Lecture 13: Generative Adversarial Networks - I

COMP 5801H/4900A: Generative AI and LLMs

2026-02-24

Sriram Subramanian
Assistant Professor & Canada Research Chair, Carleton University
Faculty Affiliate, Vector Institute for Artificial Intelligence
Faculty Affiliate, Schwartz Reisman Institute for Technology and Society

Carleton University

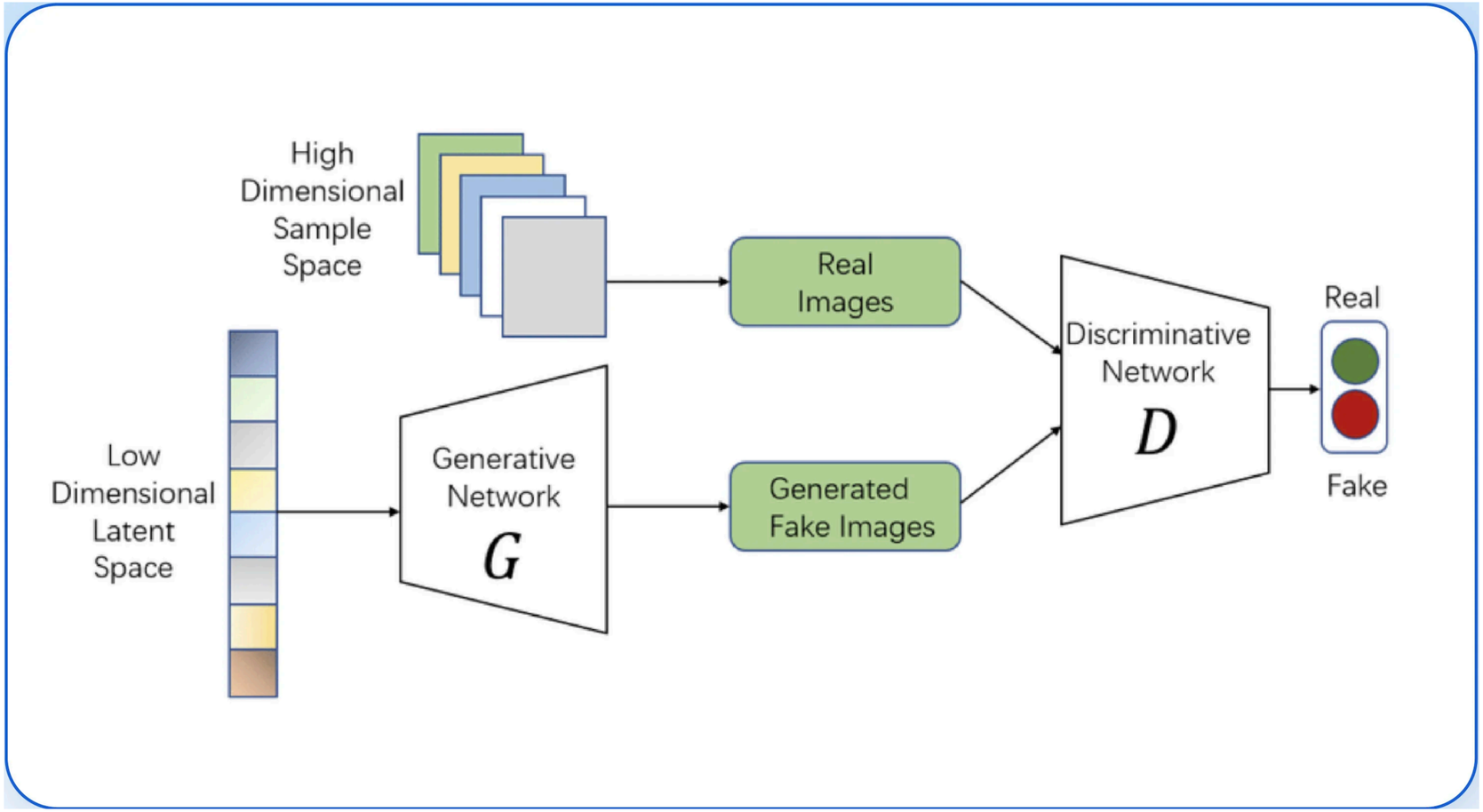
VECTOR INSTITUTE

SCHWARTZ REISMAN INSTITUTE FOR TECHNOLOGY AND SOCIETY

Outline

- Module 1: The Adversarial Concept
- Module 2: The Math of the Minimax Game
- Module 3: Training Dynamics & Challenges
- Module 4: Architectural Evolution
- Module 5: Comparison & Future

