Generative Adversarial Networks

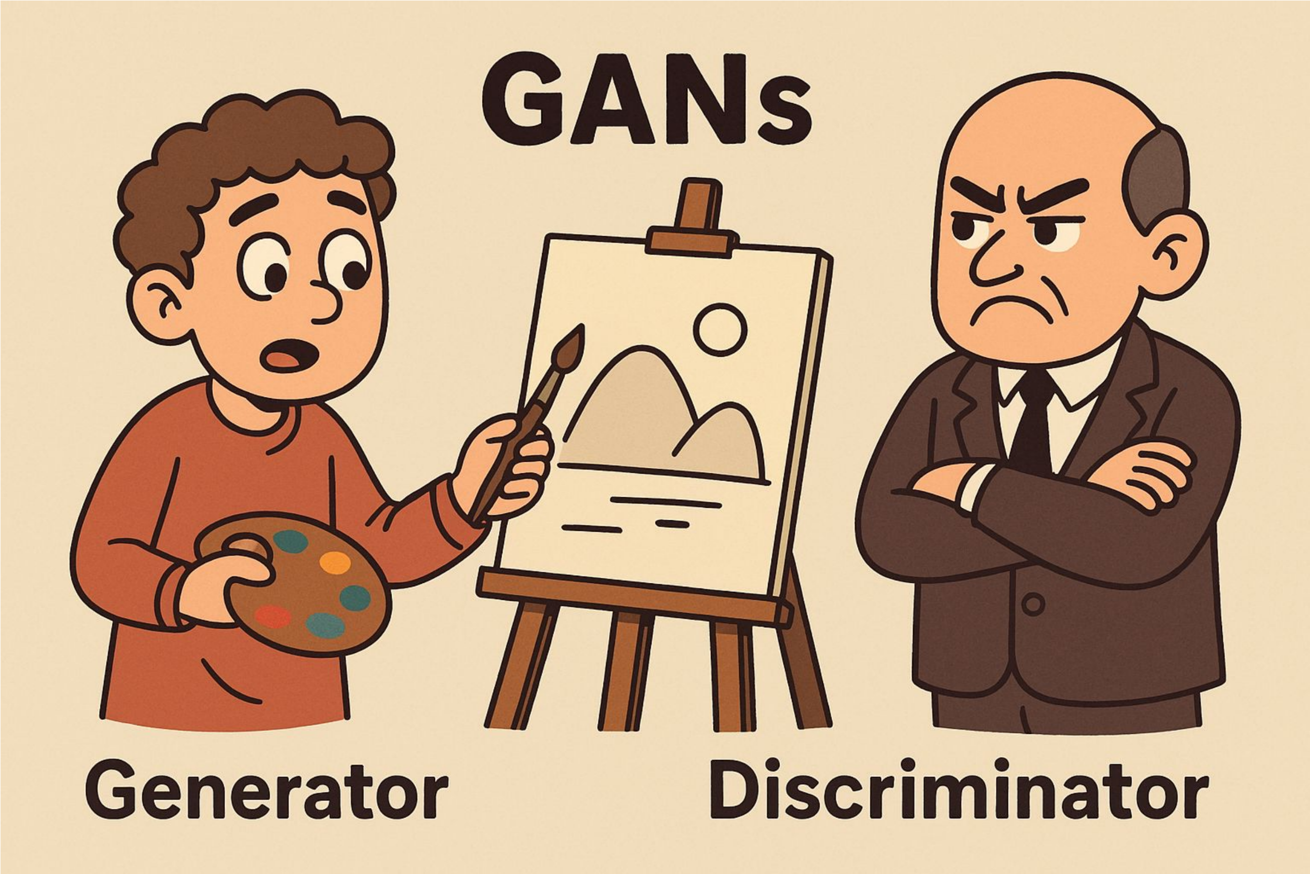
Presented By: Massa Baali

11785 Deep Learning
Spring 2025
(Based on slides from Ben Striner and Bhiksha Raj)

Outline

- Definition of GAN
- Discriminative vs. Generative Models
- Explicit vs. Implicit Models
- Problem
- What are GANs?
    - Generator definition
    - Discriminator definition
- Training GAN
- Perfect discriminator

Definition



Two neural networks:

- **Generator:** trying to create something realistic - whether it's an image, speech, or video.
- **Discriminator:** playing the role of the harsh critic, rejecting anything that doesn’t seem real.

Discriminative vs. Generative Models

- **Discriminative models learn to discriminate**
    - Determine the class given the input
        - Compute P(y|x)
- **Generative models can generate**
    - Produce more instances like the training data
        - Compute and/or draw from P(x,y)

Discriminative vs. Generative Models

Given a distribution of inputs X and labels Y

Discriminative models	Generative models
Discriminative models learn conditional distribution P(Y | X)	Generative models learn the Generative models learn the joint distribution P(Y, X)
Learns decision boundary between classes.	Learns actual probability distribution of data.
Limited scope. Can only be used for classification tasks.	Can do both generative and discriminative tasks.
E.g. Logistic regression, SVM etc.	E.g. Naïve Bayes, Gaussian Mixture Model etc.

Harder problem, requires a deeper understanding of the distribution than discriminative models.

