Generative Adversarial Nets (GANs)

João Vítor Correia Pessoa
Pedro Pablo Riascos Henao
Breno de Lima Galves

Universidade Federal do Rio de Janeiro (UFRJ)

December 1, 2025

Summary

Motivation

- Most practical ML solutions in vision and NLP were **discriminative**
- Discriminative models: efficient, robust, backbone of ML
- Examples:
  - Convolutional Neural Networks (CNNs)
  - Logistic Regression
  - Support Vector Machines (SVMs)
  - Decision Trees and Random Forests

Limitations of Discriminative Models

UFRJ
UNIVERSIDADE FEDERAL
DO RIO DE JANEIRO

Key Limitation

Discriminative models answer **"What is it?"** but not **"How was it made?"**

- No mechanism for sampling new data
- Only map $X \rightarrow Y$ (classification/regression)
- Cannot simulate complex real-world phenomena

Generative Perspective

- Generative models aim to learn:

$$P(X) \quad \text{or} \quad P(X, Y)$$

- Capture dependencies and correlations in data
- Example: realistic face generation
  - Discriminative: can classify misaligned eyes
  - Generative: would not generate unrealistic faces

Core Idea

Generative Models

Generate complex random variables X from simple distributions (Uniform, Gaussian)

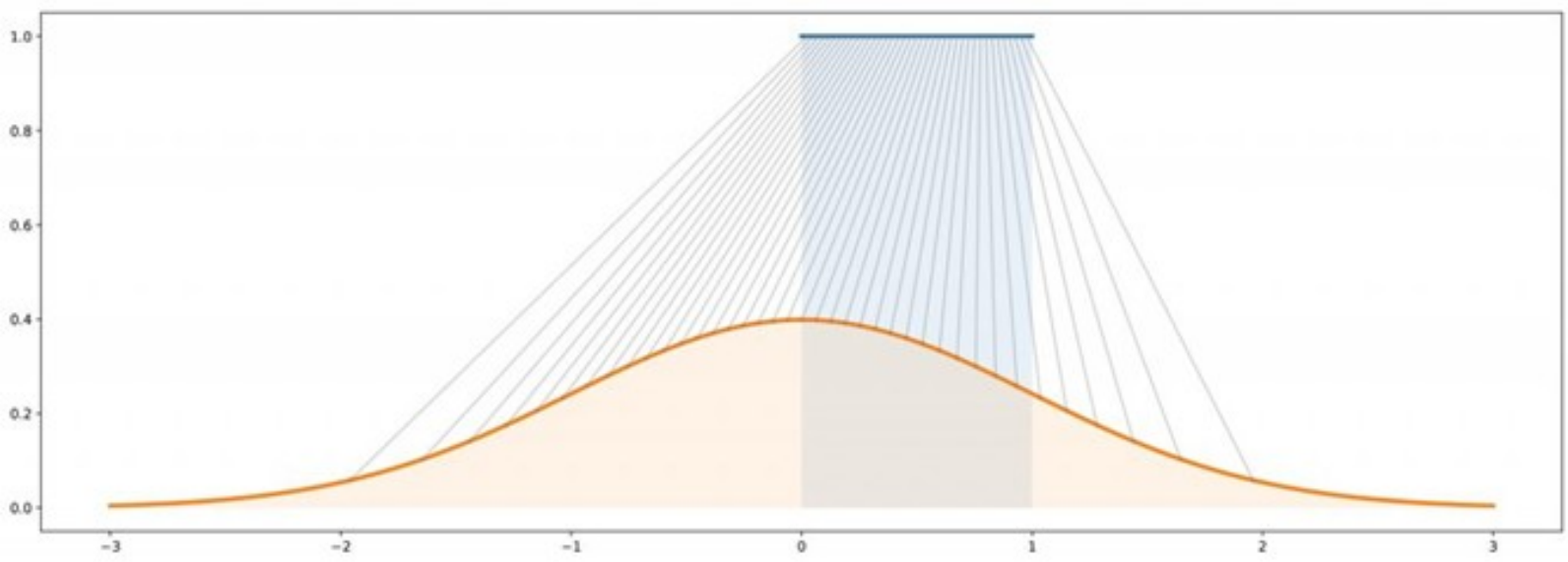
Challenge of Randomness

- Computers are deterministic $\rightarrow$ pseudo-random numbers
- Simplest case: $U \sim \mathcal{U}[0, 1]$
- Challenge: transform uniform distribution into complex distributions (e.g., pixels in realistic images)

Inverse Transform Method

1. **CDF Definition:**
$$CDF_X(x) = \mathbb{P}(X \leq x)$$
2. **Transformation:**
$$Y = CDF_X^{-1}(U), \quad U \sim \mathcal{U}[0,1]$$
3. **Result:**
$$CDF_Y(y) = \mathbb{P}(Y \leq y) = CDF_X(y)$$

Inverse Transform Visualization



Inverse Transform process

- Observe the illustration of the inverse transform method above
- In blue: the uniform distribution in $[0, 1]$
- In orange: the standard Gaussian (Normal) distribution
- In gray: the mapping from the uniform to the Gaussian distribution (CDF^{-1})

Core Concept

Inverse Transform Method

Uses CDF^{-1} to reshape a simple uniform distribution into a complex target distribution

GANs as Learned Transformations

- Generator learns complex transformation:

$$G(z) \simeq F^{-1}(z)$$

- z: simple noise vector (Gaussian/Uniform)
- G: differentiable neural network
- $G(z)$: realistic samples from target distribution
- Discriminator provides feedback via gradients

Probabilistic Graphical Models (PGMs)

- Examples: Bayesian Networks, Markov Random Fields, Boltzmann Machines
- Explicit modeling of joint distributions
- **Limitations:**
  - Complex conditional distributions
  - Poor scalability for high-dimensional data
  - Expensive and slow sampling

Restricted Boltzmann Machines (RBMs)

- Learn probability distribution via visible and hidden layers
- Training: Contrastive Divergence
- **Limitations:**
  - Approximate and unstable training
  - Difficulty with complex data
  - Poor scalability for deep architectures

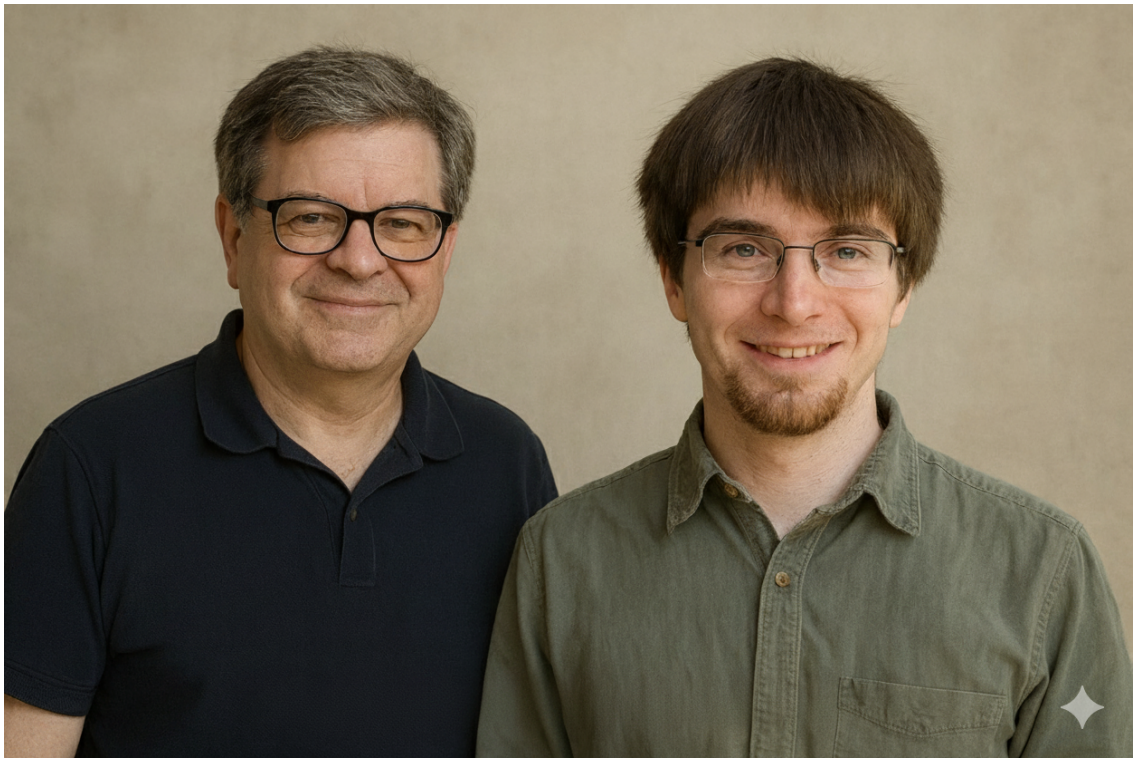
Deep Belief Networks (DBNs)