Generative Adversarial Networks (GANs)



Credit: https://www.xenonstack.com/blog/gans-for-image-synthesis

Generative Adversarial Networks (GANs)

- **The Paradigm Shift**: Moving from Likelihood Maximization (VAEs) to Game Theory (GANs).
- **Zero-Sum Game**: One network wins only if the other loses.
- **The Goal**: To reach "Nash Equilibrium"—the point where the Generator is so good the Discriminator can only guess with 50% accuracy.

The Core Philosophy: Sampling vs. Optimization

- **Explicit vs. Implicit Density**:
  - VAEs (Explicit): We define a mathematical distribution (the Gaussian) and force the data into it. We optimize a lower bound (ELBO).
  - GANs (Implicit): We don't care about the mathematical formula for the density. We just want to learn a procedure that produces samples that "look" like the data.
- **The Loss of "Distance"**:
  - In VAEs, if a pixel is off, the MSE/BCE loss increases.
  - In GANs, the "Loss" is just the opinion of another network. If the Discriminator says it looks real, it is successful, regardless of pixel-by-pixel accuracy.
- **The Sharpness Trade-off**:
  - VAEs tend to be "blurry" because they average over the Gaussian space.
  - GANs produce "sharp" results because they only need to be realistic enough to fool the Discriminator