Explicit vs. Implicit Models

Explicit distribution models

Calculates P(x ~ X) for all x

Implicit distribution models

Generate x ~ X

Poll 1

- **What is the difference between Discriminative models vs. Generative models**

– Discriminative models model the decision boundary between classes, whereas Generative models model class distributions

– Generative models model the decision boundary between classes, whereas Discriminative models model class distributions

- **What is the difference between Explicit and Implicit Generative models?**

– Implicit models compute the probability of samples, whereas Explicit models only let you draw samples from the distribution

– Explicit models compute the probability of samples, whereas Implicit models only let you draw samples from the distribution

Poll 1

- **What is the difference between Discriminative models vs. Generative models**

– Discriminative models model the decision boundary between classes, whereas Generative models model class distributions

– Generative models model the decision boundary between classes, whereas Discriminative models model class distributions

- **What is the difference between Explicit and Implicit Generative models?**

– Implicit models compute the probability of samples, whereas Explicit models only let you draw samples from the distribution

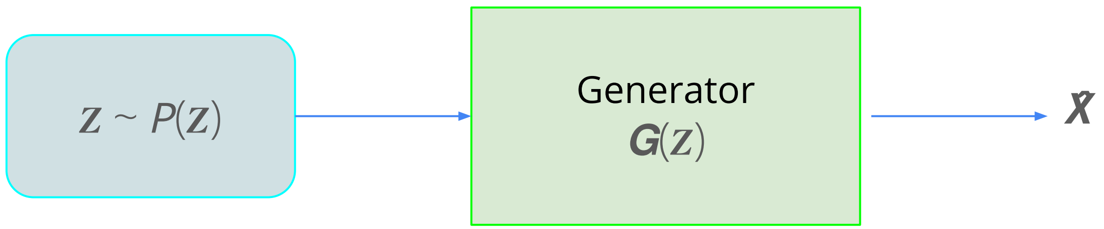
– Explicit models compute the probability of samples, whereas Implicit models only let you draw samples from the distribution

Problem

Given a large collection of face images, can a neural network learn to generate entirely new portraits? In other words, can we model and sample from the underlying distribution of faces?

- How do we even characterize this distribution?

What we have seen: VAE



$z \sim P(z)$

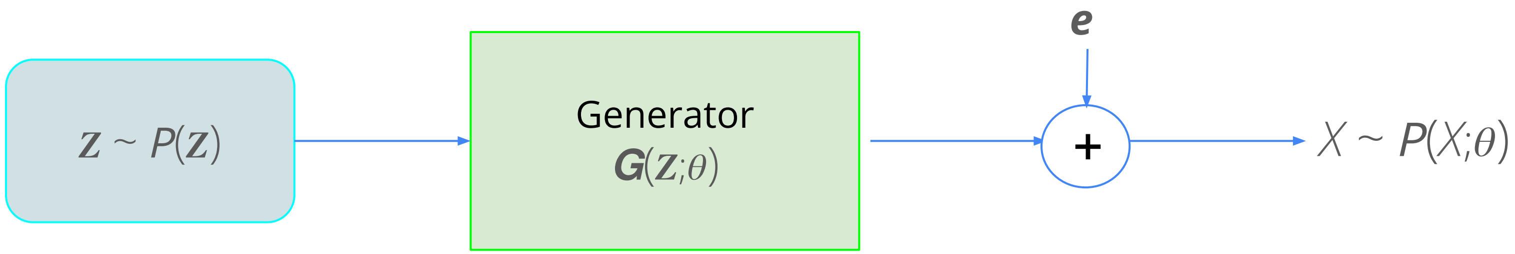
Generator

$G(z)$

$\tilde{X}$

- We **can't directly characterize** the full distribution in high-dimensional space.
- Underlying hypothesis: Real-world data is **structured**—it does not occupy the entire high-dimensional space.
- Instead, data lies on a lower-dimensional manifold embedded in the high-dimensional space
- This latent code is then **transformed** via a neural network to generate high-dimensional outputs using the **Generator**.
- Generator is a decoder of a VAE

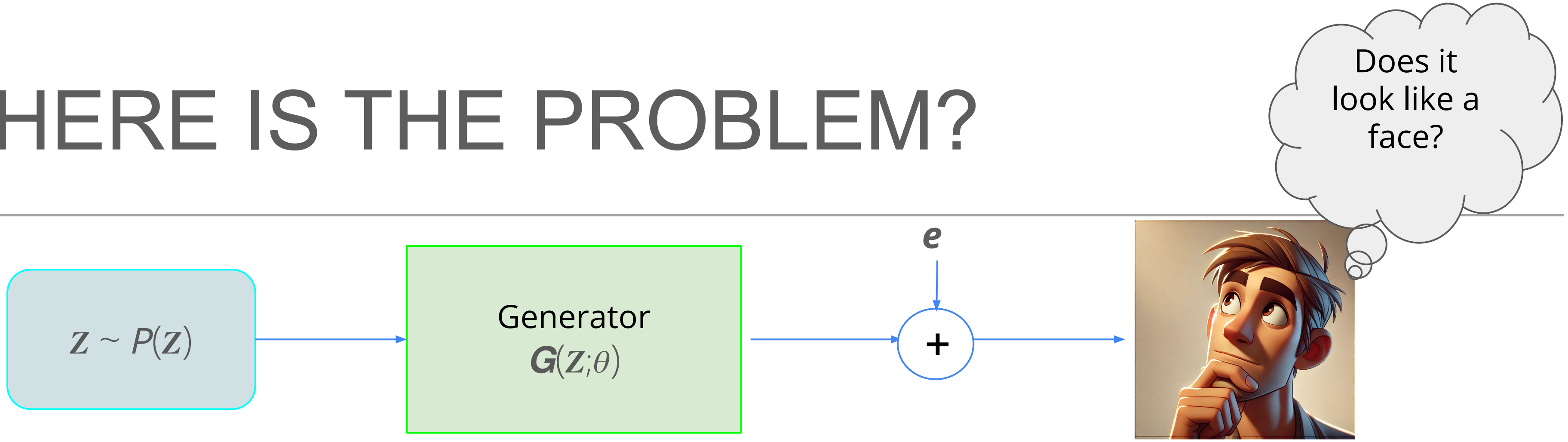
What we have seen: VAE



- Mapping z→G(z) alone cannot fully explain the data distribution.
- We introduce a correction term: random noise e that accounts for any errors in choosing the right dimensionality of the space.
- This defines a generative model: it can now draw diverse samples that better match real data.
- Trained by maximizing the likelihood of the data $\theta^* = \arg\max_\theta \quad \log P(X;\theta)$
- Equivalently, we minimize the likelihood of the data as seen in the VAE lecture

