Generative Aversarial Networks (Based on slides from Ben Striner and Hao Chen)

11785 Deep Learning
Spring 2026
Presented: Bhiksha Raj and Mengchun Zhang
Attendance: @789

What we've learned so far

- VAEs
- Diffusion models

- This week: GANS

- Connecting the dots

Discriminative vs Generative Models

Given a distribution of inputs X and labels Y.

Discriminative models

- Discriminative models learn conditional distribution P(Y | X)
- Learns decision boundary between classes.
- Limited scope. Can only be used for classification tasks.
- E.g. Logistic regression, SVM etc.

Generative models

- Generative models learn the Generative models learn the joint distribution P(Y, X)
- Learns actual probability distribution of data.
- Can do both generative and discriminative tasks.
- 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 : @790 @791

- 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

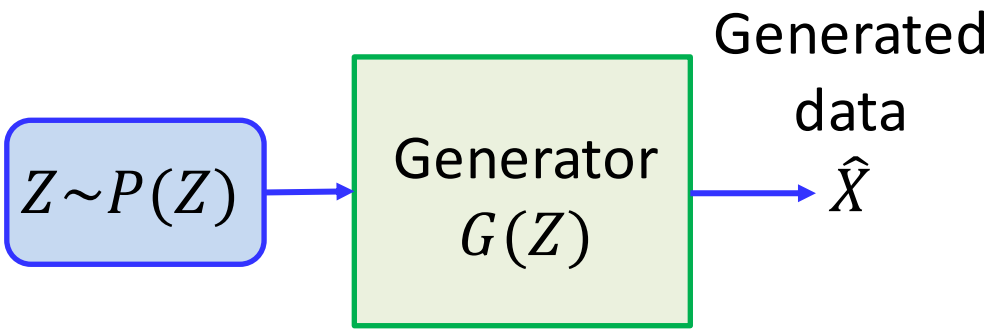
- What is the difference between Discriminative models vs. Generative models
  - **Discriminative models model the decision boundary between classes, whereas Generative models model class distributions**
  - Generative models model the decision boundary between classes, whereas Discriminative models 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**

The problem

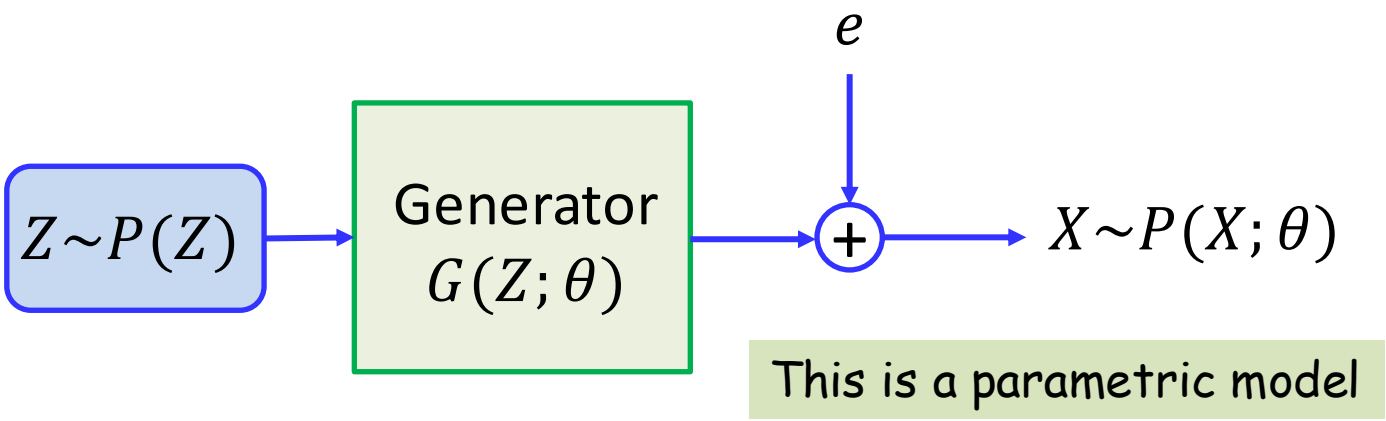
- From a large collection of images of faces, can a network learn to *generate* new portrait
  - Generate samples from the distribution of “face” images
    - How do we even characterize this distribution?

What we have seen: VAE



- Generator is a decoder of a VAE

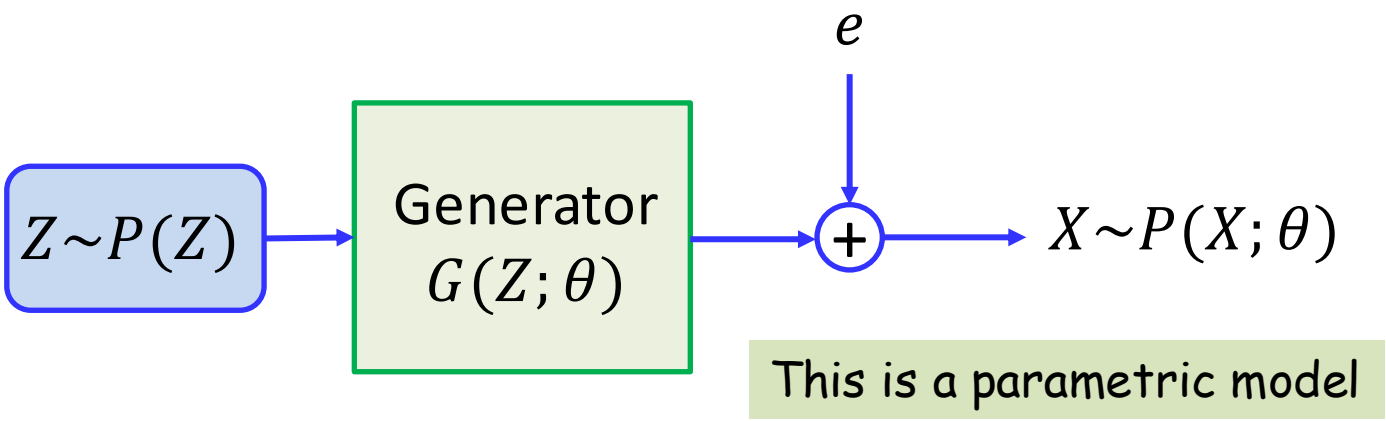
What we have seen: VAE



- Generator is a decoder of a VAE
- Trained by maximizing the *likelihood* of the data

$$\theta^* = \arg\max_{\theta} \quad \log P(X;\theta)$$

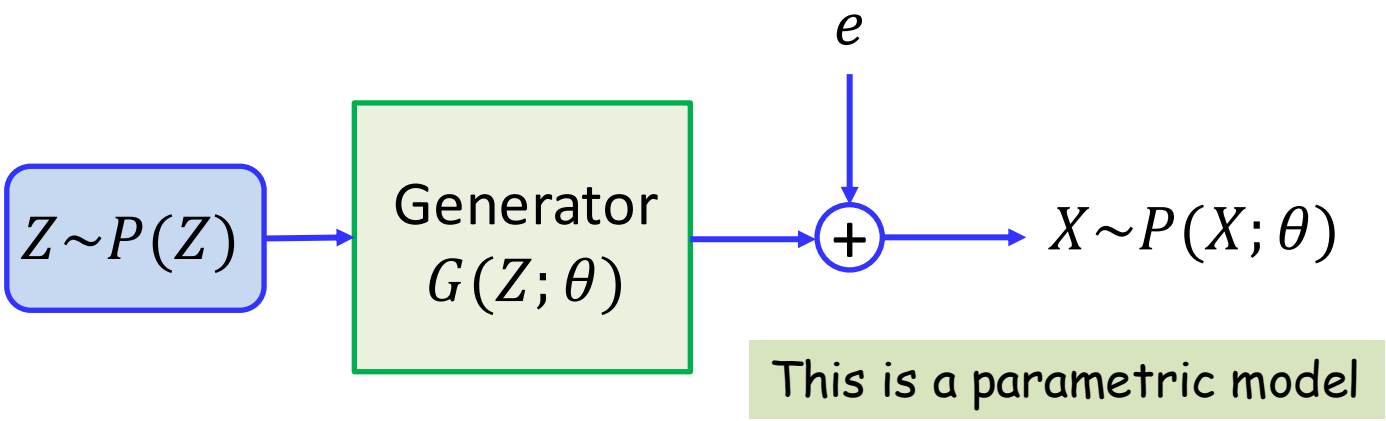
What we have seen: VAE



- Generator is a decoder of a VAE
- Trained by maximizing the *likelihood* of the data

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

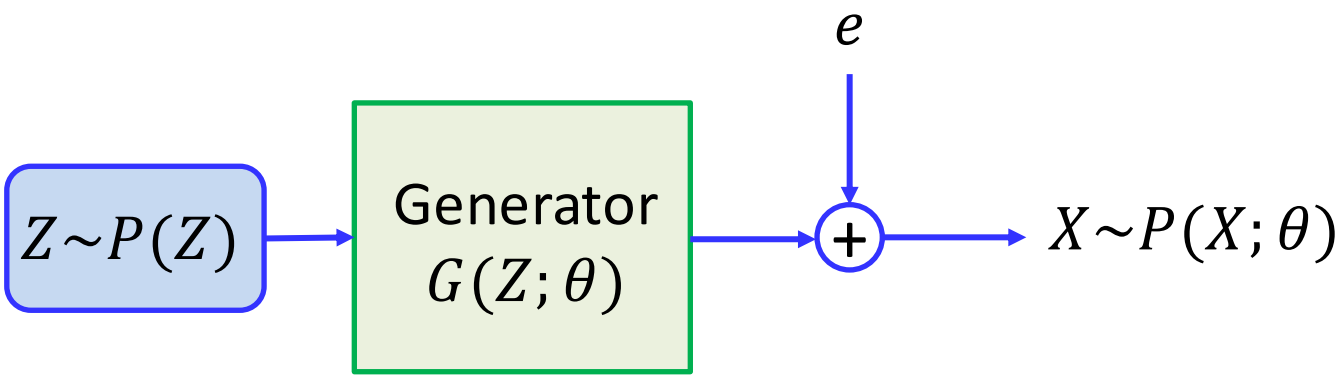
What we have seen: VAE



- Generator is a decoder of a VAE
- Trained by maximizing the *likelihood* of the data
  - Likelihood maximization does not actually relate to whether the output *actually* looks like a face

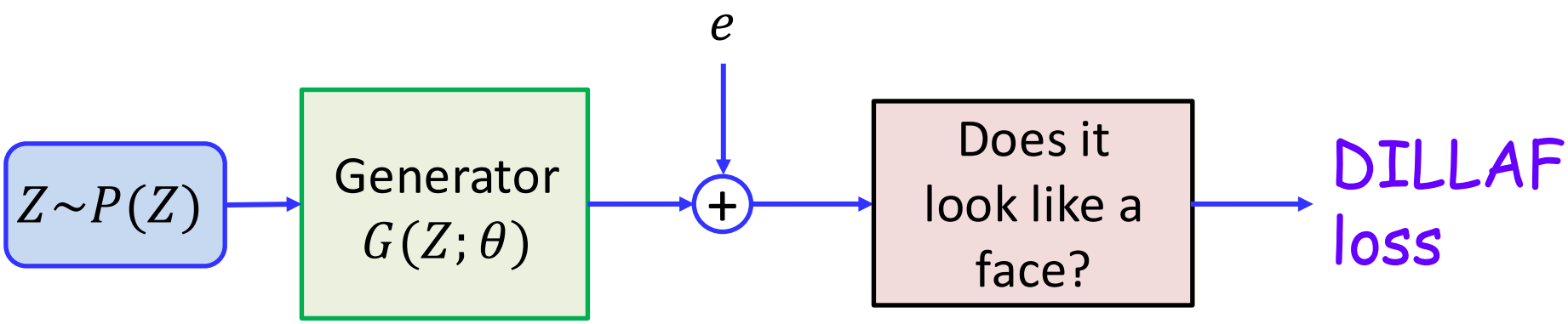
$$\theta^* = \arg\min_{\theta} \quad -\log P(X;\theta)$$

Training the VAE



- Generator is a decoder of a VAE
- Trained by maximizing the *likelihood* of the data
  - Likelihood maximization does not actually relate to whether the output *actually* looks like a face
- Can we make the training criterion more direct?