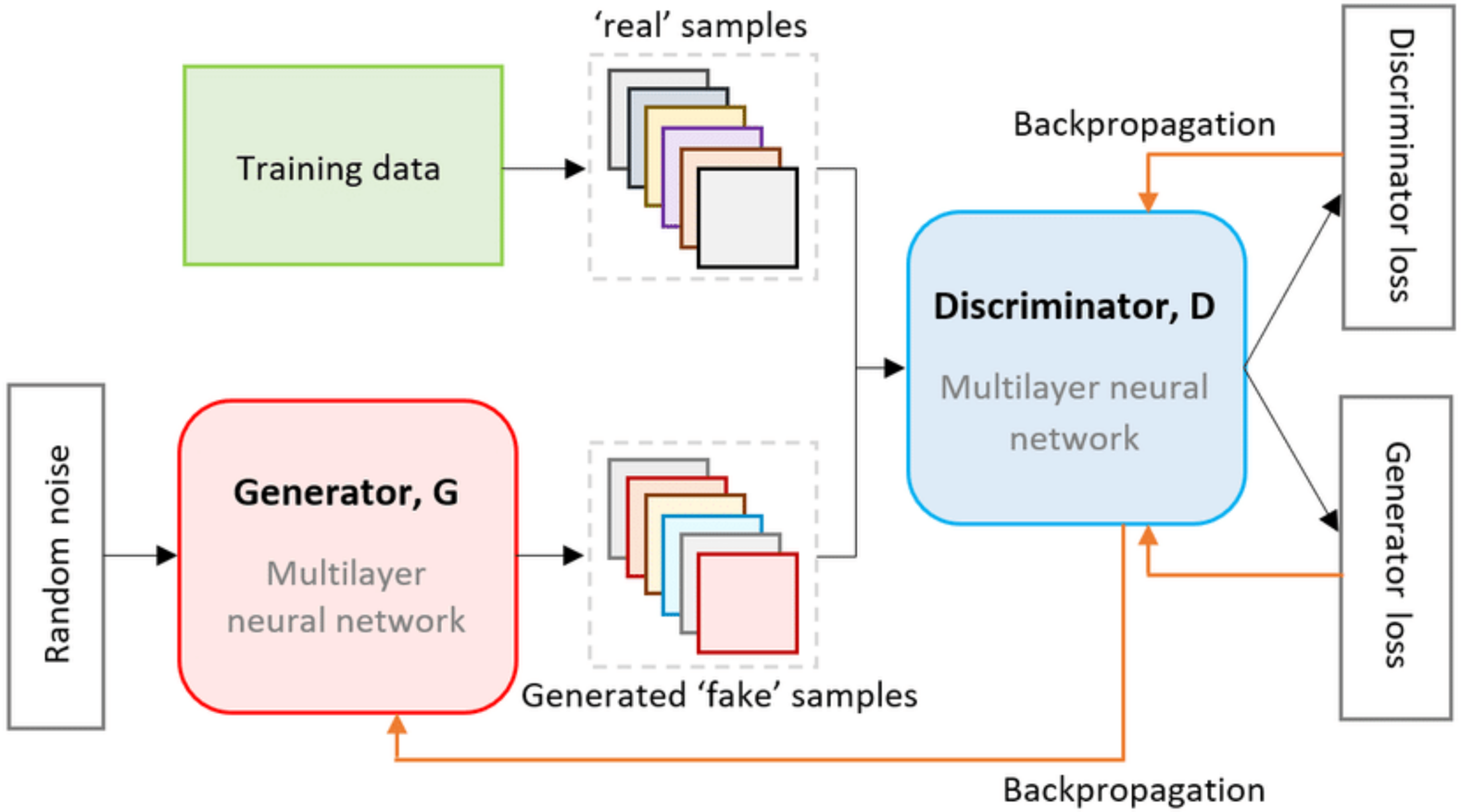
The Forger vs. The Detective

- **The Generator (G)**:
  - Role: The Counterfeiter.
  - Input: Random noise (z) from a simple distribution (e.g., Gaussian).
  - Goal: Map that noise into a complex data distribution (images) that is indistinguishable from the real thing.
  - It never sees the real images; it only hears the Detective's feedback.
- **The Discriminator (D)**:
  - Role: The Art Critic or Detective.
  - Input: Either a Real image (x) or a Fake image ($G(z)$).
  - Goal: Correcting classify "Real" as 1 and "Fake" as 0.
  - It is the teacher that the Generator must outsmart.

The "Co-evolution" Concept

- If the Detective is too lazy, the Forger stays bad.
- If the Detective is too perfect, the Forger gives up.
- Success requires both to get better at the exact same pace.

Architecture



Credit: Little et al. (2021) "Generative Adversarial Networks for Synthetic Data Generation: A Comparative Study"

The Two-Player Game: A Zero-Sum Framework

- **A Zero-Sum Game**: In game theory, a zero-sum game means one player's gain is exactly equal to the other player's loss.
- **The Objectives**:
  - **The Discriminator** (D): Wants to maximize its accuracy. It wants to be right every time.
  - **The Generator** (G): Wants to maximize the Discriminator's error rate. It wins when the Detective is fooled.
- **Sequential Optimization**: Unlike a standard neural network where we minimize one loss, here we have a "**Minimax**" problem. We are trying to find a point of stability where neither player can improve their position (Nash Equilibrium).

Why "Adversarial"?

- In VAEs, the Encoder and Decoder work together to reconstruct the image (Cooperation).
- In GANs, G and D are in **direct conflict**.
- **The Result**: This conflict prevents the model from becoming "lazy" (like the blurriness in VAEs). If G produces a blurry image, D will easily flag it as "Fake," forcing G to sharpen up to survive.

Defining the Generator (G)

- **The Input (z)**: * Typically sampled from a Standard Normal $\mathcal{N}(0,1)$
    - This represents the "seed" of the image. Just like in VAEs, different points in z will result in different generated images.
- **The Transformation**:
    - The Generator is a **differentiable function** $G(z)$.
    - It uses learned weights to "unflatten" the noise, gradually adding spatial structure (lines, then shapes, then textures).
- **Final Activation**:
    - Usually uses **Tanh** (scaling pixels between -1 and 1) or **Sigmoid** (0 to 1).
    - **Crucial Point**: G never "sees" a real image. Its only source of truth is the gradient flowing back from the Discriminator.