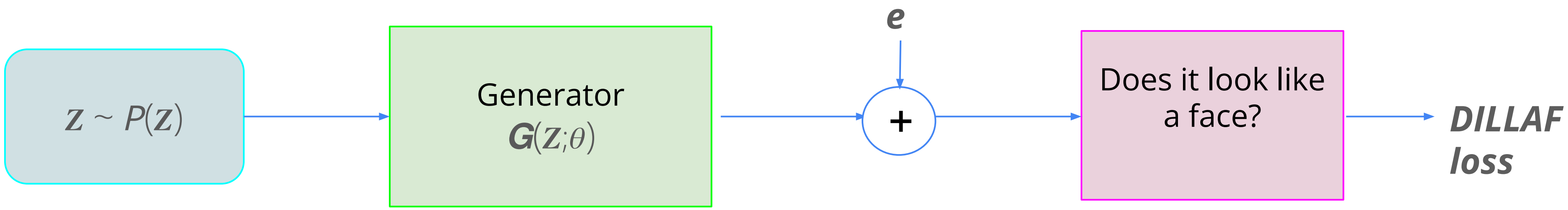
$$\theta^* = \arg\min_\theta \quad -\log P(X;\theta)$$

WHERE IS THE PROBLEM?



- Likelihood maximization does not actually relate to whether the output actually looks like a face
- It doesn't guarantee that generated samples look like real faces.
- To verify this, we need to check if the output looks like a face.
- Manually inspecting samples isn’t scalable—we need to **automate** this check.

WHERE IS THE PROBLEM?



- Replacing the human evaluator with a classifier
- The loss of the classifier is the (DILLAF) loss.

Poll 2

- **VAEs are implicit Generative models, True or False**

  – True

  – False

- **Why would likelihood maximization not result in a model that produces more facelike outputs (for a face-generating VAE)?**

  – The model can maximize the likelihood of training data without any assurance about what other (non-training) samples look like

  – The model is more likely to run into poor local optima

  – The model only captures the mode of the distribution of faces, whereas most face-like images are in the tail of the distribution

- **The face-generating model is more likely to generate face-like images if it were trained with a differentiable loss function that explicitly evaluates if the outputs look like faces or note, True or False**

  – True

  – False

Poll 2

- **VAEs are implicit Generative models, True or False**

  – True

  – False

- **Why would likelihood maximization not result in a model that produces more facelike outputs (for a face-generating VAE)?**

  – The model can maximize the likelihood of training data without any assurance about what other (non-training) samples look like

  – The model is more likely to run into poor local optima

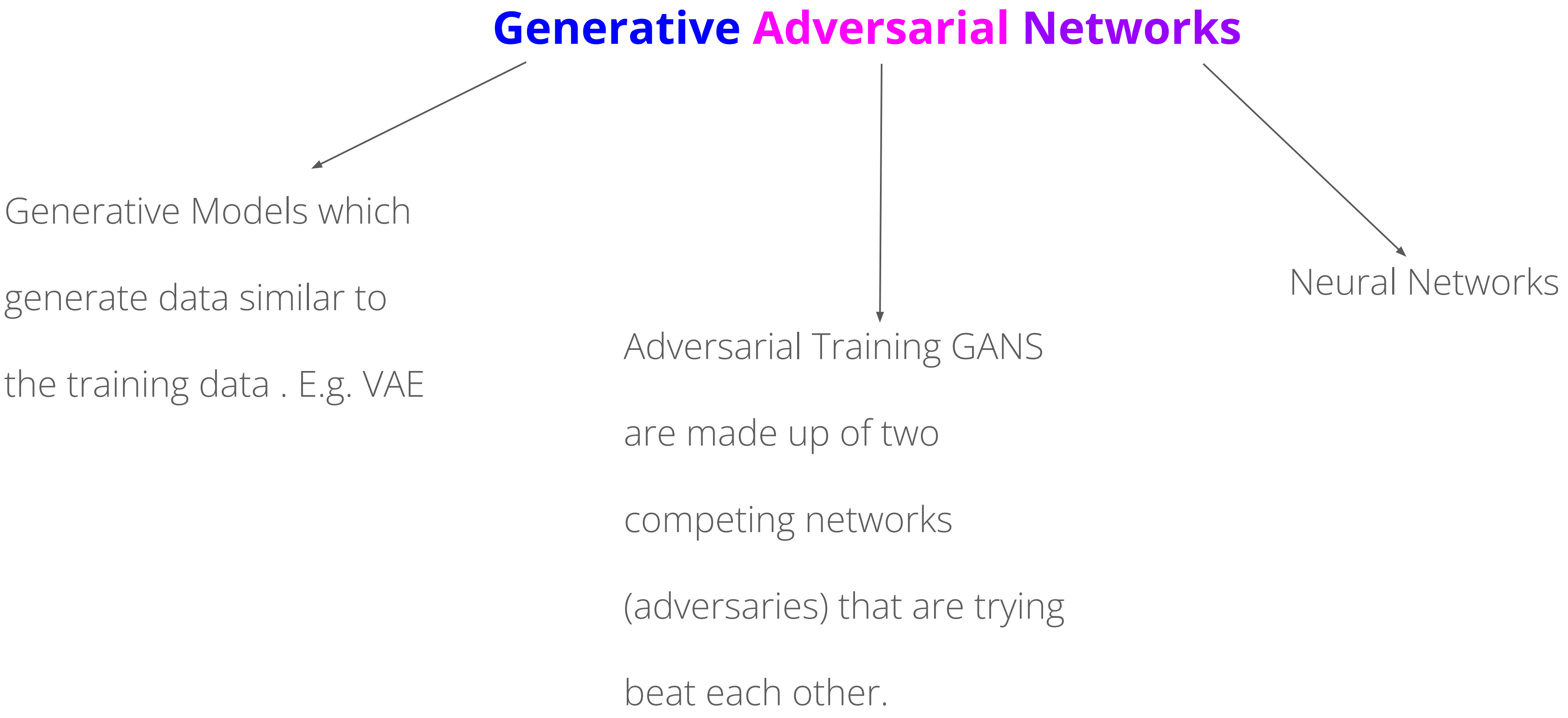
  – The model only captures the mode of the distribution of faces, whereas most face-like images are in the tail of the distribution

- **The face-generating model is more likely to generate face-like images if it were trained with a differentiable loss function that explicitly evaluates if the outputs look like faces or note, True or False**

  – True

  – False

What are GANs?