Replacing negative log likelihood with a more relevant loss



Poll 2 : @792 @793 @794

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

- Why would likelihood maximization not result in a model that produces more face-like 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

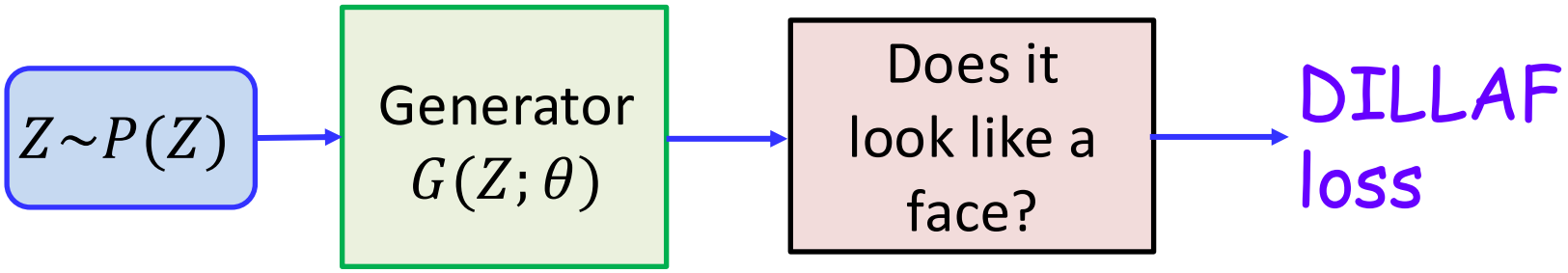
Poll 2

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

- Why would likelihood maximization not result in a model that produces more face-like 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

Replacing negative log likelihood with a more relevant loss



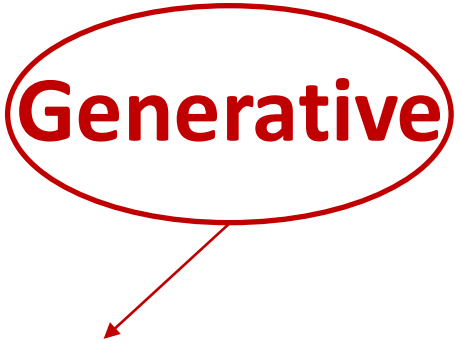
- But what is a good DILLAF loss?


What
are
GANs!

What are GANs

Generative Adversarial Networks

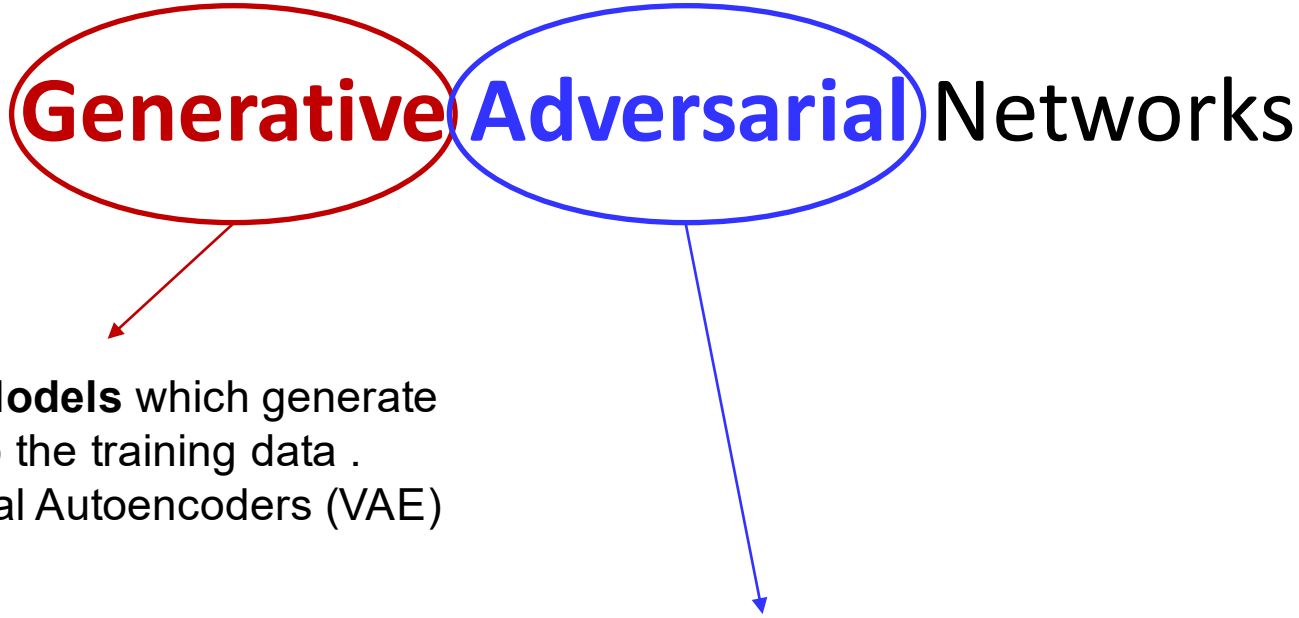
What are GANs



Adversarial Networks

Generative Models which generate data similar to the training data .
E.g. Variational Autoencoders (VAE)

What are GANs



Generative Models which generate data similar to the training data .
E.g. Variational Autoencoders (VAE)

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

What are GANs

Generative Models which generate data similar to the training data .
E.g. Variational Autoencoders (VAE)

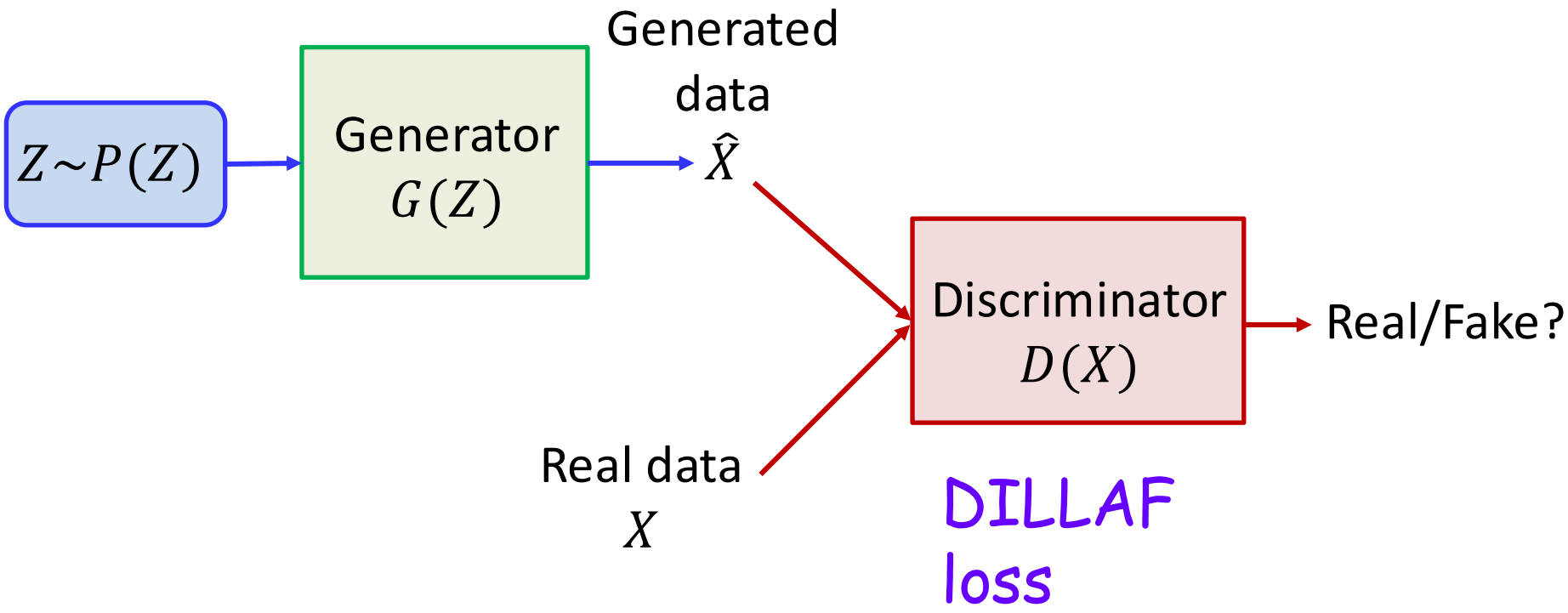
Neural Networks

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

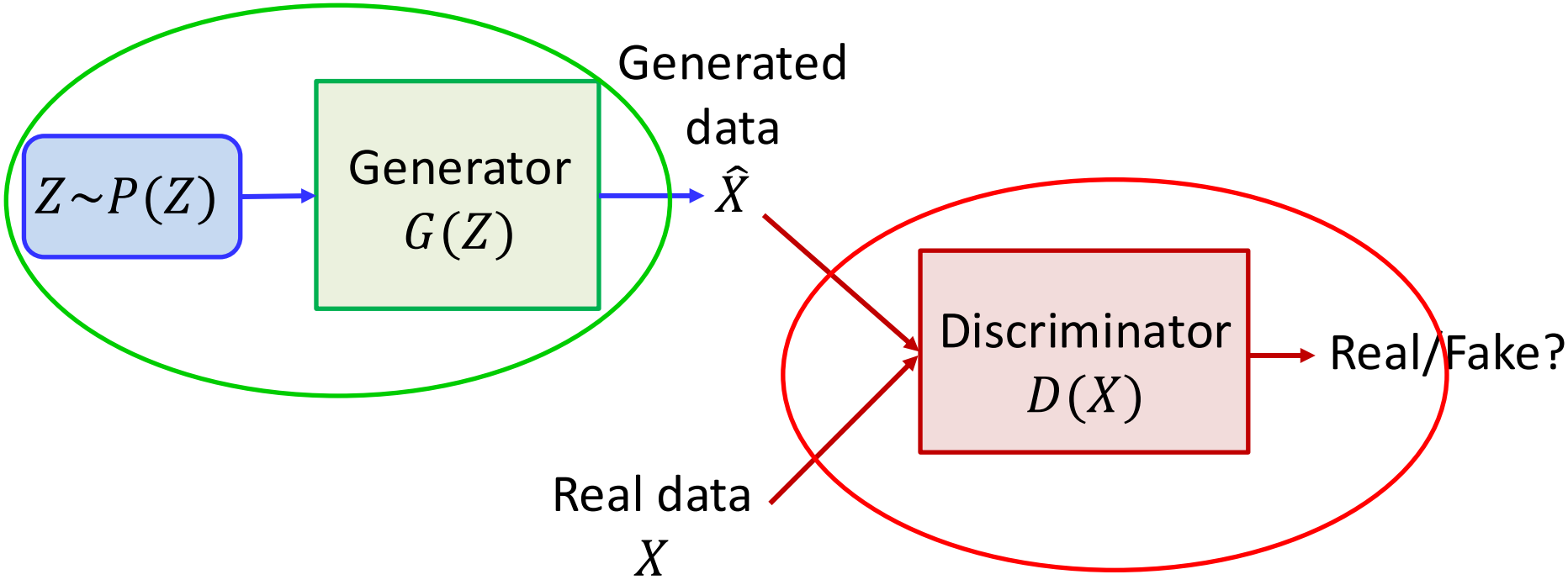
Generative Adversarial Networks

- 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!!

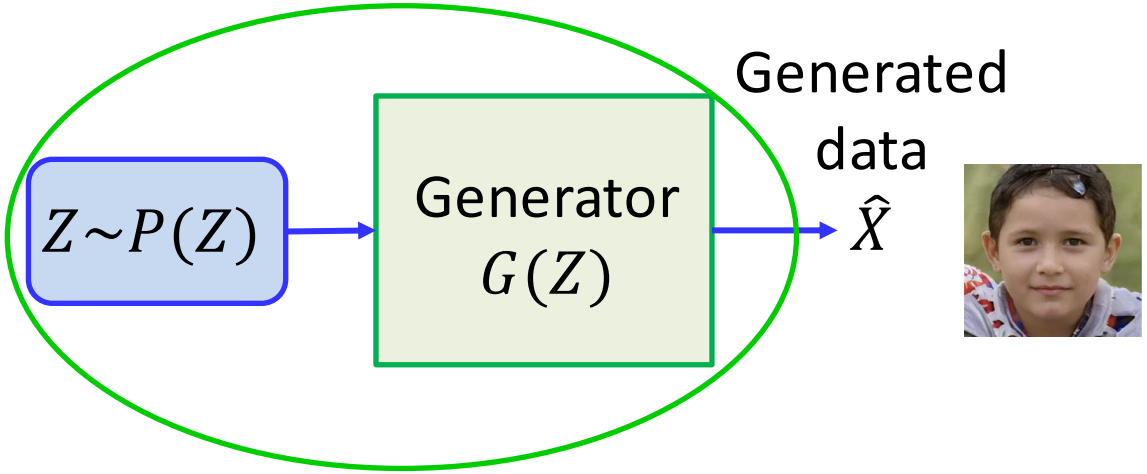
What are GANs?



What are GANs?



