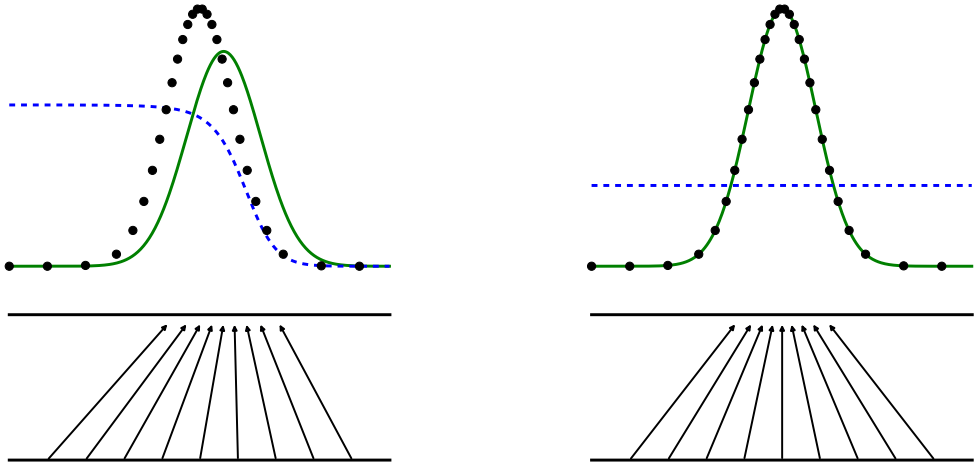
Input: Real data distribution $p_{\text{data}}(x)$, noise prior $p_z(z)$, learning rate α, mini-batch size m
Output: Trained generator G

Initialize parameters θ_D, θ_G
repeat
  for $t = 1$ **to** k **do**
    Sample mini-batch $\{x^{(i)}\}_{i=1}^{m}$ from $p_{\text{data}}(x)$
    Sample noise $\{z^{(i)}\}_{i=1}^{m}$ from $p_z(z)$
    Compute discriminator loss:
      $L_D = -\frac{1}{m}\sum_{i=1}^{m}[\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))]$
    Update θ_D by descending the gradient of L_D
  Sample noise $\{z^{(i)}\}_{i=1}^{m}$ from $p_z(z)$
  Compute generator loss:
    $L_G = -\frac{1}{m}\sum_{i=1}^{m}\log(D(G(z^{(i)})))$
  Update θ_G by descending the gradient of L_G
until *convergence*

Training Evolution (1/2)



Training Evolution (2/2)



GAN Training — How Many Updates?

Balance is crucial:

- Too many D updates:
  - D becomes unbeatable
  - G gets no useful feedback
- Too few D updates:
  - D stays weak
  - G learns wrong patterns

Empirical Rule: Use $k = 1$ discriminator update per generator update.

GAN Training Challenges

Why is GAN training so difficult?

- Networks must learn **together** — timing is crucial.
- **Mode Collapse:** Generator produces only a few repeating outputs.
- **Vanishing Gradients:** D gets too good, G receives no feedback.
- **Unreliable Loss:** Loss curves don't clearly reflect quality.

"Like a beginner always playing chess against a world champion — no progress possible."

MNIST: Dataset & GAN Experiment Setup

What is MNIST?

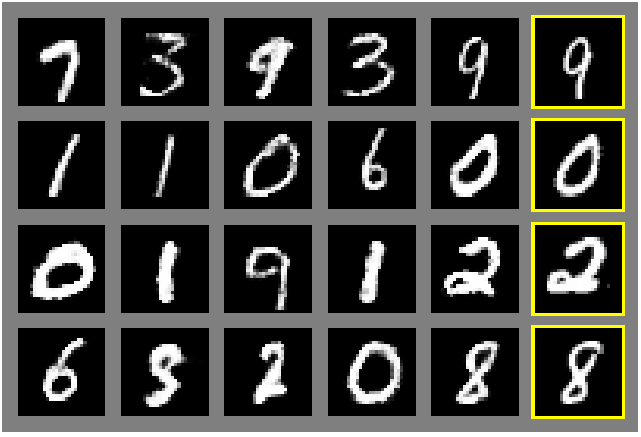
- Benchmark dataset of handwritten digits (0 to 9)
- 70,000 grayscale images — each 28x28 pixels
- Widely used for testing machine learning models