Generative Adversarial Networks

Generative Models which generate data similar to the training data . E.g. VAE

Adversarial Training GANS are made up of two competing networks (adversaries) that are trying beat each other.

Neural Networks

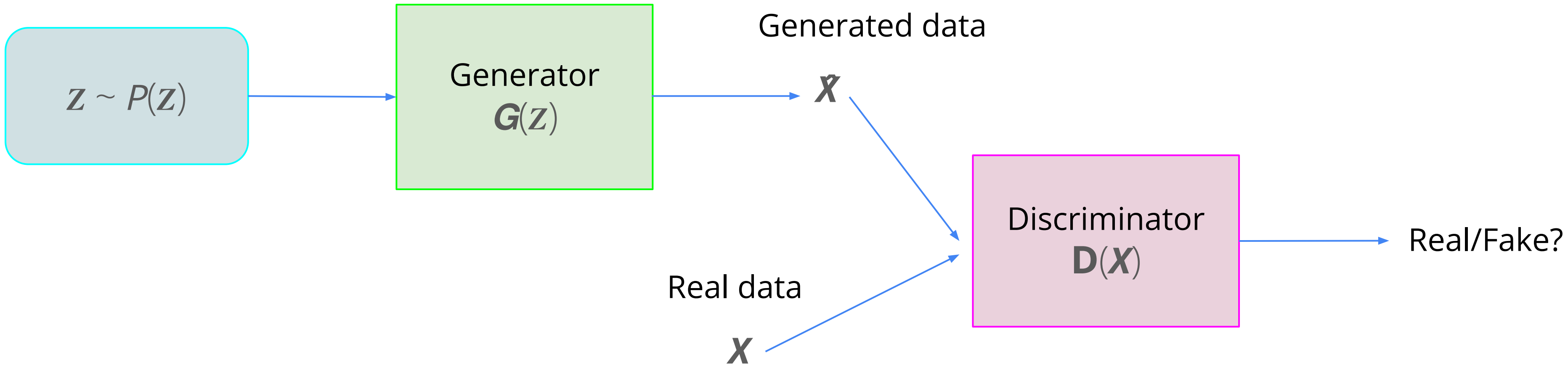
What are GANs?

- Introduced in 2014
- Goal is to model *P*(*X*), the distribution of training data

  – Model can generate samples from *P*(*X*)

- Trained using a pair of models acting as “adversaries”
  - A “Generator” that generates data
  - A “Discriminator” that evaluates it
    - The DILLAF loss!!

What are GANs?



$z \sim P(z)$

Generator

$G(z)$

Generated data

$\tilde{X}$

Real data

X

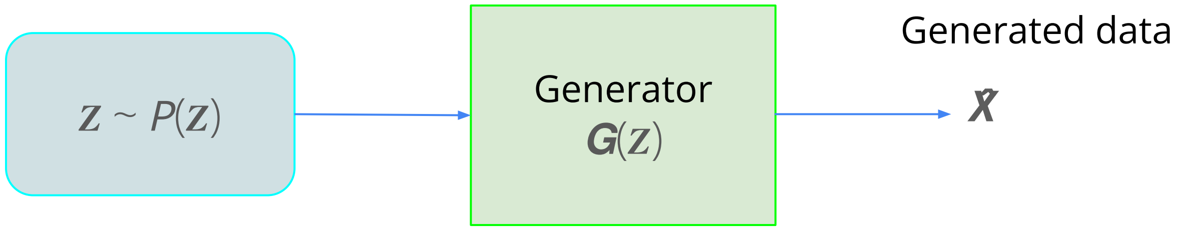
Discriminator

$D(X)$

Real/Fake?

DILLAF loss

The Generator



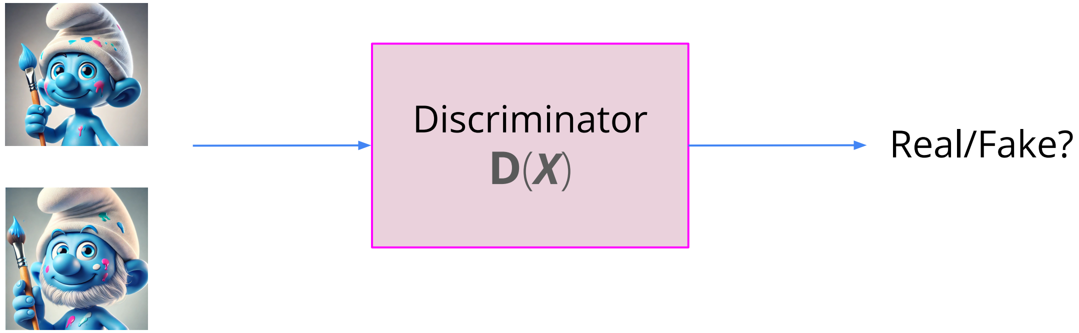
- The generator produces realistic looking $X = \boldsymbol{G}(z)$ from a latent vector $\boldsymbol{z}$

- Generator input $\boldsymbol{z}$ can be sampled from a known prior, e.g. standard Gaussian

- Goal: generated distribution, $P_G(\boldsymbol{X})$ matches the true data distribution $P_X(\boldsymbol{X})$□

  - $P_G(\boldsymbol{X})$ is the more “memorable” notation for $P_X(\boldsymbol{X})$, the probability that a generated sample $\tilde{\boldsymbol{X}}$ takes the value $\boldsymbol{X}$.

