Generative Adversarial Networks (GANs)

# 1. Introduction

Generative Adversarial Networks (GANs) are a type of generative model capable of producing entirely new data instances from a training distribution.

Example: If trained on images of horses, a GAN can generate new horse images not present in the original dataset.

Other popular generative models include diffusion models (e.g., Stable Diffusion, DALL·E), but GANs remain a foundational architecture in deep learning.

# 2. GAN Architecture

A GAN consists of two neural networks that compete against each other:

- Generator (G)

- Input: Random noise (Gaussian noise, denoted as z).

- Output: Synthetic (generated) image.

- Objective: Fool the discriminator by producing images that appear real.

- Discriminator (D)

- Input: Either real images (from dataset x) or fake images (G(z)).

- Output: Probability score (1 = real, 0 = fake).

- Objective: Correctly classify real vs. fake images.

This setup creates an adversarial game:

- The generator improves by learning how to fool the discriminator.

- The discriminator improves by detecting fakes more accurately.

# 3. Ideal Objectives

Discriminator’s Ideal Case

- Outputs 1 for real images.

- Outputs 0 for fake (generated) images.

Generator’s Ideal Case

- Produces fake images that make the discriminator output 1 (believing they are real).

Thus, their objectives are directly opposing. Training continues until the generator produces images that are indistinguishable from real ones.

# 4. Mathematical Formulation

Notation

- x → Real image from training set.

- z → Random noise input.

- G(z) → Generated image from noise.

- D(x) → Probability that x is real.

- D(G(z)) → Probability that G(z) is real.

Loss Function

The training problem is a minimax optimization:

V(D, G) = E\_{x~pdata(x)}[log(D(x))] + E\_{z~pz(z)}[log(1 - D(G(z)))]

- Discriminator maximizes:

- log(D(x)) → probability real images are classified as real.

- log(1 - D(G(z))) → probability generated images are classified as fake.

- Generator minimizes:

- log(1 - D(G(z))) → pushes the discriminator to believe fakes are real.

# 5. Training Process

1. Sample random noise z and generate an image using the generator.

2. Feed generated image into discriminator, get probability D(G(z)).

3. Feed real image x into discriminator, get probability D(x).

4. Update:

- Discriminator: Learns to increase accuracy in classifying real vs fake.

- Generator: Learns to produce more realistic images to fool the discriminator.

Repeat until the generator’s outputs are indistinguishable from real images.

At the end:

- Discriminator is discarded.

- Generator is retained and can generate new data samples.

# 6. Types of GANs

Over time, researchers have proposed several GAN variants to address limitations and expand capabilities:

- DCGAN (Deep Convolutional GAN): Uses convolutional layers, effective for image generation.

- Conditional GAN (cGAN): Generates data conditioned on labels (e.g., generate a horse vs. zebra).

- CycleGAN: Performs image-to-image translation without paired data (e.g., horse ↔ zebra).

- Pix2Pix: Image-to-image translation with paired data (maps input to target output).

- StyleGAN: Capable of generating photorealistic human faces with control over style attributes.

- BigGAN: Large-scale GAN trained on ImageNet for diverse, high-quality image generation.

- Progressive GAN: Grows the generator and discriminator progressively for stable high-resolution image synthesis.

- Wasserstein GAN (WGAN): Improves training stability by using Wasserstein distance as a loss metric.

# 7. Applications of GANs in Frame Generation

GANs are powerful for generating temporally consistent video frames. Key applications include:

- Frame Interpolation: Creating intermediate frames for smoother playback.

- Future Frame Prediction: Predicting upcoming frames from past ones.

- Slow Motion Generation: Inserting new frames to slow down video without artifacts.

- Video Super-Resolution: Enhancing video quality while preserving temporal dynamics.

# 8. Pretrained Models for Frame Interpolation

- DAIN (Depth-Aware Video Frame Interpolation): High-quality results using depth + optical flow but resource intensive.

- RIFE (Real-Time Intermediate Flow Estimation): Very fast but requires CUDA or Vulkan GPU.

- Super-SloMo: Simpler flow-based model, slower compared to RIFE.

Challenges:

- Requires GPUs (≥4 GB VRAM).

- Complex dependencies (CUDA, Vulkan, custom ops).

- CPU-only mode is very slow.

# 9. Alternatives for Limited Hardware

- Optical Flow (OpenCV): CPU-based but less realistic.

- FFmpeg Minterpolate: Fast CPU interpolation with motion compensation.

Command Example:

ffmpeg -i input.mp4 -vf "minterpolate=fps=60" output.mp4

- Hybrid Approaches: Combine optical flow + image enhancement (e.g., ESRGAN).

# 10. Pros and Cons of GANs

Advantages

- Generates highly realistic images/videos.

- Requires no labels (unsupervised).

- Captures spatial + temporal dynamics.

Disadvantages

- Training instability.

- Mode collapse (same output repeatedly).

- High computational requirements.

# 11. Summary

- GANs introduce a generator–discriminator adversarial framework for realistic data synthesis.

- Extended to video, they enable frame interpolation, super-resolution, and prediction.

- Pretrained models (DAIN, RIFE, Super-SloMo) achieve strong results but need powerful GPUs.

- Practical alternatives (OpenCV, FFmpeg) enable CPU-friendly frame generation.

- Despite challenges, GANs remain a cornerstone in generative AI research and real-world applications.