Defining the Discriminator (D)

- **The Input:**
  - The Discriminator is a "dual-input" network (though it sees only one image at a time).
  - It receives **Real Samples** (x) from the dataset and **Fake Samples** ($G(z)$) from the Generator.
- **The Task:**
  - It performs **Binary Classification (**but produces a scalar value $\in [0,1]$**)**.
  - **Output** ≈ 1: "I am confident this is a real image from the training set."
  - **Output** ≈ 0: "I am confident this was created by the Generator."
- **The "Teacher" Role:**
  - When the Discriminator is correct, it penalizes the Generator.
  - When it is fooled, it provides the "Gradient" that tells the Generator how to improve.

System Components

- **The Balancing Act**: This is a dynamic system. Unlike a VAE, where the loss usually goes down smoothly, a GAN's "Total Loss" is hard to interpret because as G improves, D gets tougher.
- **Independent Updates**: We typically update D for one step, then G for one step. They take turns "learning" from the current state of the other.
- **The Competition is the Teacher**:
  - D learns by looking at x and $G(z)$.
  - G learns **indirectly** through D. G never sees x! It only "sees" x through the eyes of the Discriminator's criticism.

The Math of the Game: The Minimax Objective

- **The objective function**

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

- $D(x)$: Detective's score for real data (should be 1).
- $D(G(z))$: Detective's score for fake data (should be 0).
- $\min_G \max_D$: A game of tug-of-war

The Math of the Game: The Minimax Objective

- **The Max Player (Discriminator D):**
  - Wants to maximize the probability of assigning the correct label to both real and fake images.
  - It wants $D(x)$ to be high (close to 1) and $D(G(z))$ to be low (so $1 - D(G(z))$ is close to 1).
- **The Min Player (Generator G):**
  - Wants to minimize the Discriminator's ability to distinguish.
  - It only appears in the second term. It wants $D(G(z))$ to be high (fooling the critic), which makes $1 - D(G(z))$ small.

The Discriminator's Objective

- **Isolating the Math for D:**

When we train the Discriminator, we treat the Generator as a "frozen" entity and focus only on this part of the value function:

$$\max_D \mathscr{L}_D = \underbrace{\mathbb{E}_{x\sim p_{data}}[\log D(x)]}_{\text{Correctly Identify Real}} + \underbrace{\mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]}_{\text{Correctly Identify Fake}}$$

- **Term 1**: If $D(x)$ is 1.0, $\log(1) = 0$ (Perfect). If $D(x)$ is 0.1, $\log(0.1)$ is a large negative number (Penalty!).
- **Term 2**: If $D(G(z))$ is 0.0, $\log(1 - 0) = 0$ (Perfect). If $D(G(z))$ is 0.9, $\log(1 - 0.9)$ is a large negative number (Penalty!).

The Generator’s Objective

- **Isolating the Math for G:**

  The Generator only cares about the second half of the equation (the part involving its fakes). However, there is a "practical" twist in how we calculate it:

  - **The Theoretical Objective**: $\min_G \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$
    - This says: "Minimize the probability that the Detective is right about my fakes."
  - The Practical (Non-Saturating) Objective: $\max_G \mathbb{E}_{z \sim p_z}[\log D(G(z))]$
    - This says: "Maximize the probability that the Detective thinks my fakes are Real."

Why the "Practical" Switch?

- **The Gradient Problem**: Early in training, the Generator is terrible. The Discriminator easily identifies fakes ($D(G(z)) \approx 0$).

- In the theoretical version, the gradient is very **"flat"** here, so the Generator learns slowly.