The Discriminator



Discriminator
D(***X***)

Real/Fake?

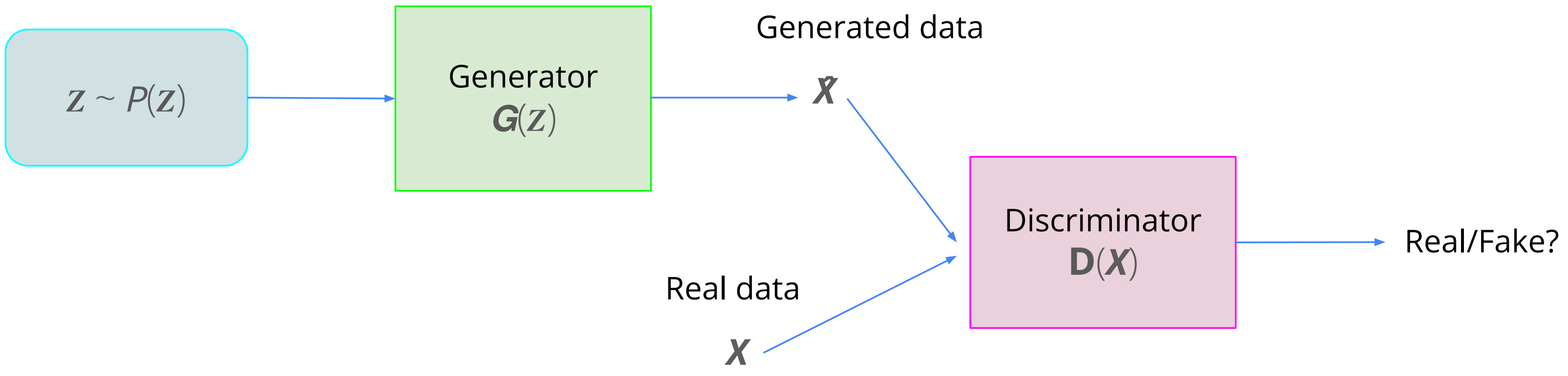
- Discriminator **D**(***X***) is trained to tell the difference between real and generated (fake) data

- Specifically, data produced by the generator

- If a perfect discriminator is fooled, the generated data

cannot be distinguished from real data

Training a GAN



$z \sim P(z)$

Generator

$G(z)$

Generated data

$\tilde{X}$

Real data

X

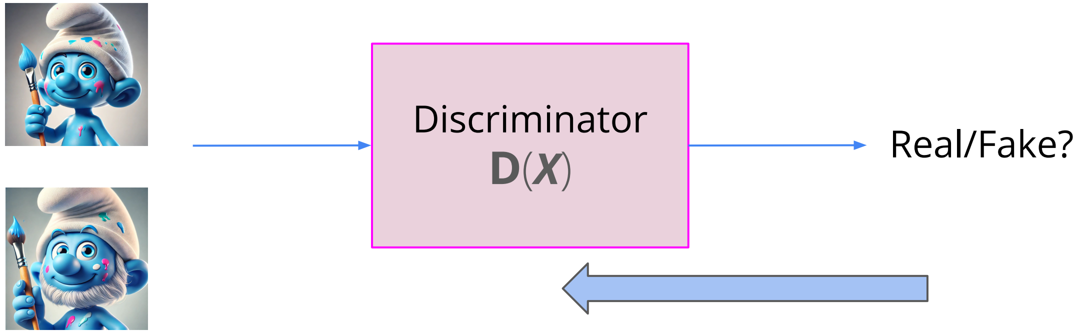
Discriminator

$D(X)$

Real/Fake?

Both, the generator and discriminator must be trained

Training the discriminator



Training the discriminator:

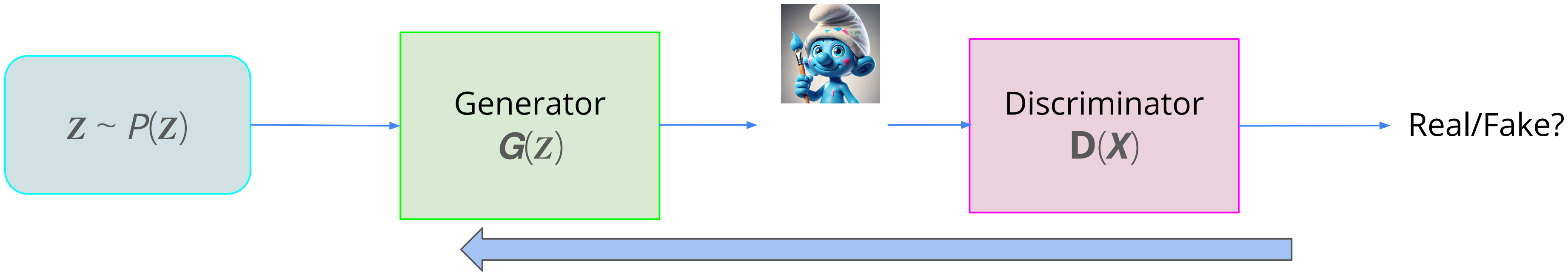
– The discriminator is provided training examples of real and synthetic faces
– The discriminator is trained to minimize its classification loss

- Minimize error between actual and predicted labels

- Discriminator parameters are trained such that

- **D**(***X***) = 1 for real faces. Maximize log **D**(***X***)
- **D**(***X***) = 0 for synthetic faces. Maximize log (1- **D**(***X***))

Training the generator



Training the generator:

– The discriminator’s loss is backpropagated to the generator

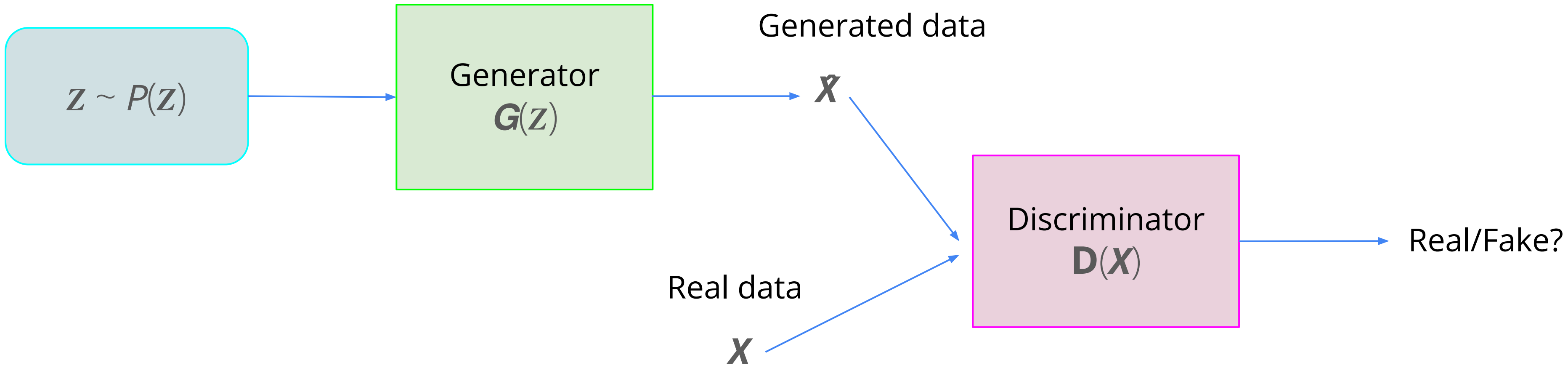
– The generator is trained to maximize the discriminator loss

- It is trained to “fool” the discriminator

- Generator parameters are trained such that

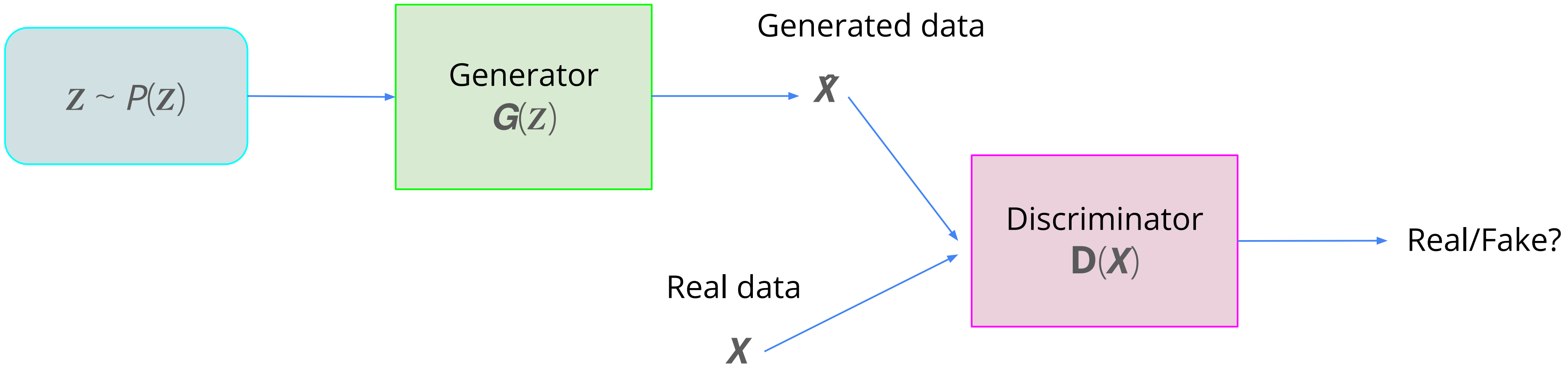
- **D**($\boldsymbol{G}(z)$) = 1 (i.e. 1- **D**($\boldsymbol{G}(z)$) = 0). Minimize log(1- **D**($\boldsymbol{G}(z)$))

The GAN formulation



- Discriminator
  - For real data X, Maximize log $\mathbf{D}(X)$
  - For synthetic data, Maximize log (1- $\mathbf{D}(\tilde{X})$)
- Generator
  - Minimize log (1- $\mathbf{D}(\tilde{X})$)

The GAN formulation



$z \sim P(z)$

Generator

$G(z)$

Generated data

$\tilde{X}$

Real data

X

Discriminator

$D(X)$

Real/Fake?

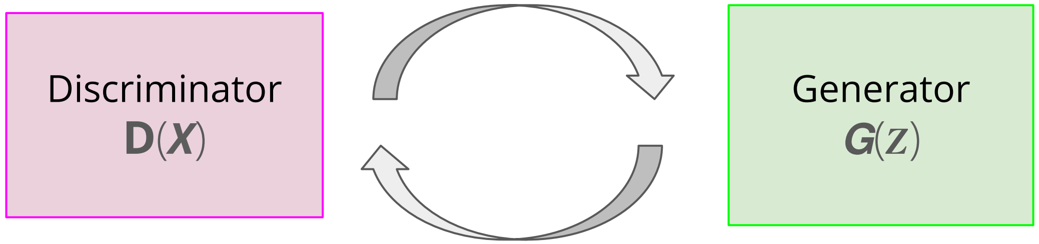
- The original GAN formulation is the following min-max optimization

$$\min_G \max_D \quad E_X \log D(X) + E_Z \log(1 - D(G(Z)))$$

Objective of D: $D(X) = 1$ and $D(G(Z)) = 0$

Objective of G: $D(G(Z)) = 1$

How to train a GAN?