- Stack multiple RBMs $\rightarrow$ deep generative model
- **Limitations:**
  - Multi-stage training (layer by layer)
  - Slow sampling
  - Blurry and limited image quality

Variational Autoencoders (VAEs)

- Probabilistic encoder-decoder framework
- Stable training, strong mathematical foundation
- **Limitations:**
  - Loss regularization $\rightarrow$ blurry images
  - Difficulty capturing high-frequency details

Historical Context



Yann LeCun and Ian Goodfellow

- Proposed by **Ian Goodfellow et al.**, NIPS 2014
- Considered by Yann LeCun as: *"The most interesting idea in the last 10 years of machine learning."*

Formal Definition

Generative Adversarial Networks (GANs)

Deep learning models composed of two neural networks in competition:

- **Generator**: creates new data from random noise
- **Discriminator**: distinguishes real training data from fake samples

Both are trained simultaneously in a **zero-sum game**:

- Generator aims to fool the Discriminator
- Discriminator aims to resist being fooled

Advantages over Previous Models

1. Direct training on sampling, no likelihoods, integrals, or normalizations
2. More realistic samples, sharper, visually faithful outputs
3. No need for inverse CDF
4. Scales well to high-dimensional data (e.g., high-resolution images)

Theoretical Potential

- GANs can, in theory, learn to imitate **any data distribution**
- Applications across domains:
  - Images
  - Music
  - Speech
  - Text
- Capable of creating worlds strikingly similar to ours

Interpretation

Are GANs robotic artists or powerful mirror neurons? Either way, they represent a paradigm shift in generative modeling

Central question

How Do Two Networks Compete to Learn?

Adversarial Setup

Competitive Training

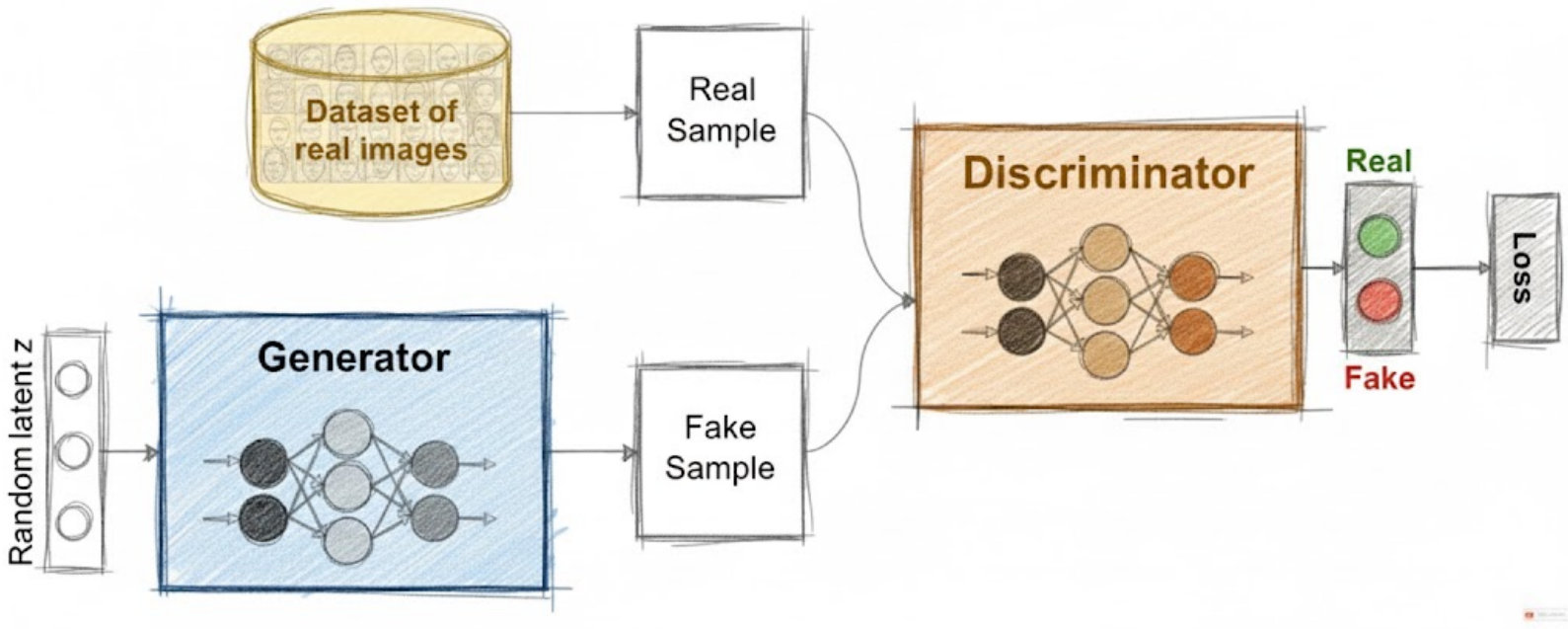
Two neural networks trained against each other:

- Generator vs. Discriminator
- The term *adversarial* comes from this competitive setup

Core Architecture

- **Generator (G)**: noise $\rightarrow$ synthetic data
- **Discriminator (D)**: real + fake data $\rightarrow$ probability

Architecture Diagram



GAN Architecture Diagram

Training Process

1. Generator creates image from random noise
2. Discriminator receives real + fake images
3. Outputs authenticity probability
4. Loss computed $\rightarrow$ backpropagation updates both networks

Key Mechanism

Balance is Crucial

- Both networks are differentiable
- Gradient flows through D back to G
- Balance matters:
  - Too strong D $\rightarrow$ G cannot learn
  - Too weak D $\rightarrow$ training loses meaning