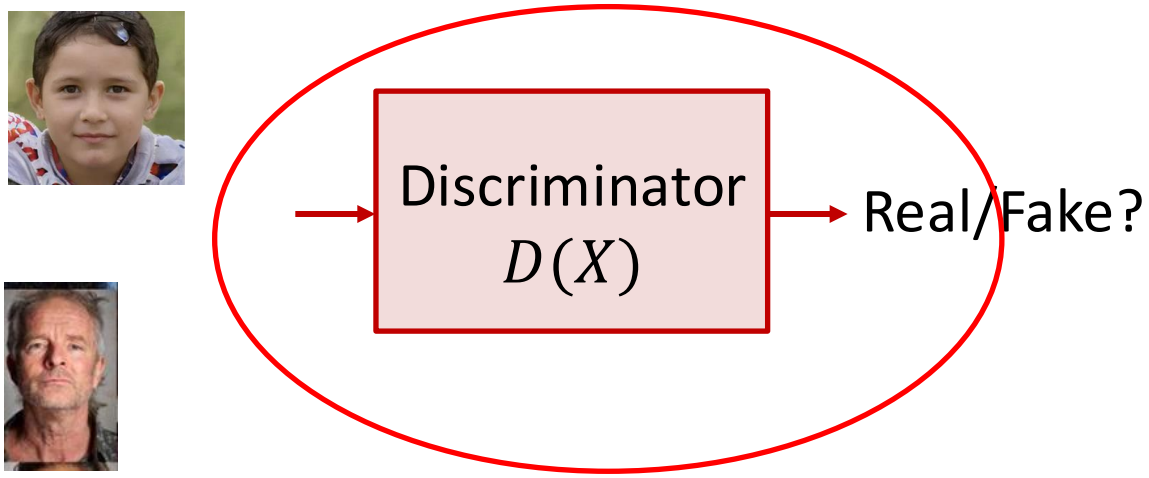
The Generator



- **The generator** produces realistic looking $X = G(z)$ from a latent vector Z
- Generator input Z can be sampled from a known prior, e.g. standard Gaussian

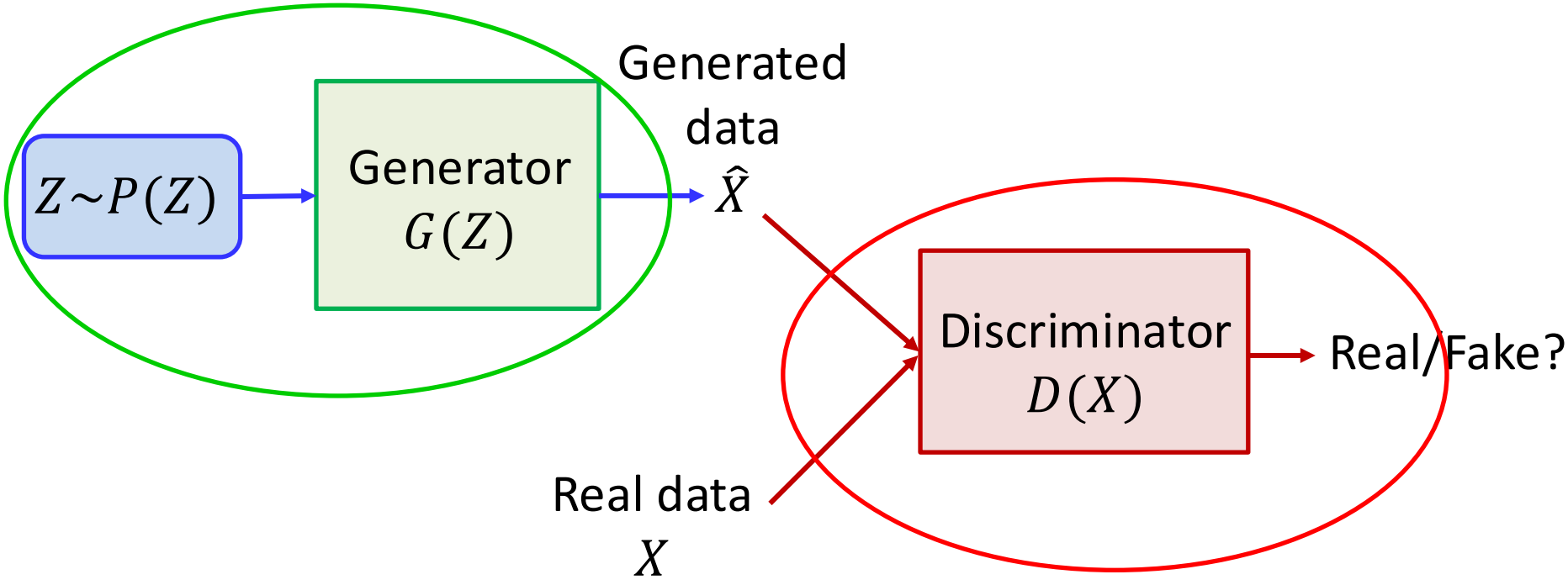
- **Goal**: generated distribution, $P_G(X)$ matches the true data distribution $P_X(X)$
  - $P_G(X)$ is the more "memorable" notation for $P_{\hat{X}}(X)$, the probability that a generated sample $\hat{X}$ takes the value X

The Discriminator



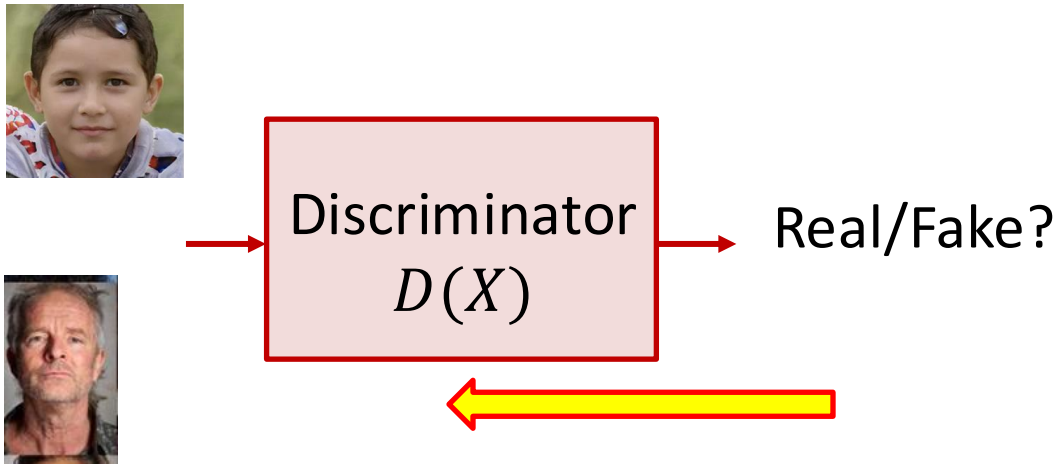
- Discriminator $D(X)$ is trained to tell the difference between real and generated (fake) data
  - Specifically, data produced by the generator
  - If a perfect discriminator is fooled, the generated data cannot be distinguished from real data

Training a GAN



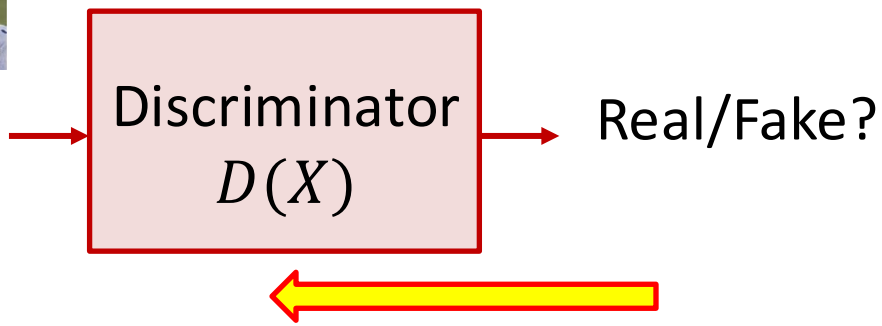
- Both, the generator and discriminator must be trained

Training the discriminator



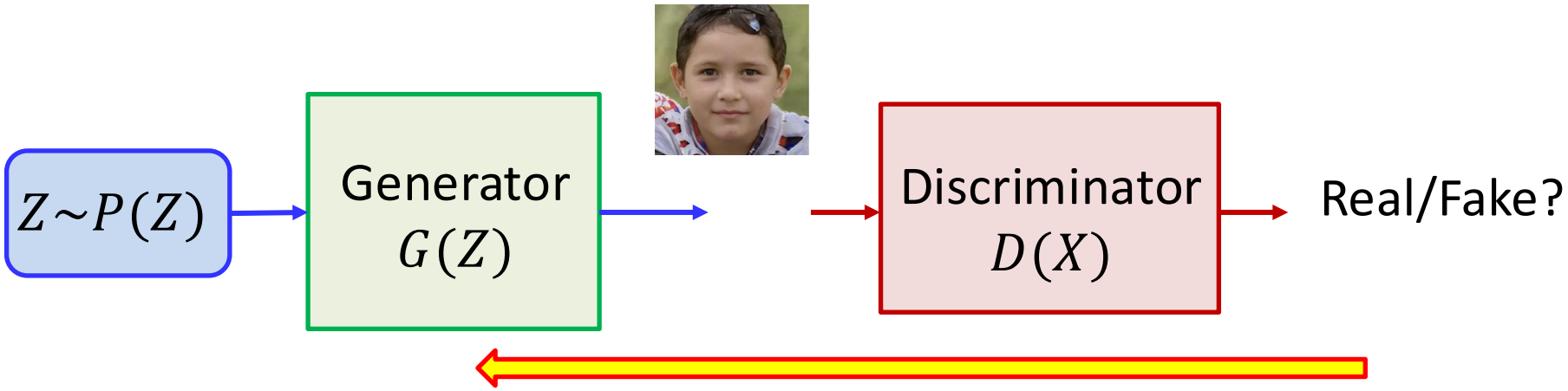
- **Training the discriminator:**
  - The discriminator is provided training examples of real and synthetic faces
  - The discriminator is trained to minimize its classification loss
    - Minimize error between actual and predicted labels
  - Discriminator parameters are trained such that
    - $D(X) = 1$ for real faces
    - $D(X) = 0$ for synthetic faces (i.e $1 - D(X) = 1$)

Training the discriminator



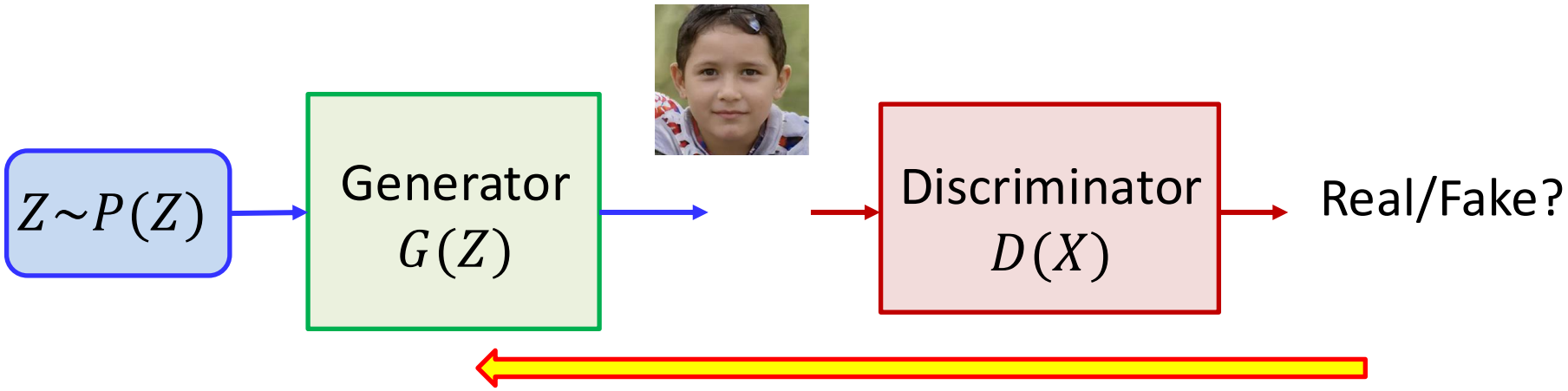
- **Training the discriminator:**
  - The discriminator is provided training examples of real and synthetic faces
  - The discriminator is trained to minimize its classification loss
    - Minimize error between actual and predicted labels
  - Discriminator parameters are trained such that
    - Maximize $\log(D(X))$ for real faces
    - Maximize $\log(1 - D(X))$ for synthetic faces