- In the practical version, the gradient is very **"steep"** when $D(G(z))$ is near zero, giving the Generator a much-needed **"kick-start."**

Notation Reference

Symbol	Name	Description
x	Sample	A specific image (either real or fake).
p_{data}	Data Distribution	The actual distribution of real images in the world.
p_g	Model Distribution	The distribution of images the Generator is currently "hallucinating."
z	Latent Noise	The random input that gives the generator the variety to create different images.

Global Optimality: The Nash Equilibrium

- What happens if we have an infinitely powerful Generator and Discriminator?
- Optimal D: For a fixed G, the best possible Discriminator is:

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

- The Equilibrium: When the Generator is perfect ($p_g = p_{data}$), the formula becomes:

$$D^*(x) = \frac{p_{data}}{p_{data} + p_{data}} = \frac{1}{2}$$

- Meaning: The Discriminator is now completely confused. It can no longer find any feature to distinguish between real and fake. It is reduced to 50/50 guessing.

Optimal Discriminator

- First, we rewrite the expectation formula as an integral over the data space x:

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

- Since we want to maximize the integral, we can simply maximize the expression inside the integral for any given point x. Let's simplify the notation:
  - Let $a = p_{data}(x)$
  - Let $b = p_g(x)$
  - Let $y = D(x)$ (this is the value we want to find)
- Now we have a simple function to maximize:

$$f(y) = a\log(y) + b\log(1 - y)$$

Optimal Discriminator

- To find the maximum, we take the derivative with respect to y and set it to zero:

$$f'(y) = \frac{a}{y} - \frac{b}{1-y} = 0$$

- Solve for y

$$\frac{a}{y} = \frac{b}{1-y}$$

$$a(1-y) = by$$

$$a - ay = by$$

$$a = y(a+b)$$

$$y = \frac{a}{a+b}$$

- Substituting our original variables back in:

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

Loss function

- If we assume the Discriminator is always "at its best" ($D = D^*$), we can plug that optimal value back into our original Value Function. Through some algebraic magic, the equation transforms into:

$$V(G, D^*) = 2 \cdot JSD(p_{data} \parallel p_g) - \log(4)$$

- What is JSD?

  The Jensen-Shannon Divergence is a “smooth” and "symmetrized" version of the KL Divergence. It is defined as:

$$JSD(P \parallel Q) = \frac{1}{2} KL(P \parallel M) + \frac{1}{2} KL(Q \parallel M)$$

  Where M is the average distribution: $M = \frac{1}{2}(P + Q)$

Why Does This Matter for Images?

- **Symmetry**: Unlike VAEs, where the order of P and Q changes the behaviour, JSD treats the "Real" and "Fake" distributions as equal partners.
- **Finite Limits**: JSD is bounded between 0 and $\log(2)$. This prevents the "exploding gradients" often seen in other metrics.
- **Sharpness**: Because JSD doesn't have the same "zero-avoiding" behaviour as the KL divergence used in VAEs, the Generator isn't forced to "cover" every single real data point with a blurry average. It can focus on creating high-quality, distinct samples.

Loss function derivation

- Substitute $D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$ into the value function equation:

$$V(G, D^*) = \mathbb{E}_{x \sim p_{data}} \left[\log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right] + \mathbb{E}_{x \sim p_g} \left[\log \left(1 - \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right) \right]$$

- Simplify the "Fake" part: $1 - \frac{p_{data}}{p_{data} + p_g} = \frac{p_{data} + p_g - p_{data}}{p_{data} + p_g} = \frac{p_g(x)}{p_{data}(x) + p_g(x)}$.

$$V(G, D^*) = \mathbb{E}_{x \sim p_{data}} \left[\log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right] + \mathbb{E}_{x \sim p_g} \left[\log \frac{p_g(x)}{p_{data}(x) + p_g(x)} \right]$$

- To turn these into KL Divergences, we need the denominator to be the average of the two distributions ($M = \frac{p_{data} + p_g}{2}$).
- To get that "2" in there, we multiply the top and bottom of the fractions by 2:

$$V(G, D^*) = \mathbb{E}_{x \sim p_{data}} \left[\log \frac{p_{data}(x)}{2 \cdot \frac{p_{data}(x) + p_g(x)}{2}} \right] + \mathbb{E}_{x \sim p_g} \left[\log \frac{p_g(x)}{2 \cdot \frac{p_{data}(x) + p_g(x)}{2}} \right]$$