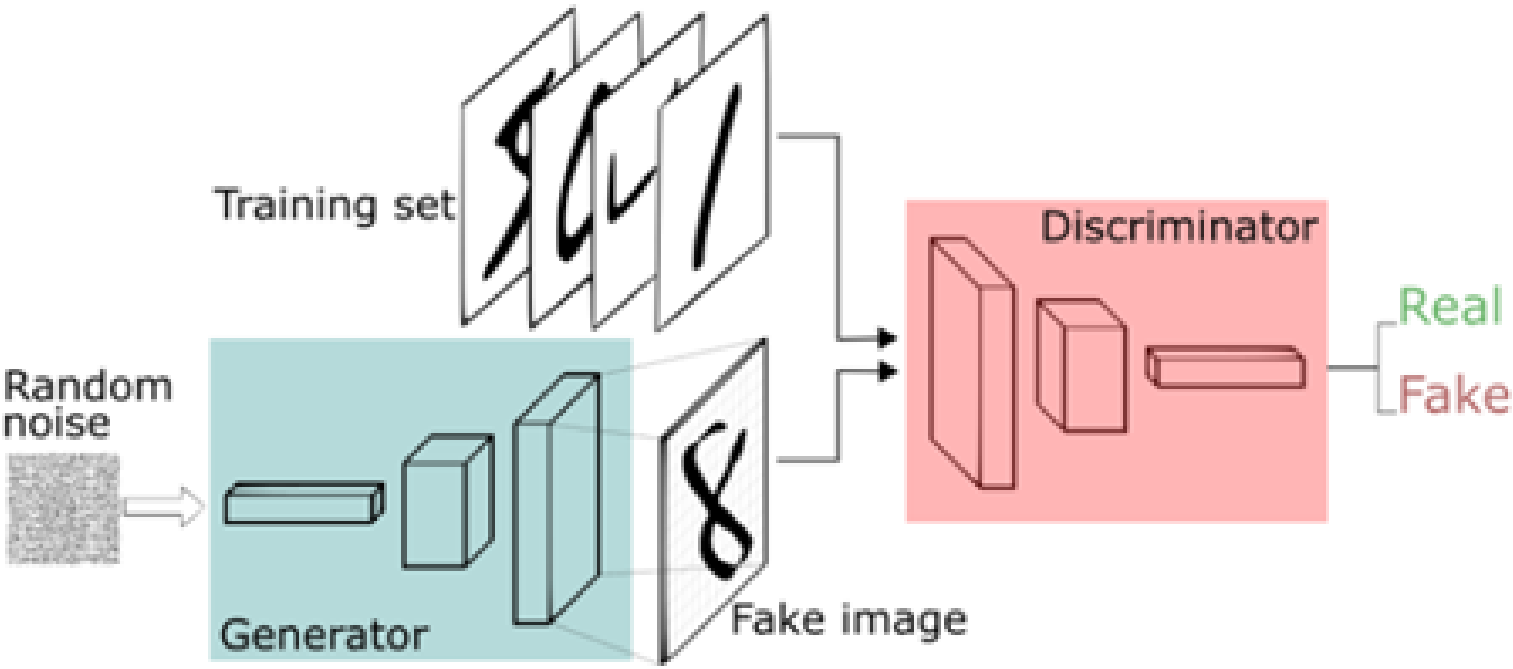
Evolution

- With stable training:
- Generator improves continuously
- Learns to produce highly realistic data
- Becomes a “forger” capable of mimicking the dataset

MNIST Example

- Discriminator: MLP classifier (real vs. fake digits)
- Generator: inverse MLP (upsampling noise $\rightarrow$ image)
- Downsampling vs. Upsampling

MNIST Diagram



MNIST Example

Opposing Loss Functions

Adversarial Objectives

- Discriminator: maximize accuracy
- Generator: minimize discriminator's success
- Loss functions push against each other
- Result: generated images converge toward realism

Beyond Basics

- How far can synthetic data go?
- Mathematical/statistical tools behind stability
- Leads to advanced research in optimization, probability, and programming frameworks

GANs as a Minimax Game: Model Definitions

- GANs are formulated as a **two-player minimax game**
- Framework is simpler when both G and D are **Multilayer Perceptrons (MLPs)**

1. **Generator (G)**:
   - Differentiable function $G(z; \theta_g)$, maps noise $z \sim p_z(z)$ to data space x.
   - Implicitly defines probability distribution p_g
2. **Discriminator (D)**:
   - Function $D(x; \theta_d)$, outputs scalar $D(x)$
   - $D(x)$ is the probability x comes from p_{data} rather than p_g

The Minimax Objective

The Minimax Objective Function $V(G, D)$

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

- **Discriminator (D)**: **maximizes** $V(D, G)$, aiming for correct classification (real vs. fake)
- **Generator (G)**: **minimizes** $V(D, G)$, equivalent to minimizing $\log(1 - D(G(z)))$

Training Implementation

- Entire system trained via backpropagation using **Iterative SGD** on minibatches
- Procedure alternates: k **optimization steps for** D and **1 step for** G

Discriminator Optimization Steps (∇_{θ_d}): (Gradient Ascent for maximizing V)

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D(x^{(i)}) + \log \left(1 - D(G(z^{(i)})) \right) \right]$$

Generator Optimization Step (∇_{θ_g}): (Gradient Descent for minimizing V)

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D(G(z^{(i)})) \right)$$

Practical Modification for G

- **Issue**: Original objective ($\min \log(1 - D(G(z)))$) has **weak gradients** when G is poor and D is confident
- This causes the objective to **saturate** early in training

Practical Fix

G is instead trained to **maximize** $\log D(G(z))$

- Result: Provides **significantly stronger gradients** at the start
- Theoretical Result: This alternative objective leads to the **same fixed point** (equilibrium) as the original minimax game

Proposition 1: The Optimal Discriminator (1/2)

Theoretical Context

The analysis is performed in a non-parametric setting (infinite capacity for G and D)

Proposition 1: For a fixed Generator G, the Optimal Discriminator D_G^* is the one that maximizes the value function $V(G, D)$ **Optimization:** The

value function $V(G, D)$ is rewritten as:

$$V(G, D) = \int_x [p_{\text{data}}(x) \log(D(x)) + p_g(x) \log(1 - D(x))] \, dx$$

- D's training is equivalent to maximizing the log-likelihood of correctly estimating the conditional probability $P(Y|x)$, where Y indicates whether sample x is real ($Y = 1$) or generated ($Y = 0$)

Proposition 1: The Optimal Discriminator (2/2)

Optimization Result: For a fixed G, the Optimal Discriminator $D_G^*(x)$ that maximizes $V(G, D)$ is given by:

$$D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}$$

Interpretation of $D_G^*(x)$:

- $D_G^*(x)$ represents the probability that an observed sample x came from the real data distribution (p_{data}), rather than the generated distribution (p_g)
- When $p_{\text{data}}(x) \gg p_g(x)$, the sample is very likely to be real, and $D_G^*(x) \approx 1$
- When $p_{\text{data}}(x) \ll p_g(x)$, the sample is very likely to be fake, and $D_G^*(x) \approx 0$
- When $p_{\text{data}}(x) = p_g(x)$, the sample has an equal chance of being real or fake, and $D_G^*(x) = 1/2$

Theorem 1: The Global Optimum

Theorem 1: By substituting $D_G^*(x)$ into $V(G, D)$, we obtain the **Virtual Training Criterion** $C(G)$:

$$C(G) = \max_D V(G, D) = E_{x \sim p_{\text{data}}}[\log D_G^*(x)] + E_{x \sim p_g}[\log(1 - D_G^*(x))]$$

The global minimum of $C(G)$ is achieved **if and only if** $\mathrm{p_g} = \mathbf{p_{data}}$.