Training the generator



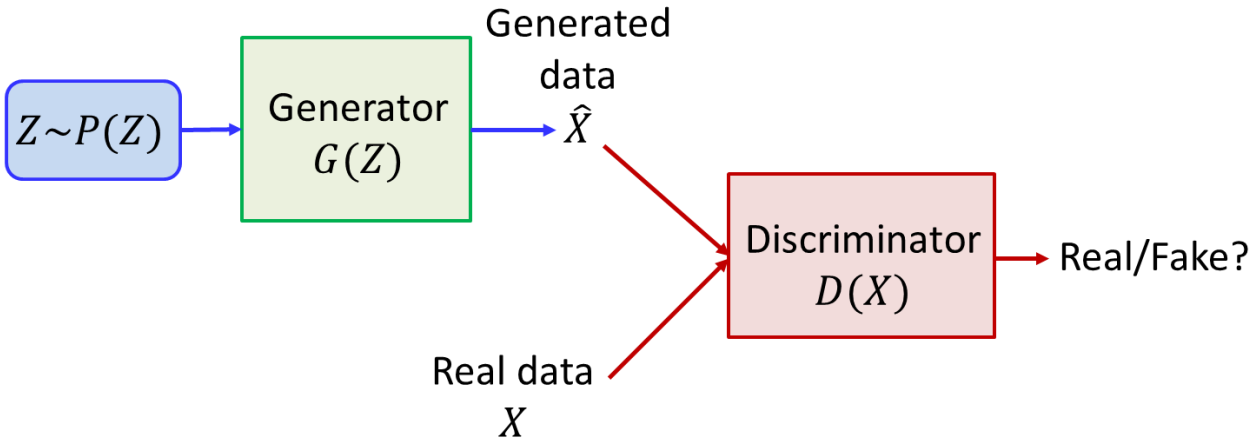
- **Training the generator:**
  - The discriminator's loss is backpropagated to the generator
  - The generator is trained to *maximize* the discriminator loss
    - It is trained to "fool" the discriminator
  - Generator parameters are trained such that
    - $D(G(Z)) = 1$ (i.e. $1 - D(G(Z)) = 0$)

Training the generator



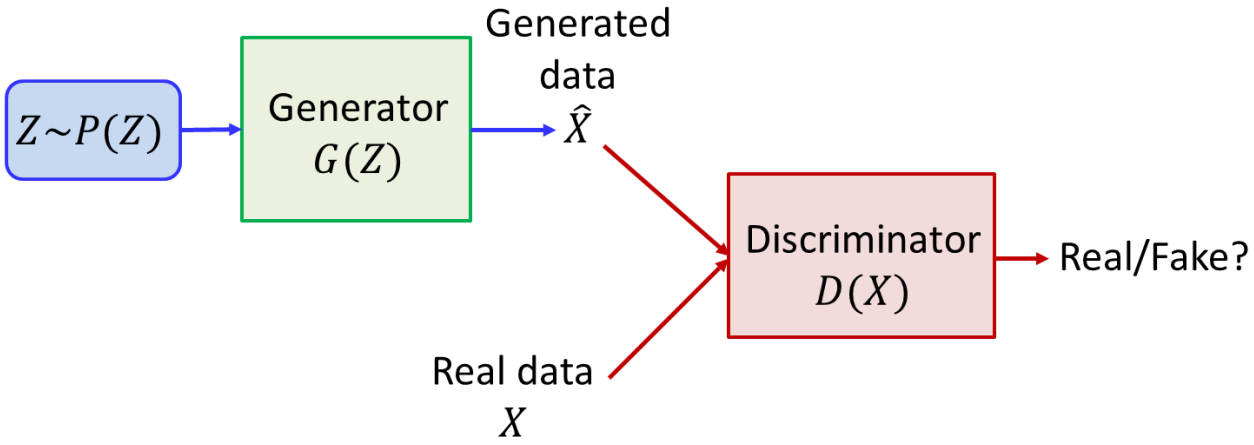
- **Training the generator:**
  - The discriminator's loss is backpropagated to the generator
  - The generator is trained to *maximize* the discriminator loss
    - It is trained to "fool" the discriminator
  - Generator parameters are trained such that
    - Minimize $\log\left(1 - D\left(G(Z)\right)\right)$

The GAN formulation



- Discriminator:
  - For real data X, Maximize $\log\left(D(X)\right)$
  - For synthetic data Maximize $\log\left(1 - D\left(\hat{X}\right)\right)$
- Generator
  - Minimize $\log\left(1 - D\left(\hat{X}\right)\right)$

The GAN formulation

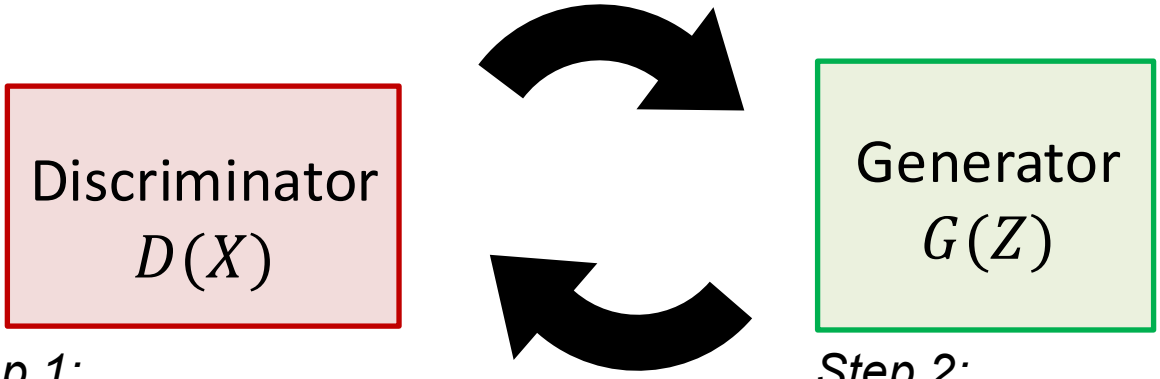


- The original GAN formulation is the following min-max optimization

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

- 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 @795 @796

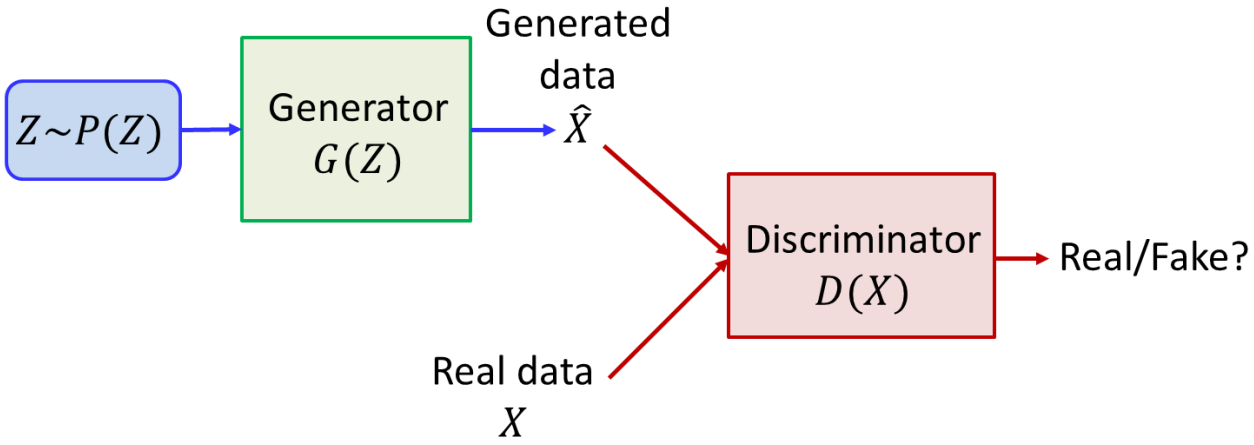
- When training a GAN, which component must you train first
  - The discriminator
  - The generator
- Which component is updated more frequently
  - The discriminator
  - The generator

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

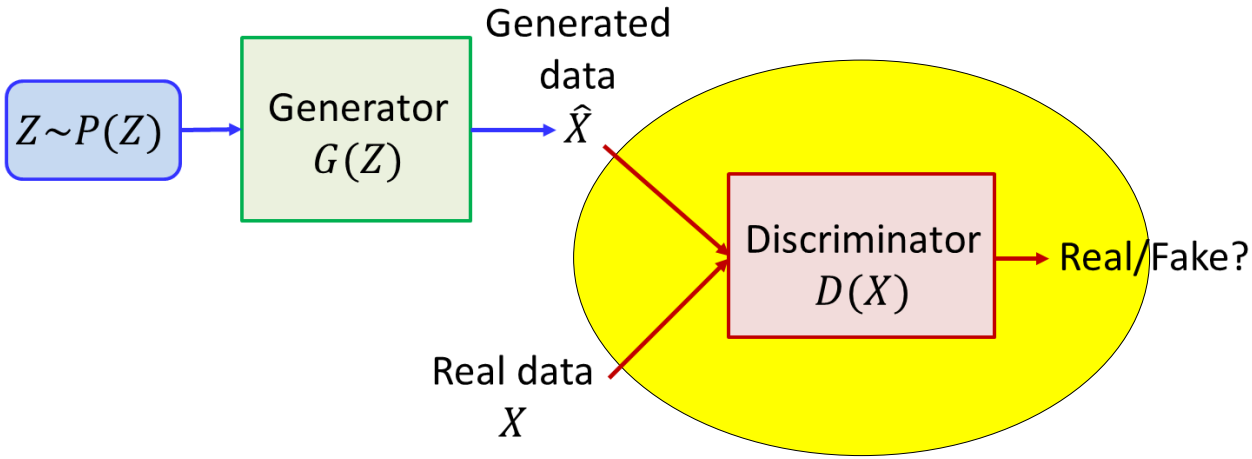
The discriminator is the (DILLAF) loss. Training the loss is more important, since the loss guides the training!

The GAN formulation



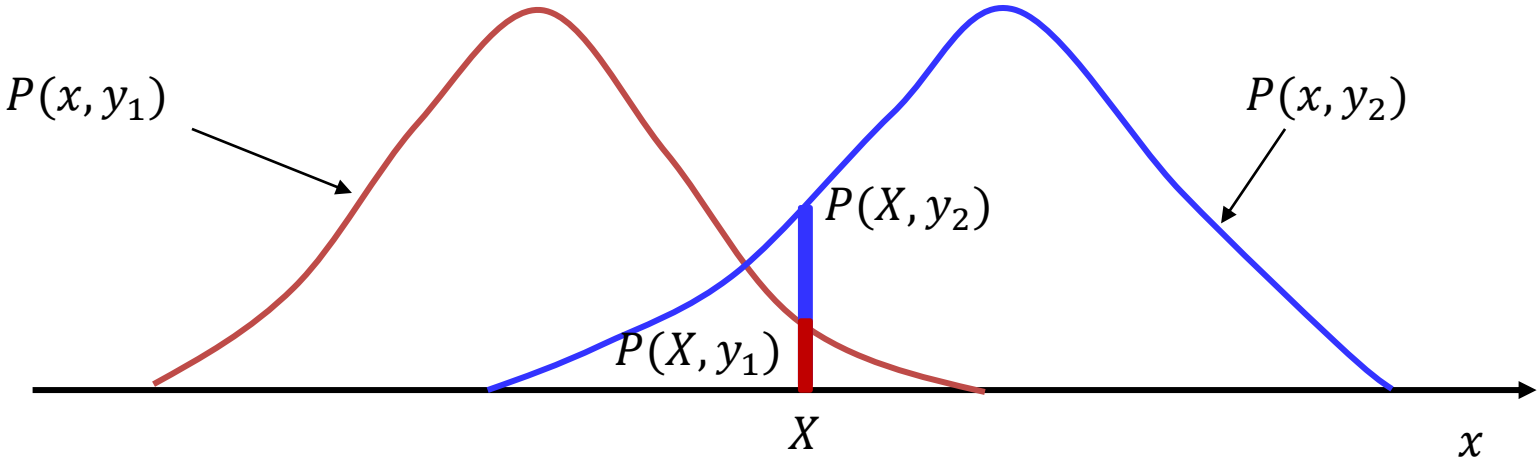
- So how does this behave when each component is optimized…

The GAN formulation



- So how does this behave when each component is optimized…
  - The optimal discriminator:

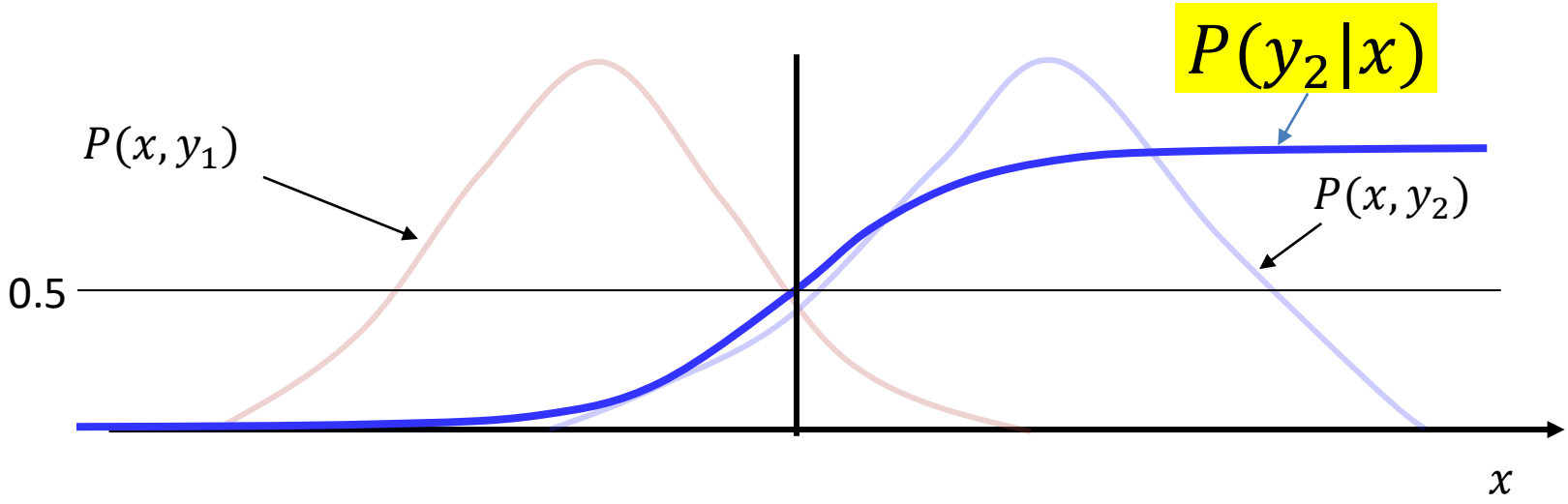
The perfect discriminator:
Consider a binary classification problem