Loss function derivation

- Using log rules ($\log \frac{A}{2B} = \log \frac{A}{B} - \log 2$), we pull out the $\log 2$ from both terms:

$$V(G, D^*) = \left(\mathbb{E}_{x \sim p_{data}} \left[\log \frac{p_{data}(x)}{M(x)} \right] - \log 2 \right) + \left(\mathbb{E}_{x \sim p_g} \left[\log \frac{p_g(x)}{M(x)} \right] - \log 2 \right)$$

- The definition of KL Divergence is $KL(P \parallel Q) = \mathbb{E}_{x \sim P}[\log \frac{P(x)}{Q(x)}]$.

$$V(G, D^*) = KL(p_{data} \parallel M) + KL(p_g \parallel M) - 2 \log 2$$

- By definition, $JSD(P \parallel Q) = \frac{1}{2} KL(P \parallel M) + \frac{1}{2} KL(Q \parallel M)$.
- Therefore, $2 \cdot JSD = KL(P \parallel M) + KL(Q \parallel M)$.
- Substituting that in (and noting that $2 \log 2 = \log 4$):

$$V(G, D^*) = 2 \cdot JSD(p_{data} \parallel p_g) - \log 4$$

Visualizing the Training: The Converging Distributions

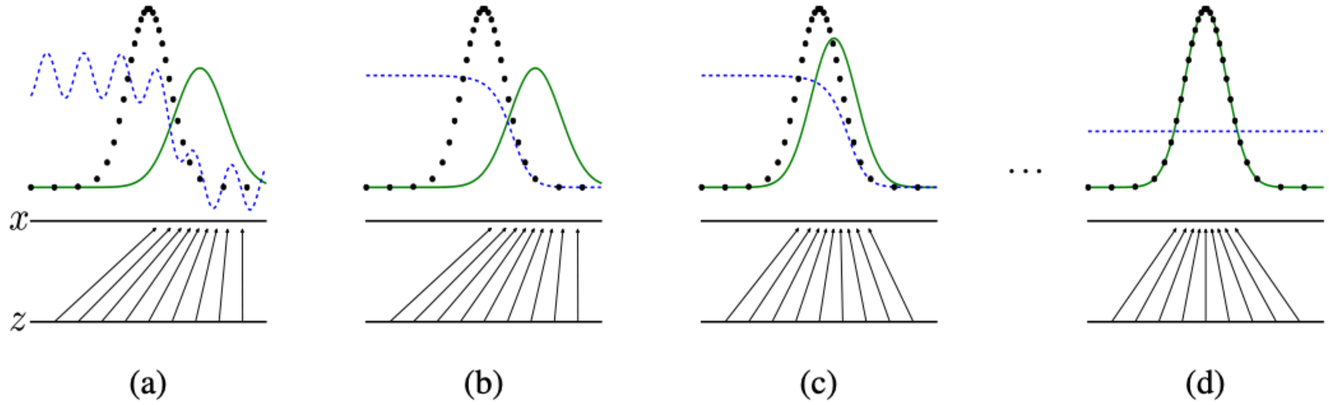


Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the **g**enerative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which $\boldsymbol{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$. (c) After an update to G, gradient of D has guided $G(\boldsymbol{z})$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(\boldsymbol{x}) = \frac{1}{2}$.

Credit: Goodfellow et al. (2014) “Generative Adversarial Nets”

Why Does the Blue Line Flatten?

- As the green line (p_g) becomes identical to the black line (p_{data}), there is no longer any statistical difference between them. At that point:

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} = \frac{p_{data}(x)}{2p_{data}(x)} = 0.5$$

- The Discriminator is now "guessing" because the forgery is perfect.