- If $p_g = p_{\text{data}}$, then $D_G^*(x) = 1/2$
- At this minimum, $C(G)$ reaches the value: $\mathrm{C}^* = -\log 4$

Proof via Jensen–Shannon Divergence (JSD)

- To prove the minimum, $C(G)$ is re-expressed using the **Kullback–Leibler (KL) Divergence** and the **Jensen–Shannon Divergence (JSD)**
- The relationship established is:

$$C(G) = -\log(4) + 2 \cdot JSD(p_{\text{data}} \parallel p_g)$$

Conclusion

- Since $JSD \geq 0$ (JSD is always non-negative)
- The minimum value $C(G) = -\log 4$ is achieved **only when** $JSD = 0$
- $JSD = 0$ requires $\mathrm{p_g} = \mathrm{p_{data}}$, proving the global optimality

Proposition 2: Convergence of Algorithm 1

- If G and D have sufficient capacity, and D is allowed to reach its optimum (D_G^*) at each step
- Then p_g is guaranteed to converge to p_{data} when G is updated to improve $C(G)$

The Mathematical Basis is Convexity:

- The criterion $C(G) = \sup_D V(G, D)$ is **convex** in p_g
- This ensures a **unique global optimum** (as proven by Theorem 1)
- The generator update corresponds to a step of gradient descent for p_g, guaranteeing convergence to the optimum with sufficiently small updates

Analogy for Global Optimum

Tug-of-War Analogy

- Training GAN = **Tug-of-War** between G and D
- **Equilibrium** is reached when the rope is centered: $p_g = p_{\text{data}}$
- At equilibrium, the Discriminator (judge) cannot distinguish the distributions: $D(x) = 1/2$ everywhere
- The JSD formalizes this "distance", which is minimized to zero at the optimum

Experiments Methodology

- **Objective:** Quantitative and qualitative assessment of generated samples.
- **Training Datasets:**
  - MNIST (Handwritten digits)
  - Toronto Face Database (TFD)
  - CIFAR-10 (Tiny images)
- **Base Architecture:** Generator (G) and Discriminator (D) models defined using **MLPs** (Multi-Layer Perceptrons)

Specific Network Architecture

- **Generator Network (G):**
  - Used a mix of **ReLU** and **Sigmoid** activations
  - **Noise** (z) is input only to the **lowest layer** of G
- **Sampling Process:**
  - Generated samples are **fair random draws** and **uncorrelated**
  - **Does not rely** on Markov Chain mixing
- **Discriminator Network (D):**
  - Used **Maxout** activations
  - **Dropout** was applied during the training of D (regularization)

Quantitative Evaluation: Likelihood Estimation

The Challenge of Explicit Likelihood

Due to the **implicit** nature of the generative model (p_g), the exact likelihood ($p(x)$) is **intractable**

Technique Used: Gaussian Parzen Window

- **Procedure:** Fit a Gaussian Parzen Window to the samples generated by G
- **Metric:** Report the **log-likelihood** under this Parzen distribution
- **Parameter σ:** Obtained through **cross-validation** on the validation set

Quantitative Evaluation: Results and Limitations

- **Limitations (Parzen):** The method has **reasonably high variance** and does not perform well in high-dimensional spaces.
- **Results:** GANs proved to be **competitive** when compared to:
  - DBN (Deep Belief Networks)
  - Stacked CAE (Contractive Autoencoders)
  - Deep GSN (Generative Stochastic Networks)

Qualitative Evaluation and Samples

- **Visual Quality:** Generated visual samples are considered **competitive** with the best generative models in the literature
- **Non-Memorization:**
  - Visualizations confirm the model **did not memorize** the training set
  - Demonstrated by comparing generated samples with their nearest training example

Advantages

- **Computational Simplicity:**
  - Requires **Backpropagation Only** for training.
  - **No Inference** or **Markov Chains** are needed for sampling (simpler than RBMs/DBMs)
  - High **Design Flexibility** for incorporating differentiable functions
- **Representational Strength:**
  - Ability to represent **sharp** (even degenerate) distributions, unlike Markov Chain methods
- **Statistical Benefit:**
  - G is updated indirectly via D gradients, which helps prevent input components from being **directly copied**, reducing *overfitting*

Disadvantages

- **Lack of Explicit $p_g(x)$:**
  - No explicit probability distribution; $p(x)$ must be **approximated** (e.g., via Parzen estimation)
- **Training Instability:**
  - Requires **Synchronization** between D and G to maintain balance
- **Risk of Mode Collapse:**
  - The **"Helvetica scenario"** occurs when G is over-trained, mapping many z values to the same x
  - **Result:** Generated samples show **low diversity** (failure to model the full data distribution)

Key GAN Variants

Due to the sheer number of models, we will focus on several notable variants:

- **DCGAN** (Deep Convolutional GAN)
- **WGAN** (Wasserstein GAN)
- **cGAN** (Conditional GAN)
- **Pix2Pix** (Image-to-Image Translation)
- **CycleGAN** (Unpaired Image-to-Image Translation)
- **Style-GAN** (Style-based Generator Architecture)

DCGAN

- Proposed by *Radford et al.* (2015)
- Designed to overcome instability and limitations of original GANs
- Introduced systematic use of CNNs for image generation
- Result: higher visual quality and improved training stability

Key Architectural Features

Generator:

- ConvTranspose2D layers for upsampling
- No fully connected layers after initial projection
- **BatchNorm** in most layers (except output)
- ReLU activations internally, tanh at output

Discriminator:

- Conv2D with stride for downsampling (no pooling)
- **LeakyReLU** activations
- **BatchNorm** (except input layer)
- Final sigmoid for real vs. fake probability

Loss Function

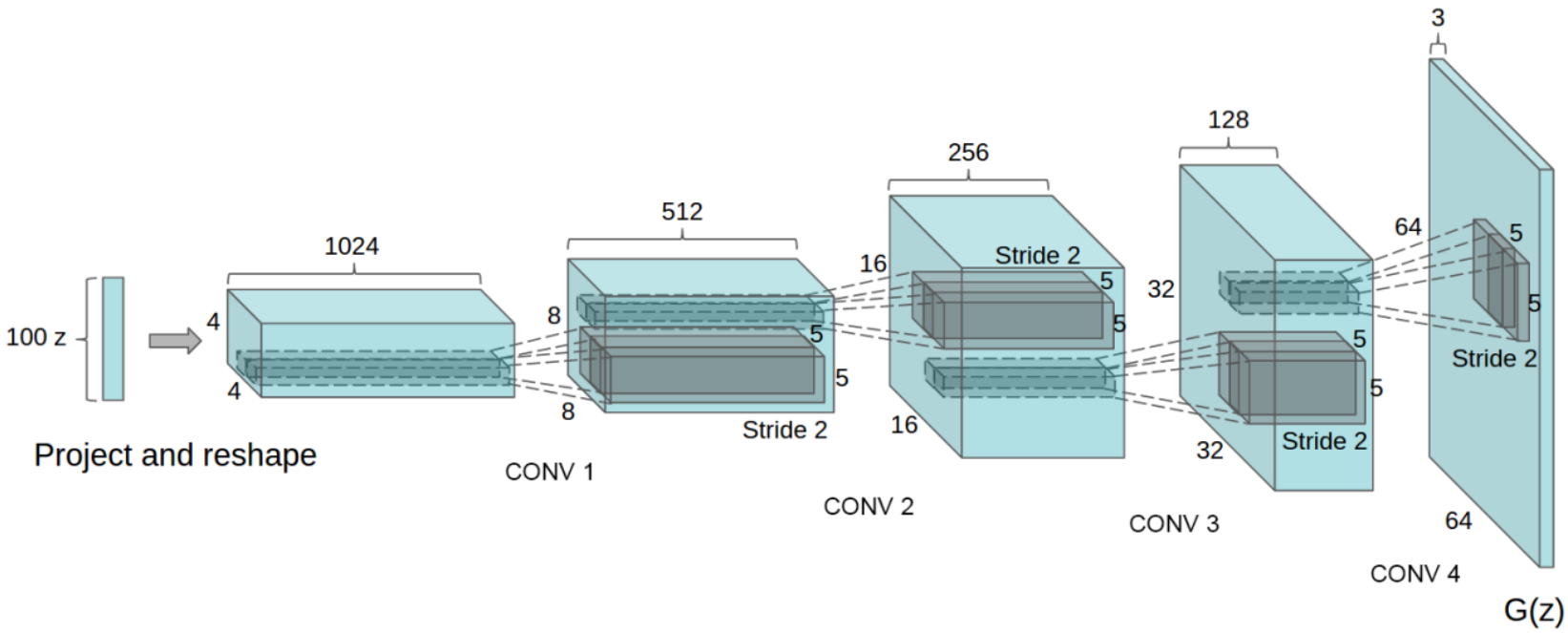
- Standard Binary Cross-Entropy (BCE) objective:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] - \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]$$

