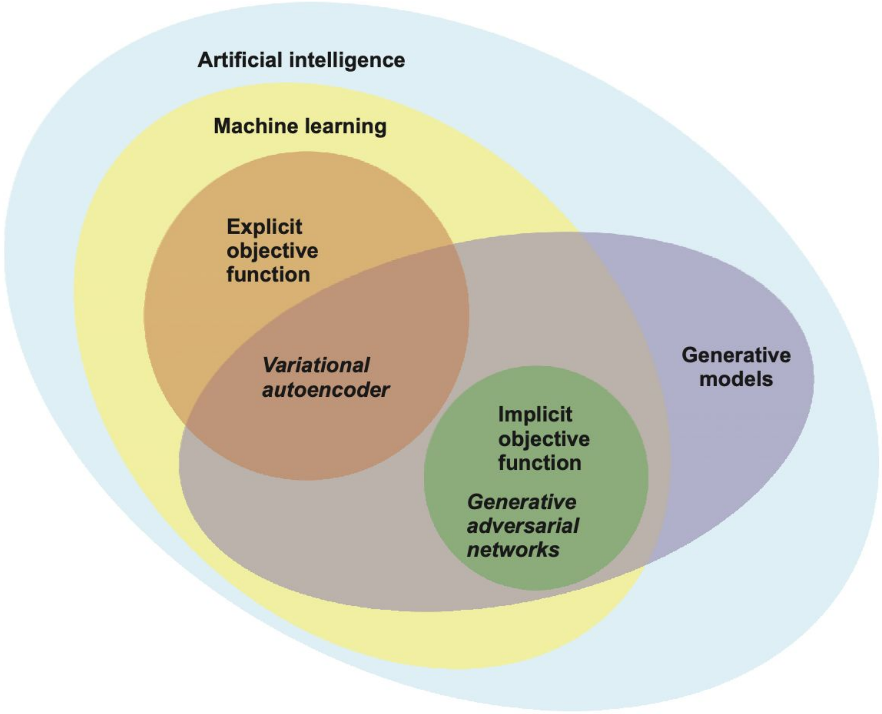
All about GANs - DCGAN & Training Challenges

From Autoencoders to GANs

Autoencoders: compress → reconstruct; learn a **compact representation** that retains essentials and minimizes reconstruction error.

GANs: two networks play "**forger vs. inspector**"—the generator creates candidates, the discriminator critiques them.



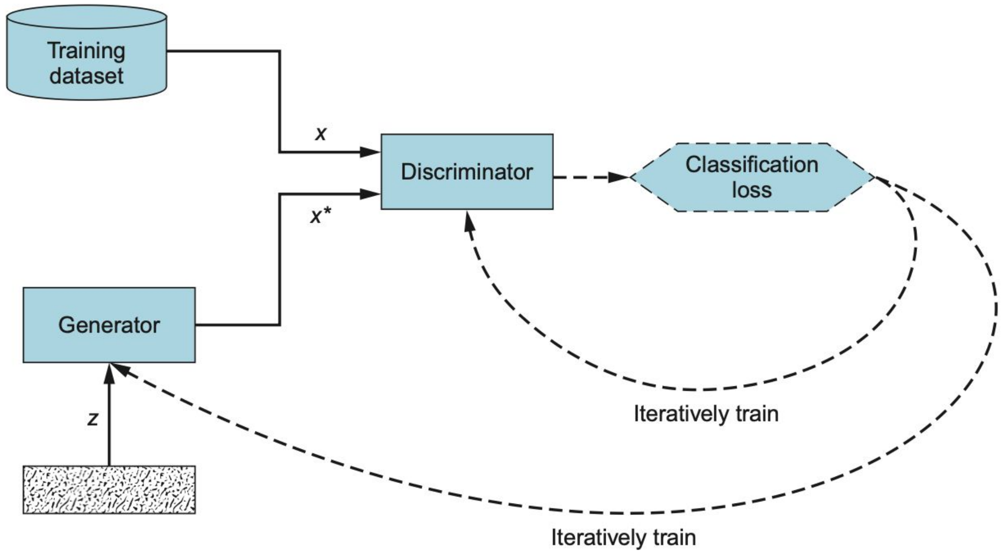
Where they sit in AI/ML

Aspect	Autoencoder	GAN
Objective	Minimize reconstruction loss (e.g., MSE, BCE).	Minimax game between generator and discriminator.
Parts	Encoder + Decoder.	Generator + Discriminator.
Latent	Explicit code z, interpretable for tasks.	Implicit, less interpretable.
Training	Single end-to-end loss, typically more stable.	Adversarial, sensitive to tuning and mode collapse.
Uses	Compression, denoising, feature extraction, anomaly detection.	Photorealistic synthesis, image-to-image translation, data augmentation.