GAN Experiment Details:

- Generator learns to create digit-like images from random noise
- Discriminator distinguishes between real MNIST digits and generated fakes
- After around **50 training epochs**, generated samples can resemble real digits

Source: MNIST dataset — LeCun et al. (1998)

GAN Example: MNIST Digit Generation



Generated handwritten digits with a simple GAN

Source: Goodfellow et al. (2014)

MNIST GAN Results — Interpretation

What you see:

- Some digits look **clear and realistic** — GAN learned the structure.
- **Yellow boxes:** Highlight typical mistakes — blurry or distorted digits.
- Not all outputs are perfect — shows GAN's limits.

Takeaway: Even a simple GAN can generate impressive results, but **training remains tricky**.

Live Example

Live Example in Python

(Switch to Python code demonstration)

Conditional GANs (cGAN)

Guided Data Generation with Labels

Conditional GANs (cGANs)

Limitation of classic GANs: The generator produces random samples — we have no control over what is generated.

Solution: Conditional GANs (Mirza & Osindero, 2014) allow us to guide generation by providing additional information (e.g., class labels).

Key idea:

- Both Generator and Discriminator receive an extra input y (e.g., digit label, category).
- The Generator learns to produce samples *conditioned* on y.
- The Discriminator checks if the sample matches the condition.

Wasserstein GANs: Overview (3/6)

Improving stability through a new distance metric.

In this part:

- Why WGAN?
- Earth Mover Distance
- WGAN Training Algorithm (Arjovsky et al., 2017)
- Practical Example

Why WGAN?

Classic GANs often struggle with:

- Unstable training
- Mode collapse (lack of diversity)
- Loss values that don't reflect real progress

WGAN (Arjovsky et al., 2017) offers:

- The **Earth Mover (Wasserstein) Distance**
- More stable gradients
- Loss that correlates with sample quality

What is the Earth Mover Distance (EMD)?

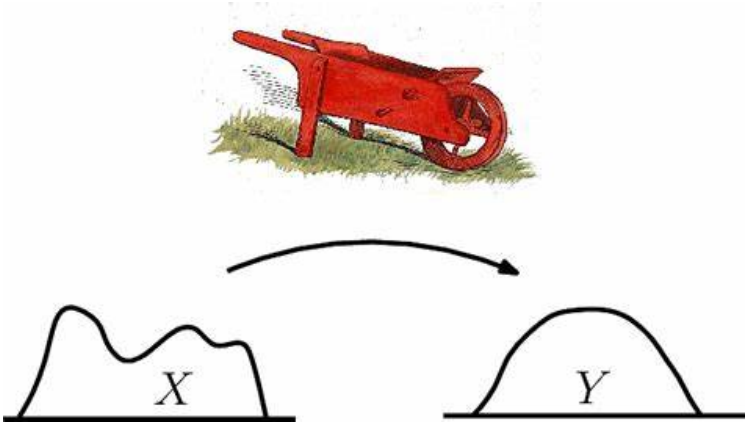
Goal: Measure how similar two data distributions are.

- Imagine moving "piles of mass" to transform one distribution into another.
- The **Earth Mover Distance (EMD)** tells us:
  "How much effort is needed to turn one distribution into the other?"
- The less effort, the more similar the distributions.

Why is it useful?

- It gives a stable, intuitive measure — even when distributions barely overlap.
- Better suited for GAN training than other metrics like the *Shannon Divergence*.

Earth Mover Distance: Visual



Moving "dirt" from one distribution to another — the less effort, the more similar they are.

What You See in the Visual

Distributions as piles of mass:

- Mass from distribution X is moved to match Y.
- The **wheelbarrow** symbolizes the transport effort.
- Less distance or less mass to move $\rightarrow$ more similar distributions.

EMD gives an intuitive way to measure how close real and generated data are.

Earth Mover Distance: Formal Definition

Mathematical Formulation:

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

The minimal "effort" to turn generated data P_g into real data P_r.