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p_z}[\log(D(G(z)))]$$

- Implemented as BCE with logits (e.g., `BCELoss`)

Generator Architecture



DCGAN Generator $G(z)$: latent vector $\rightarrow 64 \times 64 \times 3$ image

- Latent vector z (100-dim) progressively upsampled
- Feature map size doubles, depth halves at each stage
- Example: $4 \times 4 \rightarrow 8 \times 8 \rightarrow \cdots \rightarrow 64 \times 64$

Why DCGAN Outperforms Original GAN

Original GAN:

- Relied on MLPs (ignored spatial structure)
- Training highly unstable
- Generated blurry, inconsistent images
- More prone to *mode collapse*

DCGAN:

- CNNs capture local patterns (edges, textures)
- Convolutions exploit spatial structure
- Produces sharper, more realistic images

Semantic Latent Space Manipulation

Bedroom dataset: removing "window" filters in latent space

- Top: unmodified samples (bedrooms with windows)
- Bottom: windows removed or replaced by doors/mirrors
- Shows separation of scene vs. object representation

Training Stability and Rich Latent Space

Improved Stability:

- BatchNorm, ReLU/LeakyReLU, Conv/ConvTranspose reduce vanishing gradients
- Training becomes smoother and more reliable

Richer Latent Representations:

- Continuous and semantic latent space
- Enables smooth interpolation between samples
- Controllable attribute manipulation
- Style control — foundation for later models (e.g., StyleGAN)

WGAN

- WGAN was introduced by Arjovsky et al. (2017).
- It replaces the Jensen-Shannon divergence with the Wasserstein distance (Earth Mover's Distance - EMD).
- Measures the minimum effort required to transform P_g into P_r.

Wasserstein Distance

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

- $\Pi(P_r, P_g)$: set of joint distributions with marginals P_r and P_g.
- Interpretation: minimum transport cost between distributions.

Improvements of WGAN

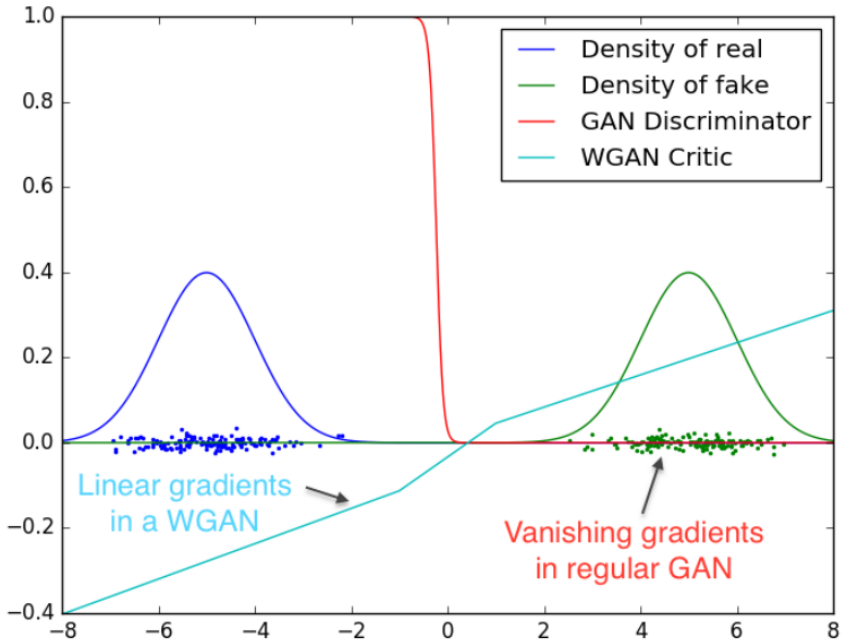
- **Stability**: Provides meaningful gradients even without initial overlap.
- **Vanishing Gradients**: Prevents loss saturation.
- **Mode Collapse**: Smooth penalization encourages sample diversity.

The Critic

- Replaces the traditional discriminator.
- Returns a score that approximates the Wasserstein distance.

$$W(P_r, P_g) = \sup_{C \in \mathcal{L}_1} \mathbb{E}_{x \sim P_r}[C(x)] - \mathbb{E}_{z \sim P(z)}[C(G(z))]$$

Discriminator vs Critic



Optimal discriminator and critic when learning to differentiate two Gaussians. The minimax GAN discriminator saturates and results in vanishing gradients, while the WGAN critic provides clean gradients across the space

Lipschitz Constraint

- The Critic must be 1-Lipschitz:

$$||C(x_1) - C(x_2)|| \leq ||x_1 - x_2||$$

- In the original WGAN: **weight clipping**
- Problem: limits modeling capacity

Loss Functions

Critic:

$$L_C = \mathbb{E}_{z \sim P(z)}[C(G(z))] - \mathbb{E}_{x \sim P_r}[C(x)]$$

Generator:

$$L_G = -\mathbb{E}_{z \sim P(z)}[C(G(z))]$$

WGAN-GP (Gradient Penalty)

- Gulrajani et al. (2017)
- Replaces weight clipping with gradient penalty
- Enforces 1-Lipschitz constraint more robustly

Critic Loss in WGAN-GP

$$L_{C_{GP}} = \underbrace{\mathbb{E}_{z\sim P(z)}[C(G(z))] - \mathbb{E}_{x\sim P_r}[C(x)]}_{\text{Wasserstein Loss}} + \underbrace{\lambda \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}[(||\nabla_{\hat{x}} C(\hat{x})||_2 - 1)^2]}_{\text{Gradient Penalty}}$$

- λ: hyperparameter (typically 10)
- $\hat{x}$: interpolation between real x and generated $G(z)$
- Penalty enforces gradient norm ≈ 1

Recap

- WGAN improves training stability and sample quality
- WGAN-GP is the preferred method, avoiding issues of weight clipping
- Produces more realistic samples and robust training