- The a posteriori probability of the classes for any instance $x = X$ is

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

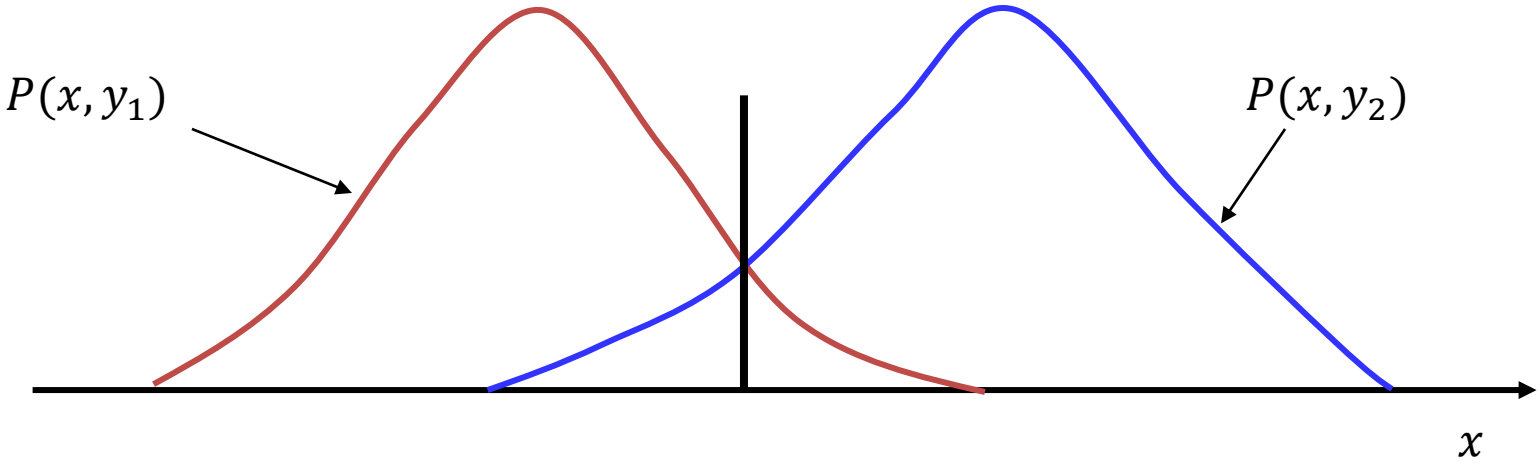
The perfect discriminator:
Consider a binary classification problem



- 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:
Consider a binary classification problem

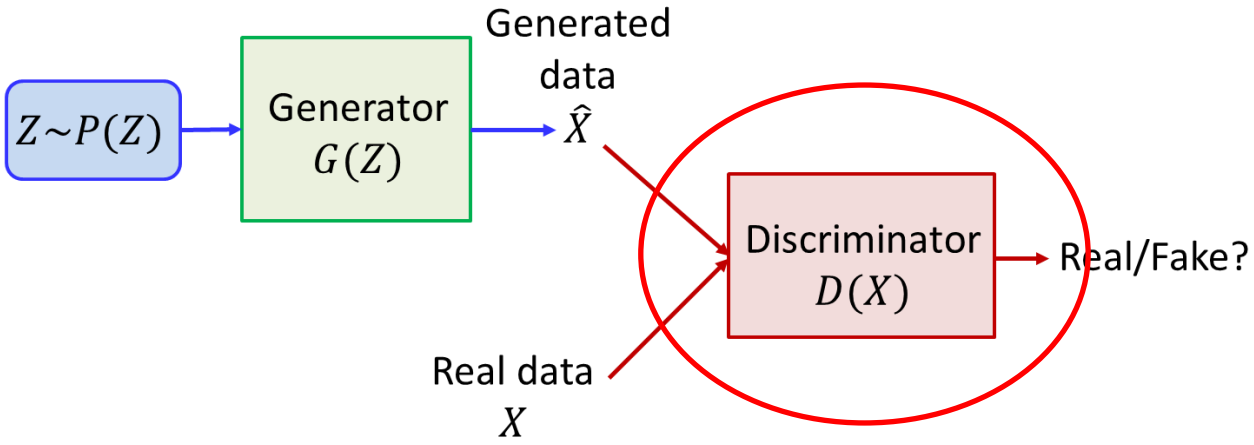


- The a posteriori probability of the classes for any instance $x = X$ is

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

- The perfect decision boundary is where $P(y_1|X) = P(y_2|X)$
  - The perfect discriminator will compute $P(y_i|X)$ for each class
  - It will assign any X to the class with the higher $P(y_i|X)$

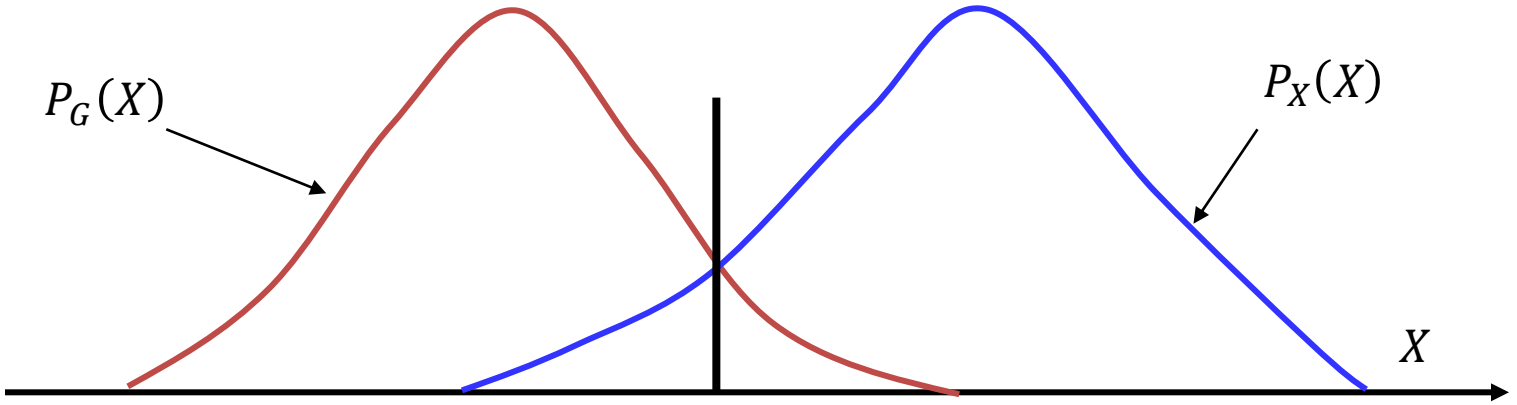
The optimal discriminator



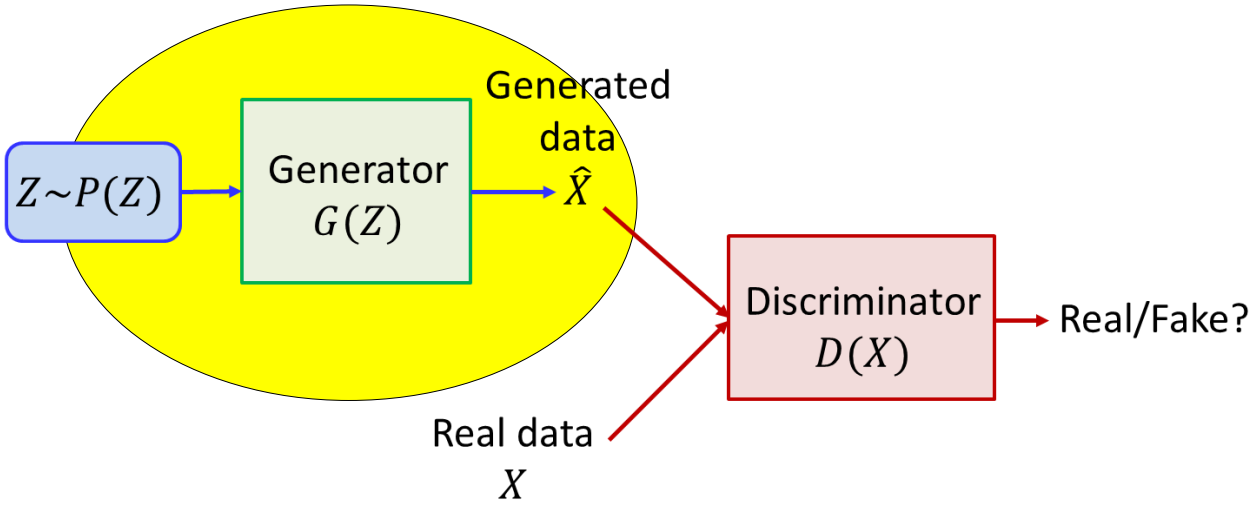
- The **optimal discriminator** would be a Bayesian classifier

$$D(X) = \frac{P_X(X)}{P_X(X) + P_G(X)}$$

  - Assuming uniform prior

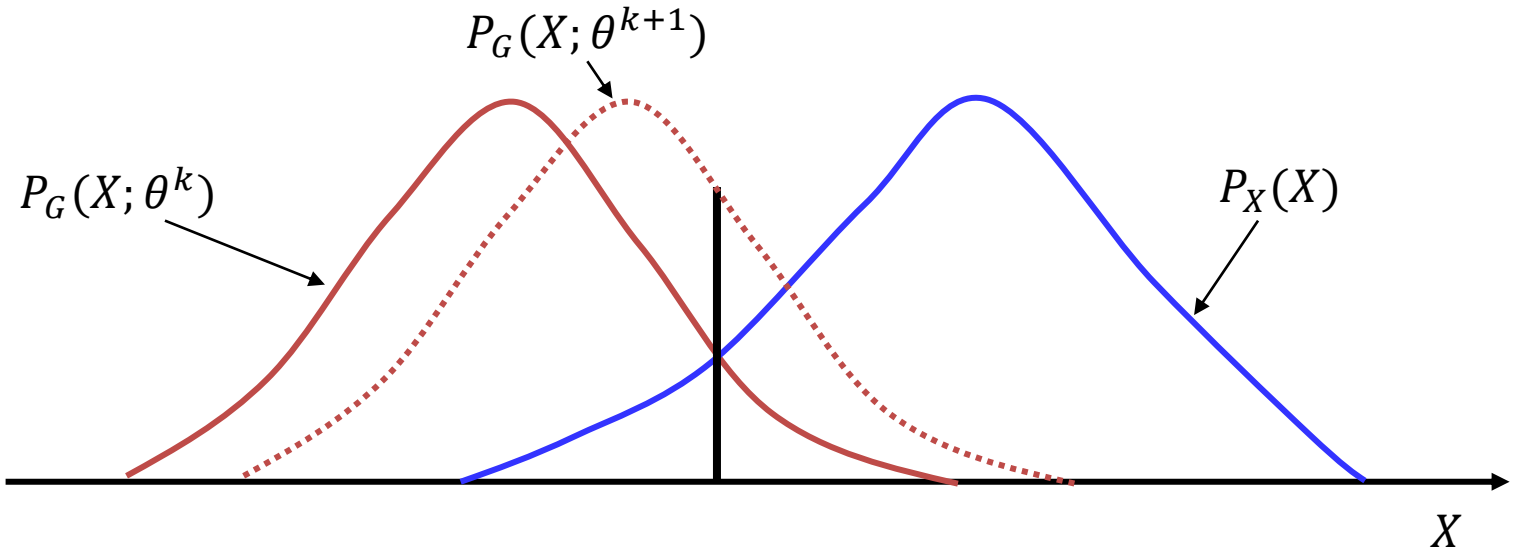


The GAN formulation



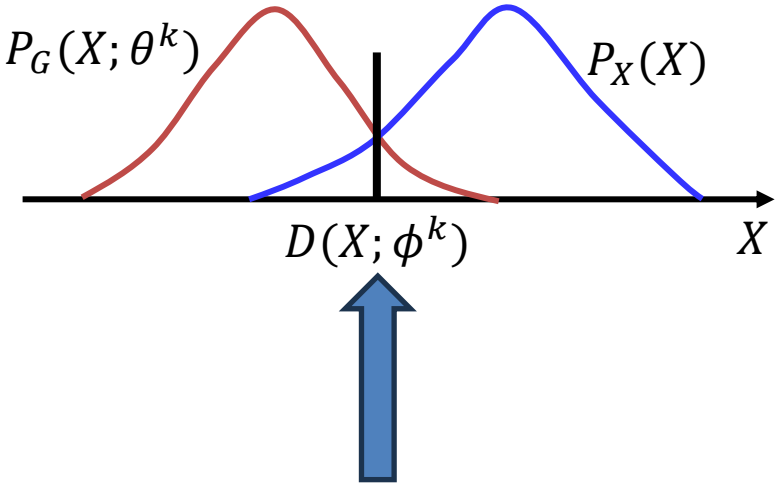
- So how does this behave when each component is optimized…