Cost Functions of GANs

What drives learning in GANs: the **cost functions** of Generator J(G) and Discriminator J(D).

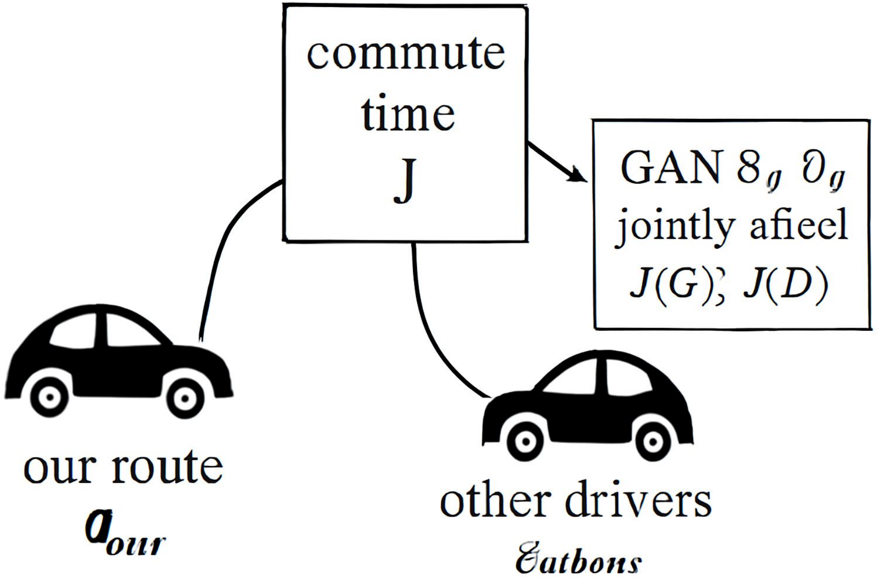
Key idea: adversarial objectives tied together in a **minimax game.**



Intuition First — The Commute Analogy

Traditional NN: $J(\theta)$ depends only on own parameters.

GANs: each player's cost depends on both players' parameters $J(G)(\theta(G), \theta(D))$, $J(D)(\theta(G), \theta(D))$.



The Road to DCGAN

GANs (2014): First attempt at AI-generated images → blurry, low-quality.

Early attempts to combine **ConvNets with GANs** struggled due to training instability.

LAPGAN (2015): Cascade of ConvNets → better images but complex and slow.

GAN (2014) → LAPGAN (2015) → DCGAN (2016).

The Breakthrough – DCGAN (2016)

Authors: **Radford, Metz, Chintala.**

- First full-scale ConvNet GAN.

Key innovations:

- Batch Normalization → stabilizes training.
- Strided Convolutions instead of pooling.
- Leaky ReLU → better gradient flow.

Result: Scalable, high-quality image generation without complex multi-network setups.

Why DCGAN Matters

Made **ConvNet + GAN i**ntegration practical.

High-quality image generation set the stage for modern GANs (**StyleGAN, Progressive GAN**).

Lesson: Stabilization + architecture design = better models.

GAN Training – The Big Picture

Challenge 1 – Discriminator Overpowers Generator

When the discriminator becomes too strong:

- Generator gets weak feedback.
- Gradients vanish → no learning happens.

Analogy: Teacher is too strict → student gives up trying.

Solutions – Weakening the Discriminator

Increase Dropout in discriminator

Reduce discriminator learning rate

Reduce number of convolutional filters

Add noise to labels

Randomly flip labels during training

Challenge 2 – Generator Overpowers Discriminator (Mode Collapse)

Generator finds a “shortcut” → produces same image repeatedly

Discriminator too weak to penalize it

Analogy: Student memorizes answers without learning concepts

Solutions – Strengthening the Discriminator

Reduce generator’s “dominance” using the opposite of previous suggestions:

- Reduce generator learning rate
- Increase batch size
- Make discriminator slightly stronger

Challenge 3 – Uninformative Loss

Generator loss ≠ image quality

Why? Discriminator changes every batch → constantly shifting "grading scale"

Challenge 4 – Hyperparameter Sensitivity

GANs have many sensitive hyperparameters:

- Network architecture, batch norm, dropout
- Learning rates, activation functions

Kernel sizes, strides, latent space size

- Small changes → large effects

Analogy: Tuning a musical instrument; even a small misalignment creates noise