Step 1: Train the Discriminator using the current Generator

Step 2: Train the Generator to beat the Discriminator

Optimize: $\min_G \max_D \; E_X \log D(X) + E_Z \log(1 - D(G(Z)))$

The discriminator is not needed after convergence

Poll 3

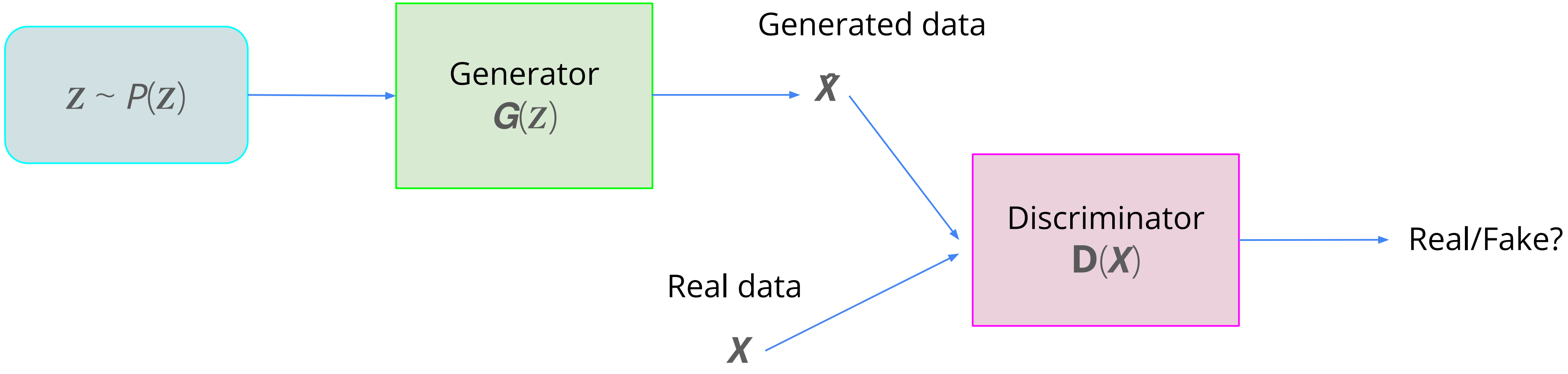
- **When training a GAN, which component must you train first**
  - The discriminator
  - The generator
- **Which component is updated more frequently**
  - The discriminator
  - The generator

Poll 3

- **When training a GAN, which component must you train first**
  - – The discriminator
  - – The generator
- **Which component is updated more frequently**
  - – The discriminator
  - – The generator

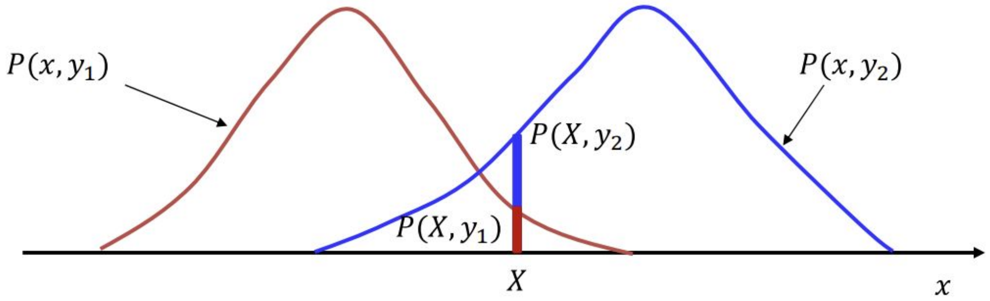
The discriminator is the (DILLAF) loss. Training the loss is more important, since the loss guides the training!

How does it behave?



So how does this behave when each component is optimized…

The perfect discriminator



- An ideal discriminator should separate real faces from fake ones.

- But in practice, distributions overlap → perfect separation is hard.

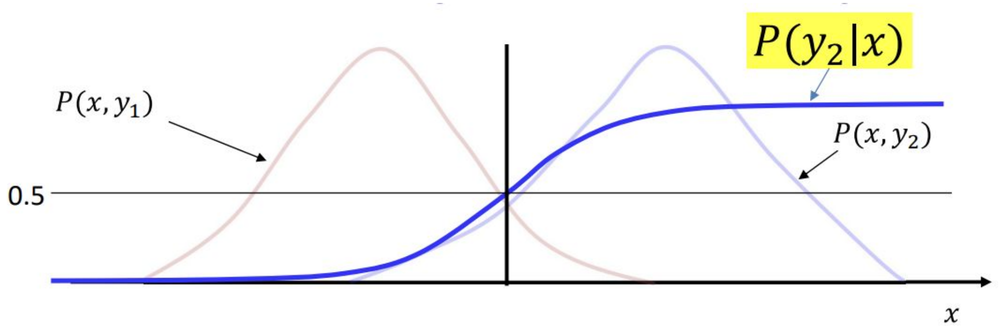
- The optimal discriminator is the one with lowest possible error.

- It estimates the probability of a sample being real, given x.