The Training Loop

- **Update the Discriminator** (D):
  - **Sample**: Grab a batch of real images (x) and a batch of noise (z).
  - **Forward Pass**:
    - $D(x)$: Score the real images (Target = 1).
    - $D(G(z))$: Score the fake images produced by the current G (Target = 0).
  - **Backprop**: Calculate the total loss and update only the weights of D.
  - **Goal**: Make D a better critic for this specific version of G.
- **Update the Generator** (G):
  - **Sample**: Grab a new batch of noise (z).
  - **Forward Pass**: Pass $G(z)$ into the Discriminator.
  - **The Trick**: Calculate the loss using the Label = 1 (as if the fakes were real).
  - **Backprop**: Flow the error from the Discriminator's output back into G. Crucially, freeze D's weights during this step.
  - **Goal**: Shift G's output to a region where D thinks the images are real.

Challenges in Training (The Dark Side of GANs)

- **Mode Collapse (The "One-Trick Pony")**
  - **The Problem**: The Generator discovers one specific type of image that the Discriminator currently finds "very real" (e.g., a really good looking '7' in the MNIST dataset).
  - **The Result**: Instead of learning to make all digits, the Generator maps every input z to that same '7'.
  - **The Loop**: The Discriminator eventually realizes all '7's are fake, then the Generator moves to a '1', and they circle each other forever without covering the whole dataset.
- **Vanishing Gradients**
  - **The Problem**: If the Discriminator is too good, its output is $0.000000...1$ for fakes.
  - **The Result**: Looking at the math of the Sigmoid function, the gradient becomes nearly zero in these "saturated" regions.
  - **The Consequence**: The Generator stops learning because its "teacher" is just giving a "0" grade without explaining why it failed.
- **Non-Convergence**
  - **The Problem**: Because G and D are playing a game, they can enter an infinite loop of "chasing" each other's weights (Oscillation) rather than settling down.

Non-Convergence

- In standard Deep Learning (like an Image Classifier), we use **Gradient Descent** to find a local minimum of a fixed loss surface. It's like finding the bottom of a valley.
- In GANs, we are looking for a **Nash Equilibrium** in a game between two players. This is much harder because:
  - **Moving Targets**: When the Generator (G) updates its weights to fool the Discriminator (D), it changes the "landscape" for D. When D updates to catch G, it changes the landscape for G.
  - **Lack of Objective Function**: There is no single "mountain" we are climbing. Instead, we have a "Min-Max" objective where the surface is constantly warping.
  - **Oscillations**: Instead of converging to a single point, the models often "chase" each other in circles. G learns to create a specific feature, D learns to reject it, G abandons that feature for a new one, and D follows.

Evaluation Metrics

- **The Challenge**: Standard loss functions (like MSE) don't work for GANs because we don't have a "target" image to compare against. We need a way to quantify two things:
  - **Fidelity**: Do the images look like clear, recognizable objects?
  - **Diversity**: Does the model generate a wide variety of objects, or just the same thing over and over?
- Instead we use the **following metrics**:
  - Inception Score (IS)
  - Frechet Inception Distance (FID)

Inception Score (IS)

- This metric uses a pre-trained image classifier (Google's **Inception v3**) to "look" at the generated images.
  - **Sharpness (Low Entropy)**: When the model sees a fake image, is it confident it's a specific class (e.g., "That is definitely a dog")?
  - **Diversity (High Entropy)**: Over a large batch of images, does the model generate many different classes (dogs, cars, trees), or just one?
  - The Goal: Higher is better.