Earth Mover Distance: Explaining the Equation

What the terms mean:

- P_r: The true distribution of real data (e.g., real images)
- P_g: The distribution of data generated by the GAN
- $\Pi(P_r, P_g)$: The set of all possible ways to "pair up" real and generated samples
- γ: One specific pairing strategy — describes exactly how much of each real sample gets "matched" to each generated sample

We search for the pairing that moves the least amount of "mass" over the shortest distance — that's the Earth Mover Distance.

Why Earth Mover Distance?

Advantages over classic GAN loss:

- Always defined — even when distributions don't overlap
- Provides **meaningful gradients** everywhere
- Strong correlation with visual sample quality
- Leads to more stable GAN training

WGAN uses EMD to overcome classic GAN instability.

Jensen–Shannon Divergence: Measuring Similarity

JSD measures how different two probability distributions are.

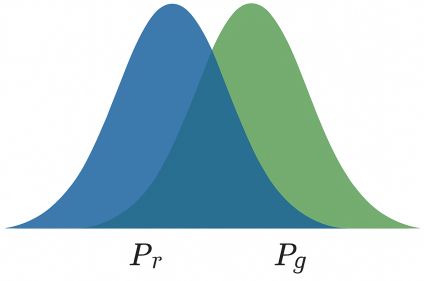
Classic GANs minimize:

$$D_{\mathrm{JS}}(P_r \| P_g)$$

Where:

- P_r: Real data distribution
- P_g: Generated data distribution

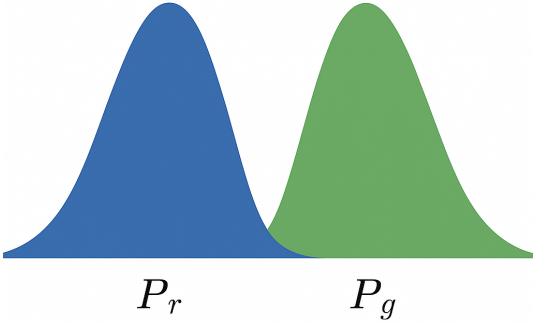
JSD Example: Overlapping Distributions



Interpretation:

- Two distributions overlap significantly.
- JSD is small — model is doing well.

JSD Problem: Non-Overlapping Distributions



The Issue:

- JSD becomes constant.
- No gradient — generator stops learning.

WGAN Training Algorithm

Algorithm 2: WGAN Training Algorithm (Arjovsky et al., 2017)

Data distribution p_{data}, noise prior $p_z(z)$, learning rate α, batch size m, number of critic updates n_{critic}, clipping parameter c Trained generator G Initialize critic parameters w, generator parameters θ ;
repeat
  for $t = 1$ *to* n_{critic} **do**
    Sample real data $\{x^{(i)}\} \sim p_{\text{data}}(x)$;
    Sample noise $\{z^{(i)}\} \sim p_z(z)$;
    Compute critic loss $L_D = -\frac{1}{m}\sum[D(x^{(i)}) - D(G(z^{(i)}))]$;
    Update w using RMSProp or Adam on L_D ;
    Clip weights: $w \leftarrow \text{clip}(w, -c, c)$;
  end
  Sample noise $\{z^{(i)}\} \sim p_z(z)$;
  Compute generator loss $L_G = -\frac{1}{m}\sum D(G(z^{(i)}))$;
  Update generator parameters θ on L_G ;
until *convergence*;

56

WGAN Algorithm: Step by Step

What's Happening?

→ The **critic** f assigns higher scores to real data, lower to fakes.

→ The **generator** G tries to increase its score by fooling the critic.

→ Weight clipping keeps the critic 1-Lipschitz (required for valid EMD estimation).

Why multiple critic updates?

- The critic must reliably estimate the EMD.
- Generator only updates when the critic is strong enough.

Why 5 Critic Updates in WGAN?

In classic GANs:

- Discriminator and Generator alternate updates.
- Usually, $k = 1$: one discriminator update per generator update.