Updating the Generator: Fooling the perfect discriminator



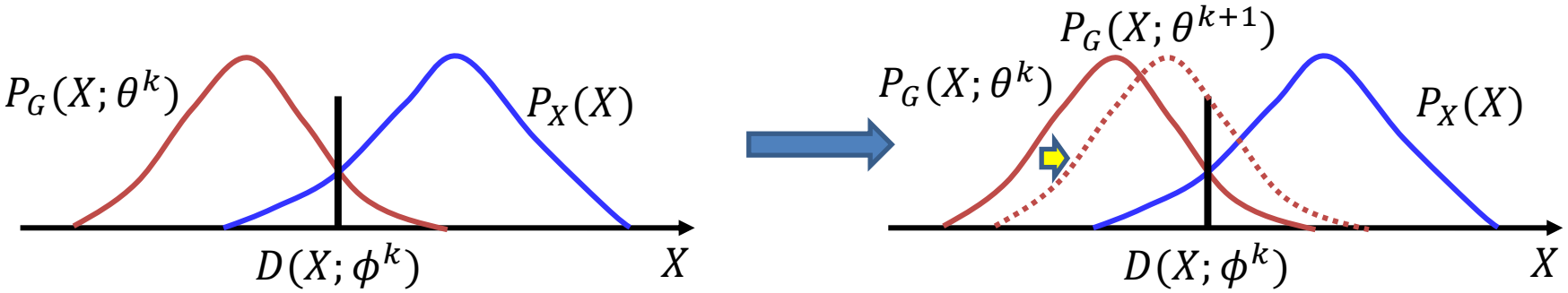
- Relearn generator parameters so that the new distribution of generated data "fools" the discriminator
  - By moving it into the region assigned to the other class by the (perfect) discriminator

The iterated learning



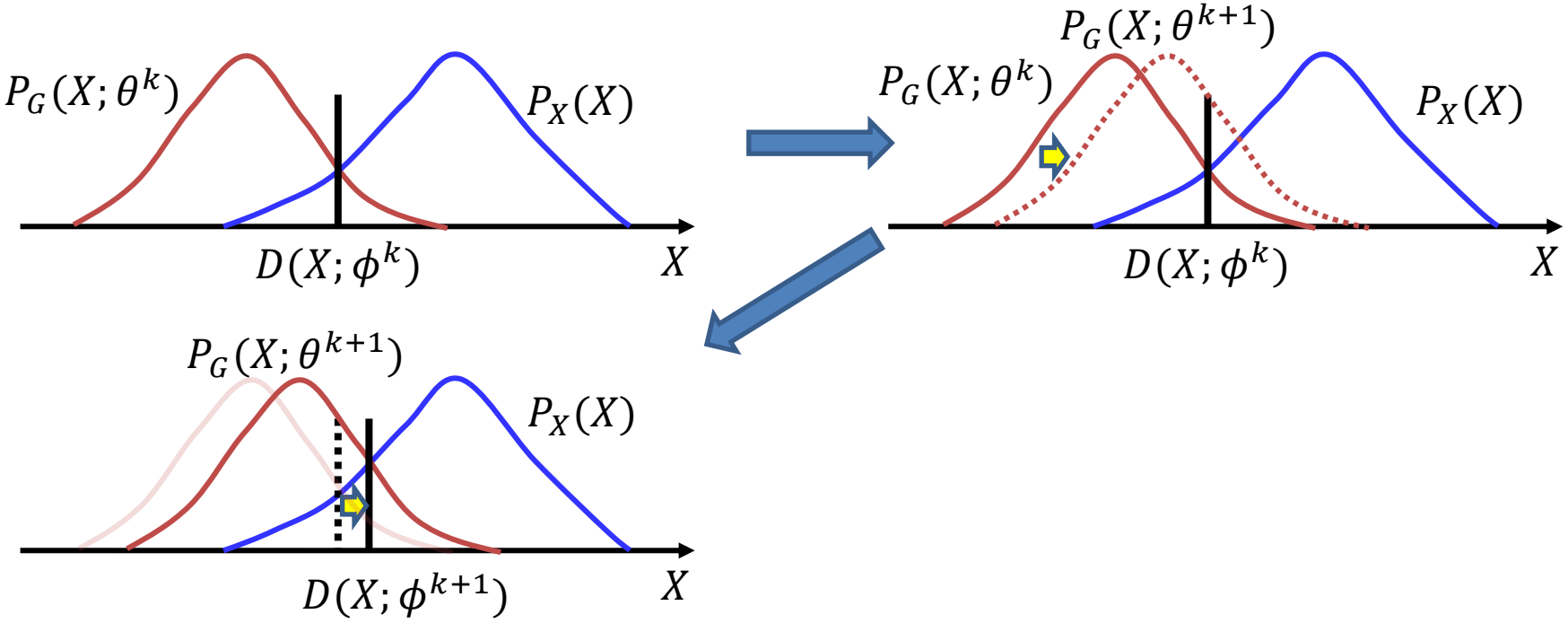
- Discriminator learns perfect boundary

The iterated learning



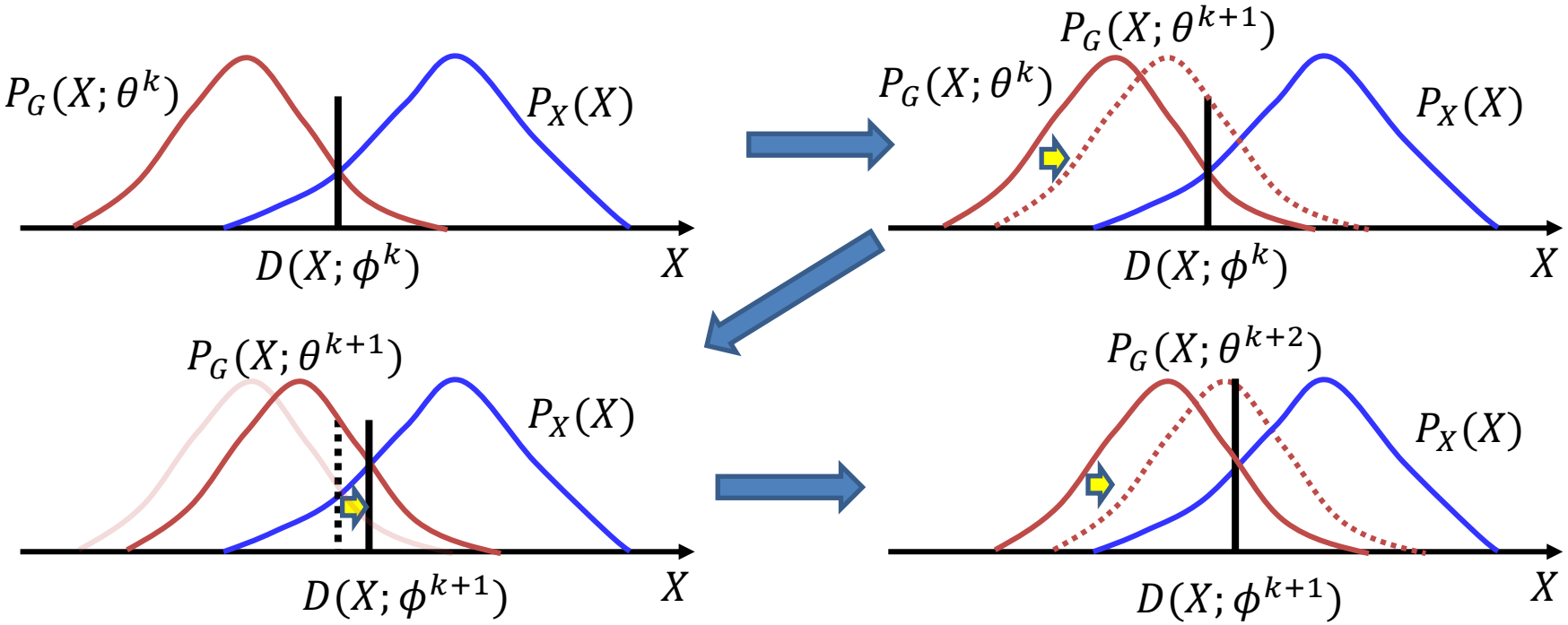
- Discriminator learns perfect boundary
- Generator moves its distribution past the boundary “into” the real distribution

The iterated learning



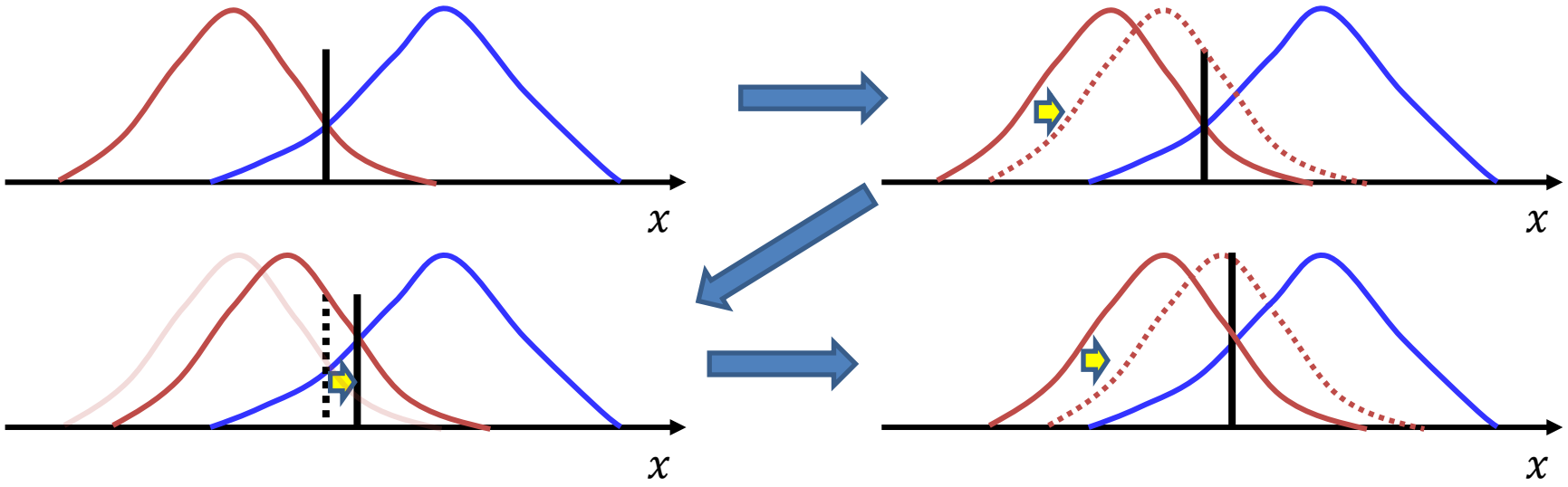
- Discriminator learns perfect boundary
- Generator moves its distribution past the boundary “into” the real distribution
- Discriminator relearns new “perfect” boundary

The iterated learning

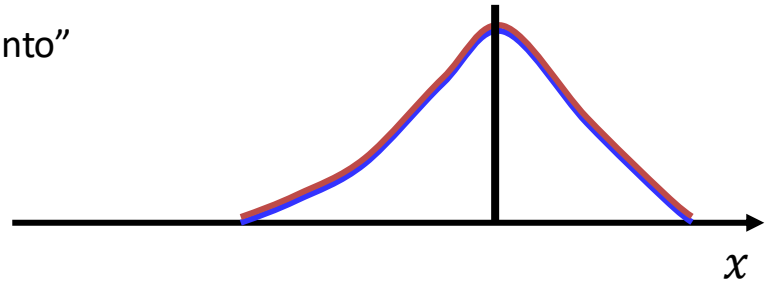


- Discriminator learns perfect boundary
- Generator moves its distribution past the boundary "into" the real distribution
- Discriminator relearns new "perfect" boundary
- Generator shifts distribution past new boundary
- …