Deep Dive: Inception Score (IS)

- The Inception Score relies on two mathematical properties of the output from a pre-trained **Inception v3** classifier:
- **Individual Quality (Conditional Probability $p(y|x)$):**
  - We feed one fake image x into the classifier.
  - If the image is sharp and recognizable, the classifier should output a high probability for **one specific class** (e.g., 0.99 for "Zebra").
  - We want the **entropy to be low** for a single image.
- **Global Diversity (Marginal Distribution $p(y)$):**
  - We feed a large batch (e.g., 50,000) of fake images.
  - We average all the label predictions. If the GAN is diverse, the average should be uniform (10% dogs, 10% planes, etc.).
  - We want the **entropy to be high** for the whole batch.
- The Formula:
  - Measure how different the "specific" label of one image is from the "average" labels of all images.

$$IS(G) = \exp(\mathbb{E}_{x \sim p_g}[KL(p(y|x) \parallel p(y))])$$

The Problems with IS

- While it was the first major metric, it has significant blind spots
- **"Internal" Mode Collapse**:
  - IS can't tell if you are generating 1,000 different Dalmatians or the exact same Dalmatian 1,000 times.
  - As long as it looks like a Dalmatian, the score stays high.
- **Classifier Bias:**
  - If your GAN generates something that isn't in the 1,000 ImageNet classes (like a specific medical X-ray), the Inception model won't know what to do with it, rendering the score meaningless.
  - IS is only guaranteed to work if the classes you are interested in is contained within ImageNet

The Problems with IS

- **No Comparison to Reality**:
  - IS only looks at the fake images.
  - It doesn't actually check if the fakes look like the specific real dataset you used.
- **Vulnerability to Artifacts:**
  - Sometimes, "adversarial noise" (random-looking pixels) can trick the Inception network into being very confident about a class, giving a high score to an image that looks like garbage to a human.
  - Needs constant human validation

Fréchet Inception Distance (FID)

- **The Problem with IS**: The Inception Score never actually looks at the real training data. It only cares if the fakes are recognizable.
- **The Solution**: FID measures how similar the distribution of generated images is to the distribution of real images.
- How it Works (The Feature Space):
  - We take a batch of **Real Images** and a batch of **Fake Images**.
  - We feed both through a pre-trained **Inception v3** network.
  - We "capture" the data at the **coding layer** (the last pooling layer before the final classification).
  - This layer represents high-level features (eyes, wheels, textures) rather than raw pixels.

The Math of FIS: Comparing Gaussians

- We represent the features of the real images as a Gaussian distribution with mean μ_r and covariance Σ_r. We do the same for the fakes (μ_g, Σ_g).
- The FID is the **Wasserstein-2 distance** between these two Gaussians:

$$d^2((\mu_r, \Sigma_r), (\mu_g, \Sigma_g)) = \|\mu_r - \mu_g\|^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$$

  - **Part 1** ($\|\mu_r - \mu_g\|^2$): Do the fakes have the same average features as the real images?
  - **Part 2** ($Tr(\ldots)$): Do the fakes have the same variety and correlations as the real images?
  - The Goal: Lower is better.

Why FID is Better than IS

- **Consistency**: FID is much more robust to noise.
- **Diversity Check**: If the Generator suffers from Mode Collapse (e.g., it only makes one type of dog), its covariance Σ_g will be very different from the real data's Σ_r, resulting in a bad (high) score.
- **Sensitivity**: If images are blurry or have weird artifacts, the Inception features will shift, and the FID will increase.

Summary — Mastering the GAN Framework

- **The Fundamental Game**
  - GANs are a Zero-Sum game where the Generator (G) creates and the Discriminator (D) critiques.
  - The equilibrium is reached when $D(x) = 0.5$, meaning the forgeries are indistinguishable from reality.
- **The Math of Success**
  - Under optimal conditions, the GAN objective minimizes the **Jensen-Shannon Divergence** (JSD).
  - Unlike VAEs, this metric doesn't force the model to be "blurry," leading to the **sharp, high-fidelity images** GANs are famous for.

Summary — Mastering the GAN Framework

- **Measures**
  - **Inception Score (IS)**: Focuses on "recognizable objects" and "class variety."
  - **FID**: It uses the **Trace** of covariance matrices to ensure the fake distribution matches the real distribution in feature space.
- **Hard Truths of Training**
  - **Instability**: The "rotational" gradients mean the models often orbit the solution rather than hitting it.
  - **Mode Collapse**: The risk of the Generator becoming a "one-trick pony" by ignoring the diversity of the dataset.