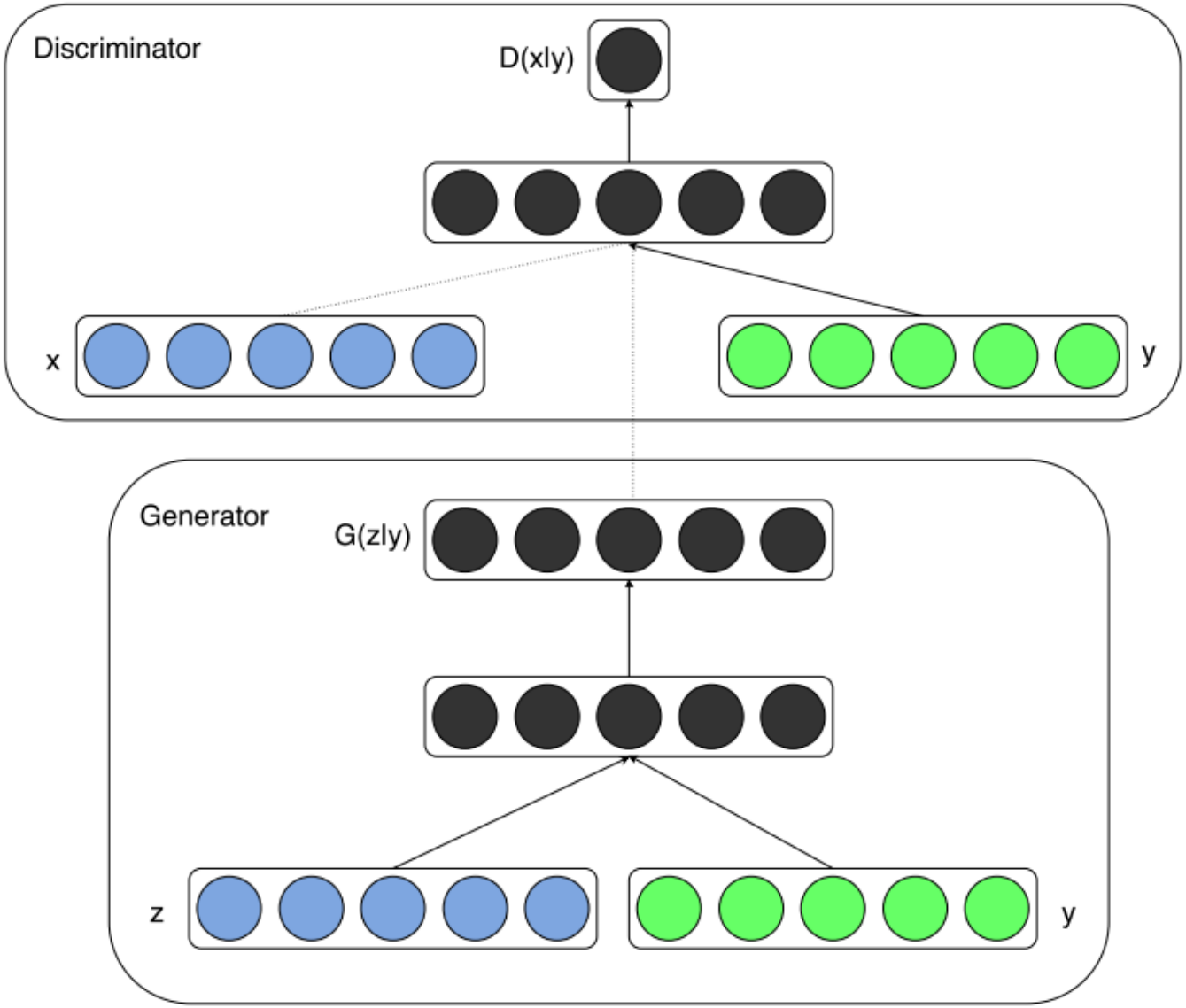
cGAN

- Conditional GAN (cGAN) introduced by Mirza and Osindero (2014).
- Extends the original GAN by adding conditional information (y).
- Enables controlled generation of samples based on input conditions.

Architecture

- Condition y can be a class label, semantic map, or other modality.
- Both Generator (G) and Discriminator (D) receive y as input.
- Provides control over generated outputs.

Conditional adversarial net



Generator

- Input: noise vector z and condition y.
- Learns mapping:

$$G : z, y \rightarrow x'$$

- Typically, z and y are concatenated before being fed into the network.

Discriminator

- Input: sample (x or x') and condition y.
- Learns to classify whether x is real or fake given y.

$$D : (x \text{ or } x'), y \rightarrow \text{probability (real vs. fake)}$$

- y is concatenated with x at input or intermediate layers.

Loss Function

- Modified from original GAN loss to include condition y.
- Objective:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim P_z(z)}[\log(1-D(G(z|y)))]$$

Loss Details

Discriminator Loss (L_D):

$$\max_D V(D,G) = \mathbb{E}_{x\sim P_{data}(x)}[\log D(x,y)] + \mathbb{E}_{z\sim P_z(z)}[\log(1 - D(G(z,y),y))]$$

Generator Loss (L_G):

$$\min_G V(D,G) = \mathbb{E}_{z\sim P_z(z)}[\log(1 - D(G(z,y),y))]$$

Key Applications

- **Image Generation**
  - Condition (y): Class Label ("Cat", "Car")
  - Output (x'): Image corresponding to the class
- **Image-to-Image Translation (Pix2Pix)**
  - Condition (y): Semantic Map (street contours)
  - Output (x'): Realistic street photo
- **Text Generation**
  - Condition (y): Sentence prefix
  - Output (x'): Continuation of the sentence

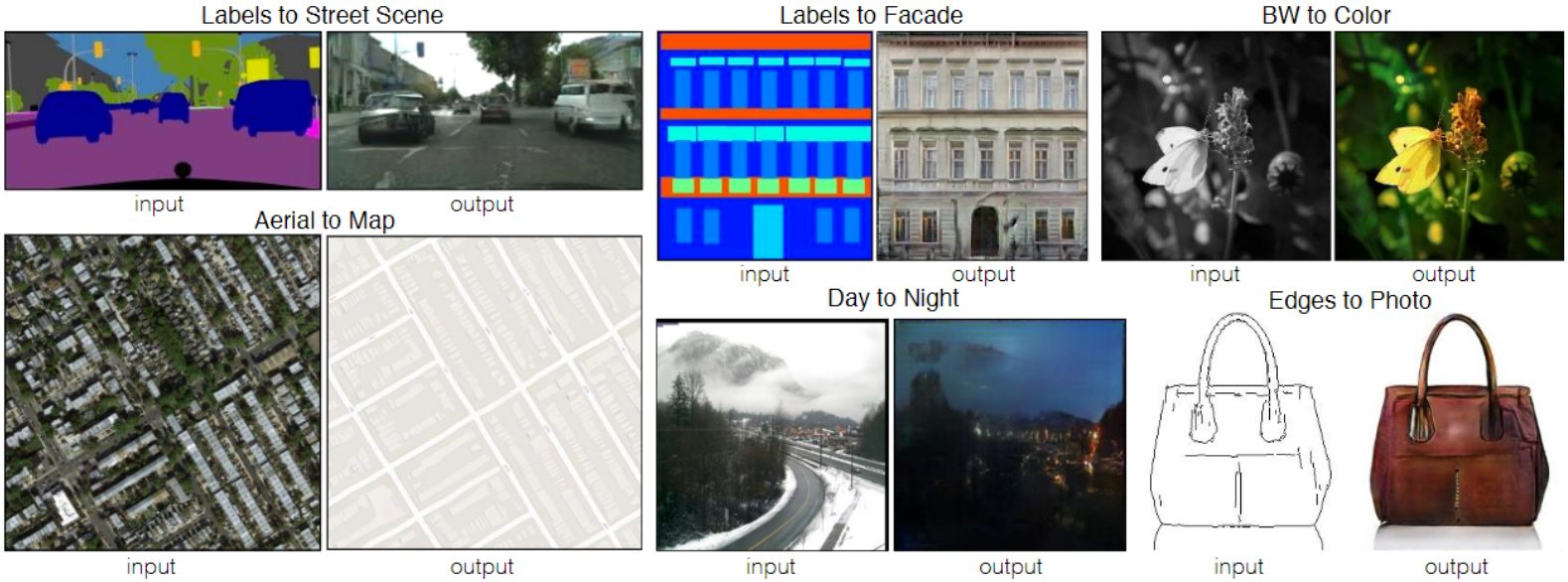
Recap

- cGAN extends GANs with conditional control.
- Enables targeted and meaningful sample generation.
- Foundation for many applications in vision and language.