In WGAN:

- The Critic estimates the Wasserstein distance.
- A good distance estimate is crucial for stable learning.
- **Empirically:** $n_{\text{critic}} = 5$ critic updates per generator update produce:
  - More accurate distance estimate.
  - More stable and meaningful generator updates.

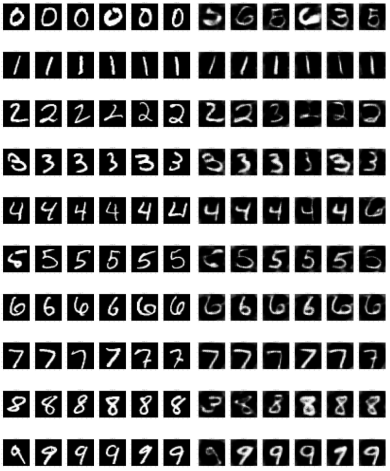
The Critic needs more time to do its job properly so more updates.

WGAN Experiment: MNIST Digit Generation

Does WGAN generate better digits?

We train a WGAN on the MNIST dataset to evaluate its stability and output quality.

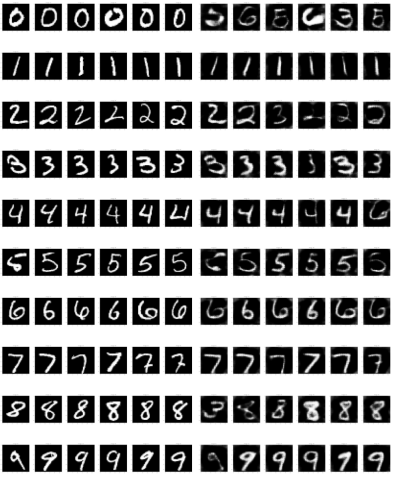
WGAN Results: Generated MNIST Digits



Digits generated by WGAN after training on MNIST.

Visual Comparison: GAN vs. WGAN on MNIST

Classic GAN



WGAN

Why Does the WGAN Image Look Better?

- **GAN:** Repetitive digits, signs of mode collapse
- **GAN:** Some blurry, distorted shapes
- **WGAN:** Greater diversity — all digit classes appear
- **WGAN:** Sharper, cleaner digit outlines

WGAN produces more stable, diverse, and realistic results on MNIST.

Fréchet Inception Distance (FID)

What is FID?

- Compares real and generated images in feature space
- Lower FID = more realistic, diverse samples
- Widely used for GAN evaluation

Results on MNIST:

Model	**FID Score ↓**
GAN	29.4
cGAN (Appendix)	*22.7*
wGAN	14.3

Model Comparison: Summary Table

Model	Visual	Diversity	Stability	Loss Interpretability
GAN	Medium	Low	Poor	Poor
cGAN	Medium	Medium	Poor	Poor
wGAN	High	High	Good	Good

WGAN have best performance across quality, diversity, and stability.

Outlook & Future Directions

- **WGAN-GP:** Improved stability with gradient penalty
- **Diffusion Models:** Even better image quality (e.g., DALL·E 2)
- **Cross-modal Generation:** From text or audio to images
- **Scientific Uses:** Medicine, molecules, climate simulation
- **Hybrid Models:** GANs meet VAEs and Transformers

GANs are just the beginning — generative AI keeps evolving.

Conclusion

- GANs enable machines to create — images, audio, data
- But classic GANs struggle with stability and control
- WGANs improve training through better distance measures
- cGANs allow control over what is generated

Takeaway: GANs are powerful — but understanding their foundations is key to using them effectively.

Credits

- **Graphical Visuals:** Most illustrations and visual comparisons were generated with **ChatGPT's image generation tool**.
- **Algorithms:** GAN training algorithm adapted from **Goodfellow et al. (2014)** and WGAN training algorithm adapted from **Arjovsky et al. (2017)**.
- **GAN Architecture & Training Evolution:** Reproduced from **Goodfellow et al. (2014)**.
- **MNIST Dataset: Reproduced from author ashinbabu[Github] and WGANs Arjovsky, Chintala and Bottou and respectively.**
- **AI-generated Faces:** Downloaded from: **thispersondoesnotexist.com**.

This work is a combination of research, reproduction, and educational visualization.