- Mathematically: the a posteriori probability of the classes for any instance x = X is:

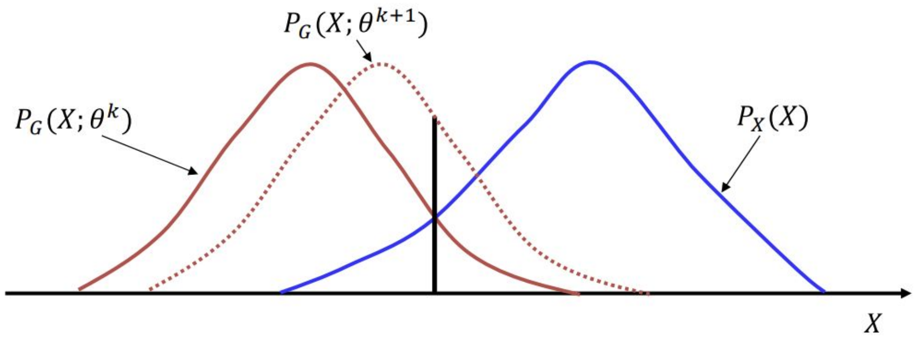
$$P(y_i|X) = \frac{P(X,y_i)}{P(X,y_1)+P(X,y_2)}$$

The perfect discriminator



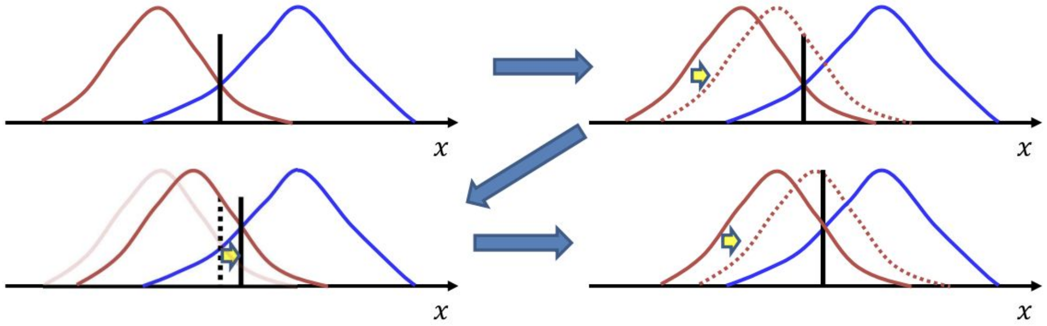
- At the start, the generator's output is poor (random noise or bad faces).
- The discriminator quickly learns the decision boundary between real and fake.
- A well-trained discriminator is key—it gives useful feedback to guide the generator.
- Once the discriminator is solid, the generator shifts its output distribution to fool it.
- This back-and-forth improves the generator over time.

Updating the Generator: Fooling the perfect discriminator

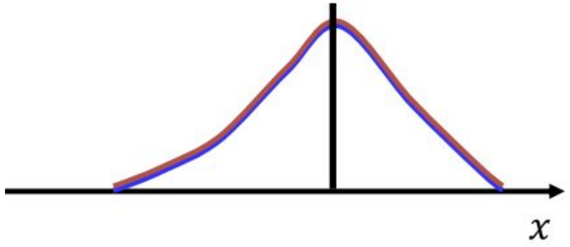


- The generator updates its parameters to improve output.
- It shifts its distribution closer to real data (e.g., moves right).
- The goal is to fool the discriminator into thinking generated samples are real.
- This process continues as training goes on.

Iterated learning



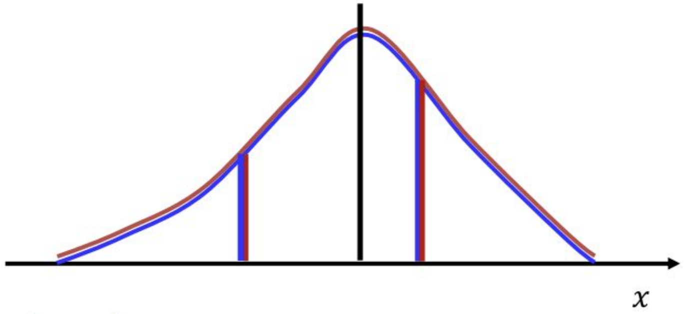
- ○ Discriminator learns perfect boundary
- ○ Generator moves its distribution past the boundary “into”
- ○ the real distribution
- ○ Discriminator relearns new “perfect” boundary
- ○ Generator shifts distribution past new boundary
- ○ …
- ○ In the limit Generator’s distribution sits perfectly on “real”
- ○ distribution and the perfect discriminator is still random



What Happens When GANs Are Perfectly Trained?

- The discriminator becomes a perfect judge — it separates real and fake really well.
- The generator keeps updating until its output looks exactly like real data.
- This happens by minimizing the difference between real and fake distributions.
- That difference is called Jensen-Shannon Divergence (JSD) — a measure of similarity.
- When the generator is perfect:
  - It fools the discriminator every time.
  - The discriminator outputs 0.5 for everything (can’t tell real from fake).
  - No more learning — both networks stop improving.

The optimal generator with the optimal discriminator



- The generator of the fully optimized GAN will generate $P_{\boldsymbol{G}}(X) = P_{\boldsymbol{X}}(\boldsymbol{X})$, i.e. the distribution of the generated data will be identical to that of the original data

Jensen-Shannon Divergence (JSD)

○ This training procedure minimizes the JSD between fake and real distributions.

$$
\begin{aligned}
&JSD\ (P, Q) \\
&= \ 0.5\ KL(P, 0.5(P + Q)) \ + \ 0.5KL(Q, 0.5(P + Q))
\end{aligned}
$$

Min-Max Stationary Point

There exists a stationary point:

- If the generated data exactly matches the real data, the discriminator outputs 0.5 for all inputs
- If discriminator outputs 0.5, the gradients for the generator is flat, so generator does not learn
- Unfortunately, this is also true of a random discriminator

Min-Max Optimization

- Generator and the discriminator need to be trained simultaneously

  – If discriminator is undertrained, it provides sub-optimal feedback to the generator

  – If the discriminator is overtrained, there is no local feedback for marginal improvements

Poll 4

- **Identify potential reasons a GAN could fail**

– Generator always generates the same face that fools the discriminator

– The JSD may have poor derivatives preventing the model from learning

– The discriminator may be random resulting in no derivatives

– The discriminator may be too certain, resulting in no derivatives

Poll 4

- **Identify potential reasons a GAN could fail**

– Generator always generates the same face that fools the discriminator

– The JSD may have poor derivatives preventing the model from learning

– The discriminator may be random resulting in no derivatives

– The discriminator may be too certain, resulting in no derivatives

Thank you :-)