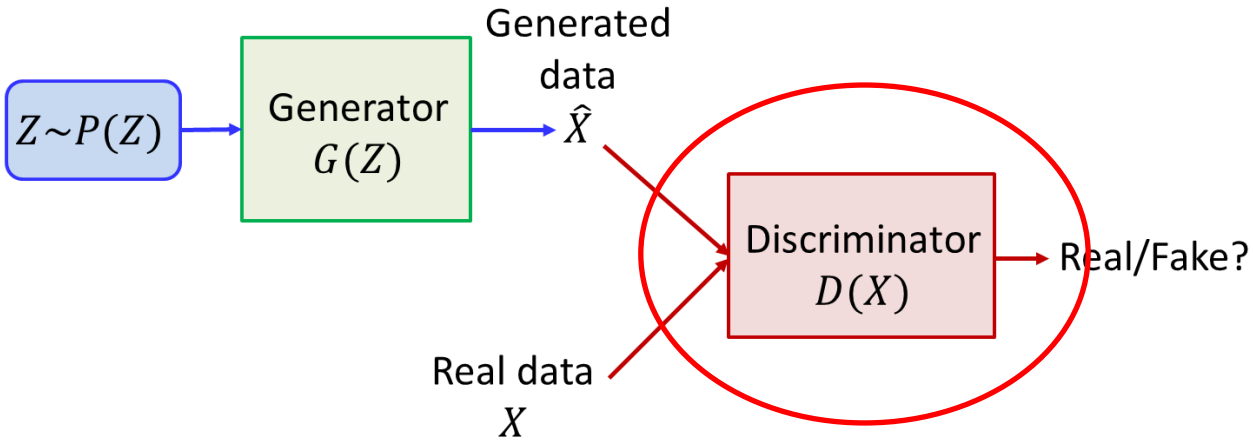
The iterated learning



- 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



Analysis of optimal behavior: The optimal discriminator

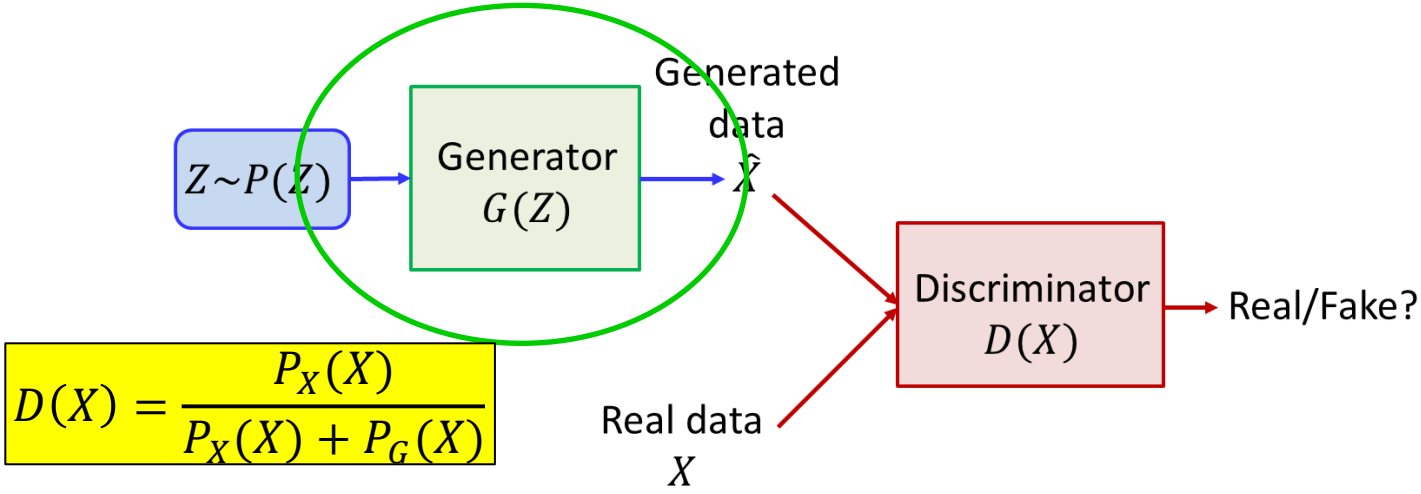


- The **optimal discriminator** would be a Bayesian classifier

$$D(X) = \frac{P_X(X)}{P_X(X) + P_G(X)}$$

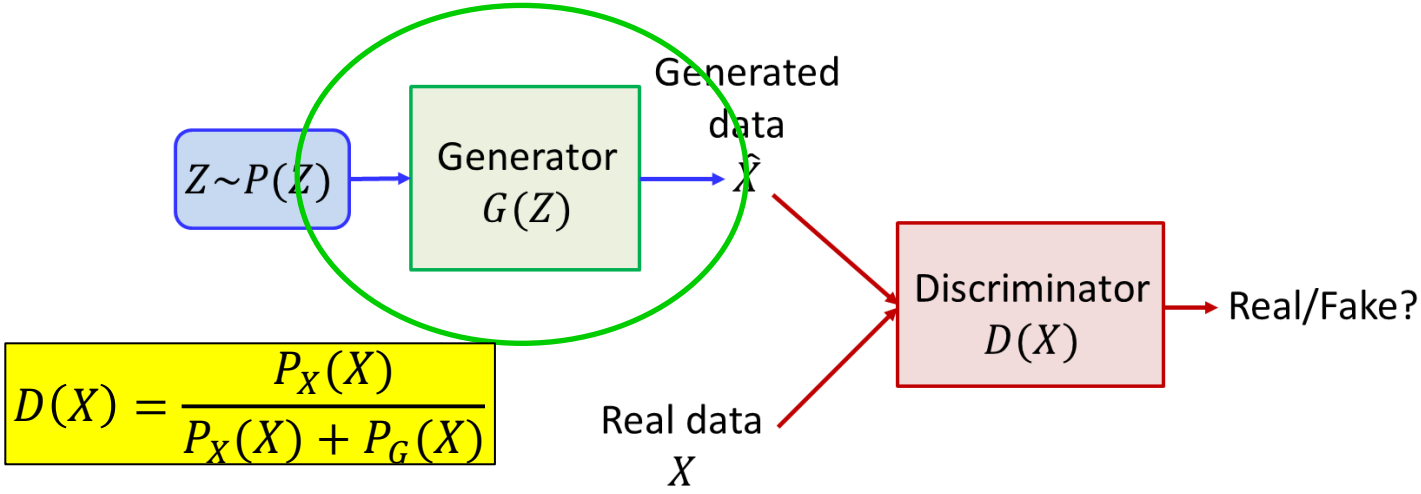
  - Assuming uniform prior

Analysis of optimal behavior:
The optimal generator



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

Analysis of optimal behavior: The optimal generator

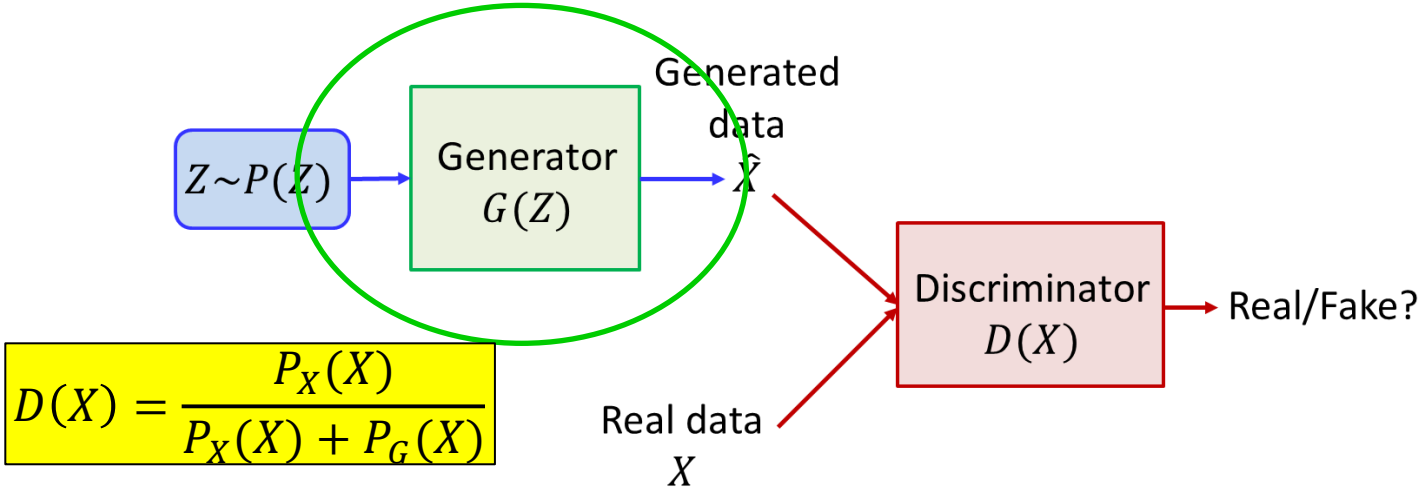


$$\min_G \max_D E_X \log D(X) + E_Z \log(1 - D(G(Z)))$$

- **With a perfect discriminator**:

$$L = E_{X \sim P_X(X)} \log D(X) + E_{X \sim P_G(X)} \log(1 - D(X))$$

Analysis of optimal behavior: The optimal generator



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

- **With a perfect discriminator**:

$$L = E_{X \sim P_X(X)} \log D(X) + E_{X \sim P_G(X)} \log(1 - D(X))$$
$$= E_{X \sim P_X(X)} \log \left(\frac{P_X(X)}{P_X(X) + P_G(X)} \right) + E_{X \sim P_G(X)} \log \left(\frac{P_G(X)}{P_X(X) + P_G(X)} \right)$$

The KL Divergence

$$KL\,(P,Q) = \sum_X P(X)\log(P(X)/Q(X))$$

- What are the problems with this?

$$KL\,(Q,P) = \sum_X Q(X)\log(Q(X)/P(X))$$

- What are the problems with this?

The KL Divergence

$$KL\,(P,Q) = \sum_X P(X)\log(P(X)/Q(X))$$

- What are the problems with this?

$$KL\,(Q,P) = \sum_X Q(X)\log(Q(X)/P(X))$$

- What are the problems with this?

KL is not symmetric, and runs into issues if either P or Q become 0 (whichever is entirely inside the log)

The Jensen Shannon Divergence

$$JSD\,(P,Q) = 0.5\,KL(P, 0.5(P+Q)) + 0.5KL(Q, 0.5(P+Q))$$

- If the term inside the log is 0, both P and Q are 0
  - 0 log 0 = 0, so there are no problems
- Also, this is symmetric:

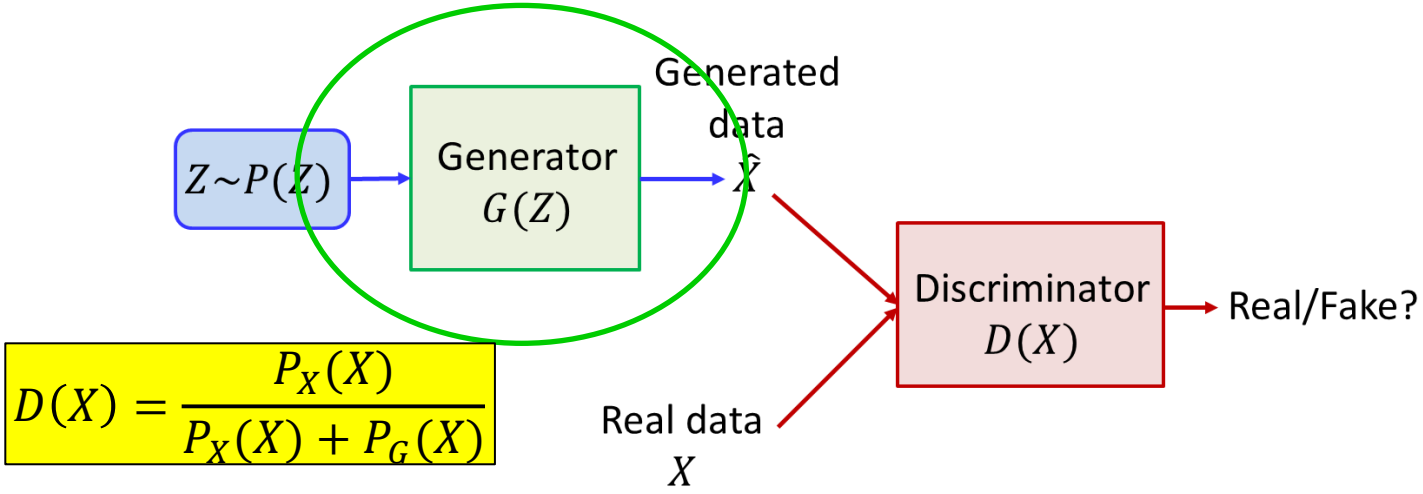
$$JSD\,(P,Q) = JSD\,(Q,P)$$

The Jensen Shannon Divergence

$$JSD\ (P, Q)$$
$$= \ 0.5\ KL(P, 0.5(P + Q))\ +\ 0.5KL(Q, 0.5(P + Q))$$

- A symmetric variant of KL that does not exaggerate instances to which one of the distributions assigns 0 probability
  - $KL(P, Q) = \ \sum_X P(X) \log(P(X)/Q(X))$ blows up the contributions of X with $Q(X)\ =\ 0$