Pix2Pix (Image-to-Image Translation)

- Pix2Pix proposed by Isola et al. (2017)
- Built directly on the Conditional GAN (cGAN) architecture
- Learns a mapping from input image (x) to output image (y) based on a condition

Image-to-Image Translation Overview



General-purpose conditional adversarial networks for image-to-image translation. Same architecture and objective, trained on different paired datasets

Fundamental Requirement: Paired Data

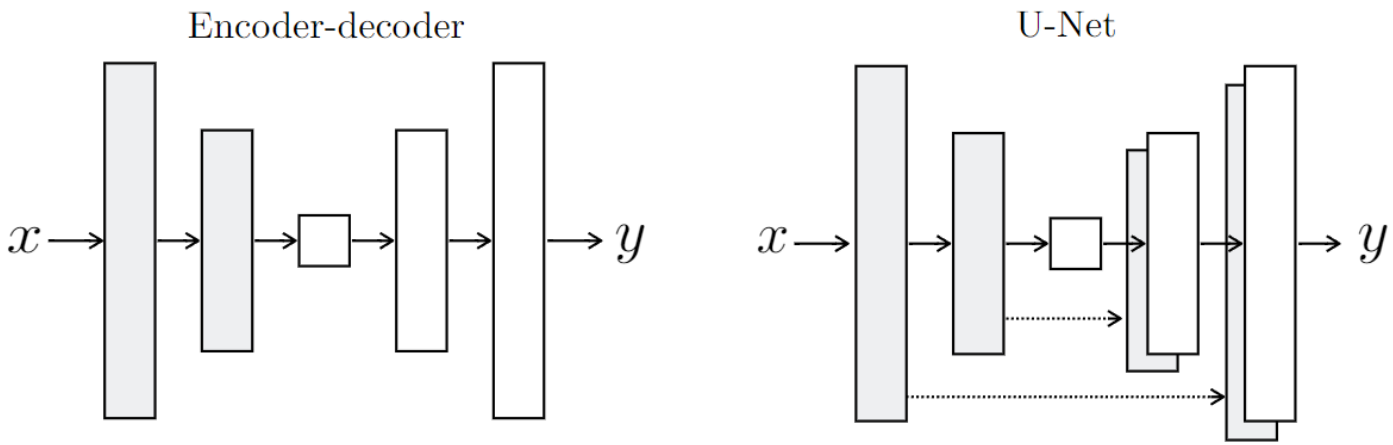
- Pix2Pix requires paired training data.
- Each input image (Domain A) must have a corresponding aligned output image (Domain B).
- **Contour/Sketch Map → Real Photo**
  - Application: Image generation from drawing
- **Semantic Map → Street Photo**
  - Application: Urban landscape synthesis
- **Infrared Image → RGB Image**
  - Application: Modality conversion

Generator (U-Net)

- Uses U-Net architecture instead of a standard fully convolutional network
- Encoder-decoder with skip connections linking encoder layers to corresponding decoder layers
- **Advantage:** Skip connections transfer low-frequency information (contours, global structure) directly to the output, preserving input structure and reducing blurring

Generator Architectures

- Two choices for the generator architecture
- **U-Net**: encoder-decoder with skip connections
- Skip connections link mirrored layers in encoder and decoder stacks
- Advantage: preserves spatial information and reduces blurring in generated images



U-Net generator: encoder-decoder with skip connections between mirrored layers

Discriminator (PatchGAN)

- Uses PatchGAN architecture.
- Classifies $N \times N$ patches of the output image as real or fake
- **Advantage:** Forces the Generator to focus on high-frequency details (texture, style), producing sharper and more realistic outputs

Total Loss Function

$$L_{Total}(G, D) = L_{cGAN}(G, D) + \lambda L_{L1}(G)$$

- Combination of adversarial loss and reconstruction loss
- λ (commonly $\lambda = 100$) controls the weight of the L1 loss

Loss Details

Adversarial Loss (L_{cGAN}):

$$L_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))]$$

Reconstruction Loss (L_{L1}):

$$L_{L1}(G) = \mathbb{E}_{x,y}[||y - G(x)||_1]$$

- Ensures generated image is not only realistic but also matches the target image

Effect of Different Losses

- Different loss functions induce different quality of results
- Each column shows outputs trained under a different loss
- Highlights how adversarial loss, L1/L2 reconstruction loss, or combinations affect realism and fidelity

Comparison of results under different loss functions. Each column corresponds to a distinct training loss

Recap

- Pix2Pix enables supervised image-to-image translation using paired data
- U-Net generator preserves structure, PatchGAN discriminator enforces detail
- Combined adversarial and L1 losses yield realistic and faithful translations

CycleGAN

- Proposed by Zhu et al. (2017)
- Elegant solution for image-to-image translation when paired data is scarce or unavailable
- Learns mappings between two domains X and Y without requiring 1:1 correspondence

Innovation: Unpaired Data

- CycleGAN works with unpaired datasets
- Learns translation between collections of images from Domain X and Domain Y
- **Domain X: Horses → Domain Y: Zebras** Application: Species conversion
- **Domain X: Summer Photos → Domain Y: Winter Photos** Application: Season transfer
- **Domain X: Photos (Style A) → Domain Y: Paintings (Style B)** Application: Artistic style transfer

Architecture

- Two complementary GANs form a cycle:
- **GAN $X \to Y$:**
  - Generator G: maps $X \to Y$
  - Discriminator D_Y: distinguishes real Y from generated $G(X)$
- **GAN $Y \to X$:**
  - Generator F: maps $Y \to X$
  - Discriminator D_X: distinguishes real X from generated $F(Y)$

Total Loss Function

$$L_{Total} = L_{GAN}(G, D_Y) + L_{GAN}(F, D_X) + \lambda L_{cyc}(G, F)$$

- **Adversarial Loss (L_{GAN}):** ensures generated images are indistinguishable from real ones in target domain
- **Cycle Consistency Loss (L_{cyc}):** enforces that translation and re-translation return the original image
- **Identity Loss (L_{id}, optional):** preserves color and composition if input already belongs to target domain

Cycle Consistency Details

Forward Cycle ($X \to Y \to X$):

$$L_{cyc\_forward} = \mathbb{E}_x[||F(G(x)) - x||_1]$$

Backward Cycle ($Y \to X \to Y$):

$$L_{cyc\_backward} = \mathbb{E}_y[||G(F(y)) - y||_1]$$

$$L_{cyc}(G, F) = L_{cyc\_forward} + L_{cyc\_backward}$$

Impact and Applications

- Enables image-to-image translation without paired datasets
- Applications: species conversion, season transfer, artistic style transfer
- Opened new possibilities in domains where paired data is prohibitive

Beyond CycleGAN

- **ProGAN** (Karras et al.): introduced progressive growing for higher resolution
- **StyleGAN**: evolution of ProGAN, with style-based generator architecture.
- Marked unprecedented leaps in photorealistic image generation quality and resolution

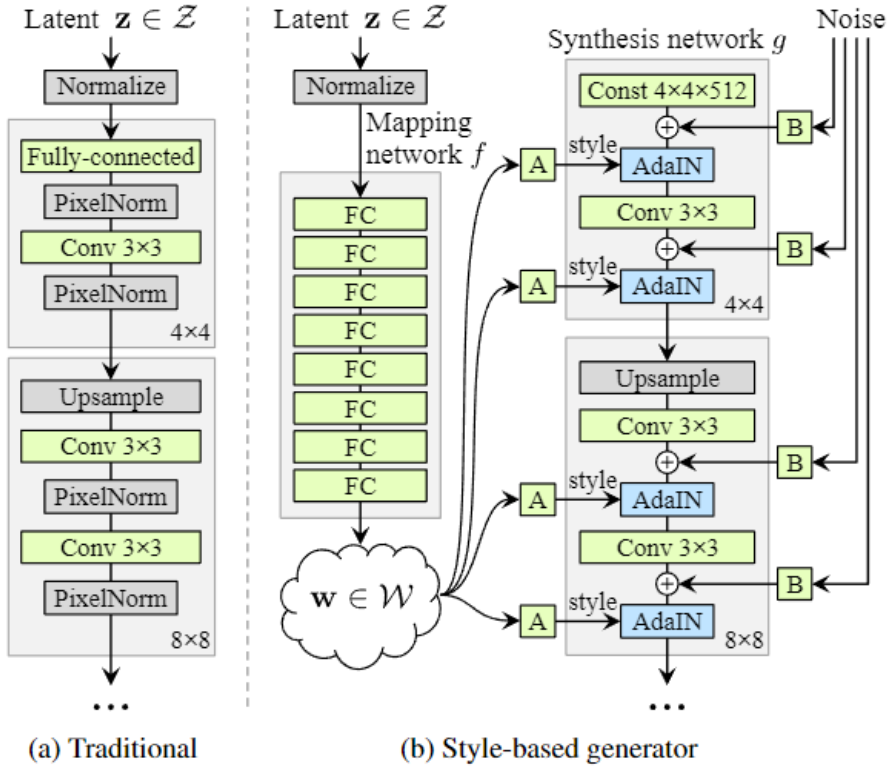
StyleGAN

- Proposed by Karras et al. (2018)
- Built on ProGAN but introduces a new generator architecture inspired by style transfer
- Goal: disentangle latent factors, enabling intuitive control over generated image attributes

Key Architectural Innovations

- **Mapping Network:**
  - Transforms latent vector z into intermediate vector w using an 8-layer MLP
  - w resides in latent space $\mathcal{W}$, designed to be less entangled than Z
  - Facilitates separation of variation factors (pose, color, identity)
- **Style Injection (AdaIN):**
  - Generator renamed to Synthesis Network, starts from a constant tensor
  - Styles (w) applied at each resolution block via Adaptive Instance Normalization (AdaIN)
  - Low-resolution styles: control global attributes (pose, face shape, hair type)
  - High-resolution styles: control fine details (hair color, freckles, wrinkles)
- **Noise Injection:**
  - Adds uncorrelated stochastic noise at each synthesis block
  - Controls fine stochastic variation (hair strands, textures) without affecting high-level attributes

StyleGAN Generator Summary



(a) Traditional (b) Style-based generator

StyleGAN generator: mapping network f, synthesis network g, AdaIN style injection, and noise addition

Truncation Trick in StyleGAN

- **Goal:** Improve image quality by sampling from a truncated latent space
- Compute center of mass of W:

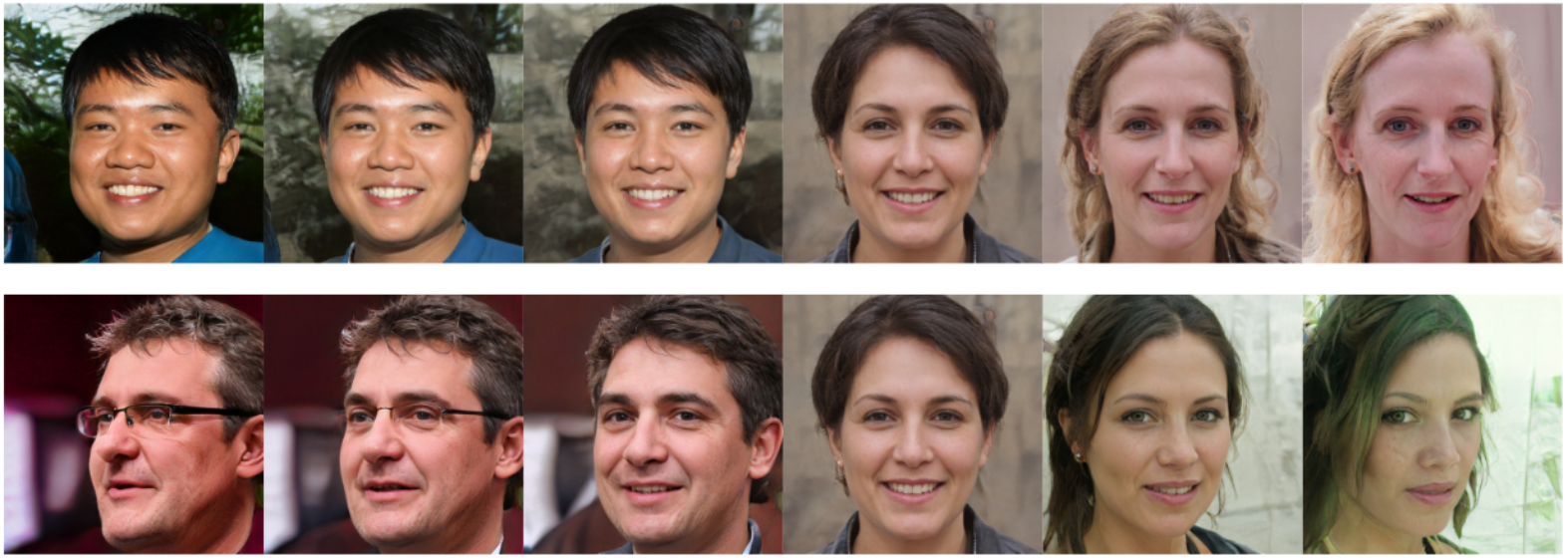
$$\bar{w} = \mathbb{E}_{z \sim P(z)}[f(z)]$$

  representing the "average face" in FFHQ
- Scale deviation from $\bar{w}$:

$$w' = \bar{w} + \psi(w - \bar{w}), \quad \psi < 1$$

- $\psi \to 0$: all faces converge to the mean face (no artifacts).
- Negative ψ: produces the opposite or "anti-face"
- High-level attributes often flip: viewpoint, glasses, age, coloring, hair length, gender
- Works reliably in W space without modifying the loss function

Truncation Trick Example



Effect of truncation trick as a function of style scale ψ: mean face at $\psi = 0$, anti-face at negative ψ.

StyleGAN2 (2020)

- Addressed visual artifacts in StyleGAN1 (droplet artifacts, static textures)
- **Key Improvements:**
  - Removed AdaIN mean/variance normalization; replaced with style modulation and demodulation
  - Introduced **Path Length Regularization**: enforces smooth, proportional mapping in latent space
  - Eliminated skip connections from ProGAN design for greater stability
- **Results:**
  - Sharper, more coherent images
  - Better disentanglement of latent space
  - Higher stability and quality across scales

Impact

- StyleGAN revolutionized controllable image generation
- Enabled semantic manipulation of attributes (pose, style, fine details)
- StyleGAN2 further improved realism and coherence, eliminating artifacts
- Established a foundation for subsequent models (StyleGAN3, diffusion-based approaches)

Final Considerations

- GANs established a powerful new paradigm for generative modeling, framed as a zero-sum minimax game between two networks
- This approach avoids complex and often intractable probability calculations required by earlier methods
- Subsequent research has focused primarily on control and application
- Advanced architectures such as **StyleGAN** introduced controllable and disentangled latent spaces, transforming GANs from simple data generators into models that enable semantic control over the generation process