Analysis of optimal behavior:
The optimal generator



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

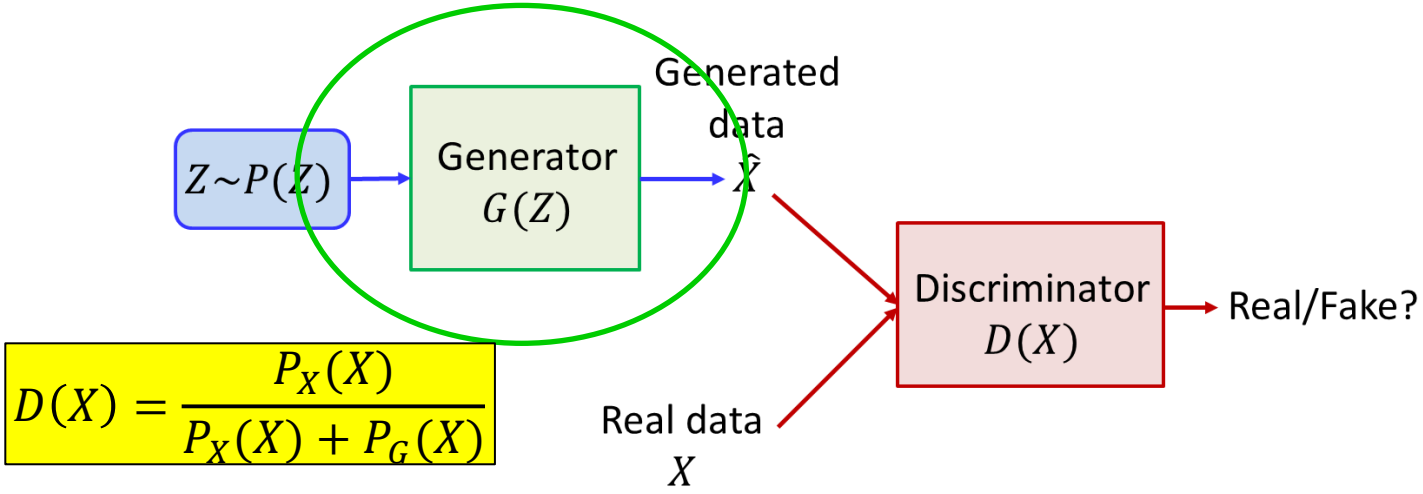
- **With a perfect discriminator**:

$$L = E_{X \sim P_X(X)} \log D(X) + E_{X \sim P_G(X)} \log(1 - D(X))$$
$$= E_{X \sim P_X(X)} \log\left(\frac{P_X(X)}{P_X(X) + P_G(X)}\right) + E_{X \sim P_G(X)} \log\left(\frac{P_G(X)}{P_X(X) + P_G(X)}\right)$$

- This is just the Jensen-Shannon divergence between $P_X(X)$ and $P_G(X)$ to within a scaling factor and a constant

$$L = 2JSD\big(P_X(X), P_D(X)\big) - \log 4$$

Analysis of optimal behavior: The optimal generator

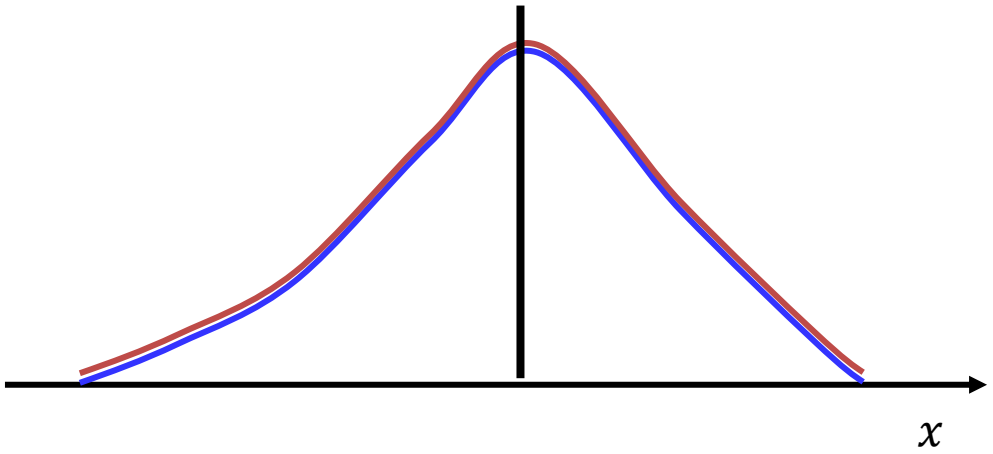


- The optimal generator:

$$\min_G 2JSD\big(P_X(X), P_G(X)\big) - \log 4$$

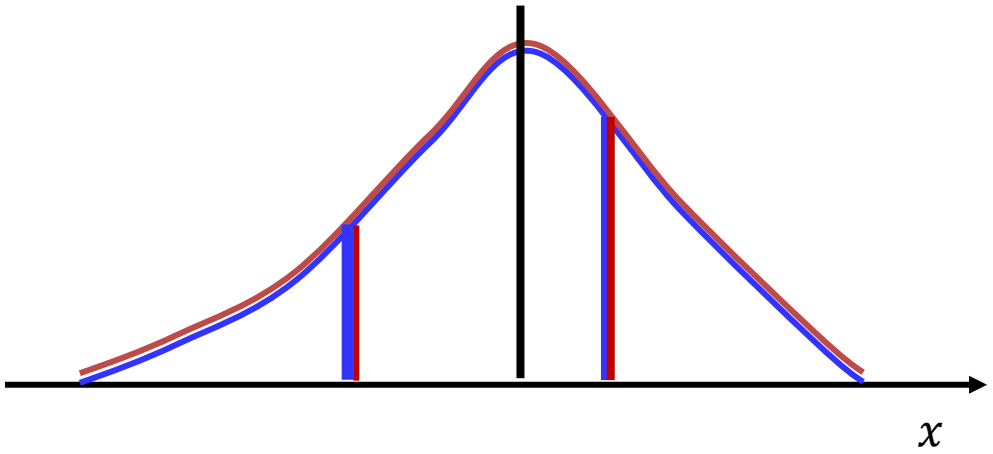
- The optimal generator minimizes the Jensen Shannon divergence between the distributions of the actual and synthetic data!
  - Tries to make the two distributions maximally similar

The optimal generator with the optimal discriminator



- The generator of the fully optimized GAN will generate $P_G(X) = P_X(X)$, i.e. the distribution of the generated data will be identical to that of the original data

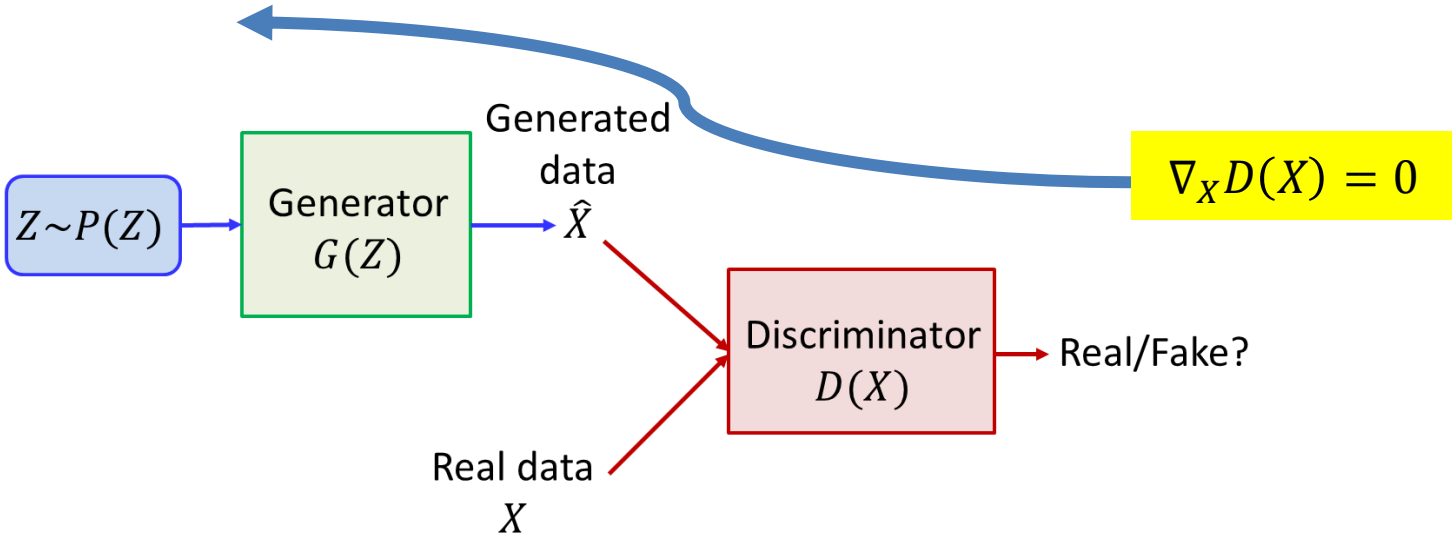
The optimal generator with the optimal discriminator



- The generator of the fully optimized GAN will generate $P_G(X) = P_X(X)$, i.e. the distribution of the generated data will be identical to that of the original data

- At any X, $P_G(X) = P_X(X)$
  - I.e. $D(X) = \frac{P_X(X)}{P_X(X)+P_G(X)} = 0.5$
  - The derivative of $D(X)$ w.r.t X = 0

The optimal generator with the optimal discriminator



- $\nabla_X D(X) = 0$
- All derivatives going backward are 0
- There will be no further updates

Min-Max Stationary Point

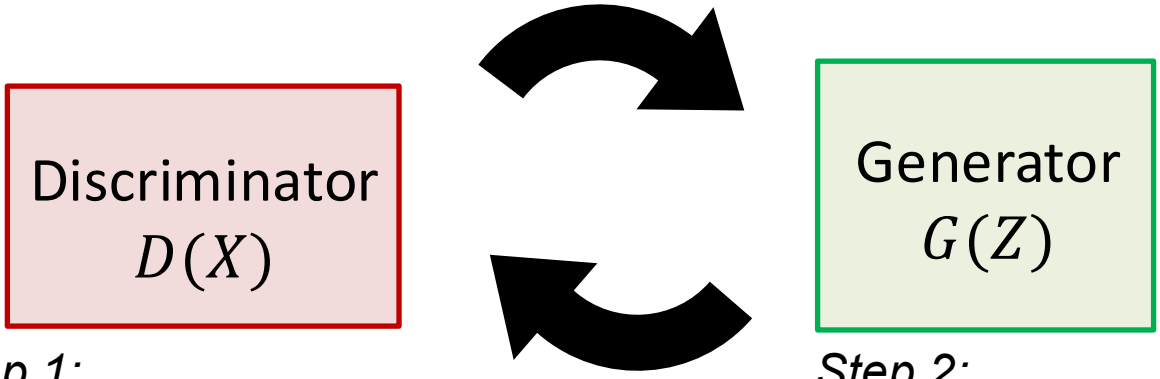
- There exists a stationary point:
  - If the generated data exactly matches the real data, the discriminator outputs 0.5 for all inputs
  - If discriminator outputs 0.5, the gradients for the generator is flat, so generator does not learn
  - Unfortunately, this is also true of a random discriminator

- Stationary points need not be stable (depends on the exact GANs formulation and other factors)
  - Generator may overshoot some values or oscillate around the optimum
  - A discriminator with unlimited capacity can still assign an arbitrarily large distance to 2 similar distributions

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

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

Features and Challenges

- GANs can produce clear crisp results for many problems
- But they also have stability issues and are hard to train
  - Problems such as “mode collapse” are frequent
    - Producing outputs with very low variability

Poll 4 : @797

- 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**

VAEs vs GANs

VAEs

- Minimizing the KL divergence between distributions of synthetic and true data
- Uses an encoder to predict latent distributions to optimize generator
- More complex formulation
- Simpler optimization. Trains faster and more reliably
- Results are blurry

GANs

- Minimizing the Jenson-Shannon divergence between distributions of synthetic and true data
- Use a discriminator to optimize generator
- Simpler formulation
- Noisy and difficult optimization
- Sharper results

Original paper (GAN, 2014)

Output of original GAN paper, 2014 [GPM+14]

GANs with time

- Better quality
- High Resolution

https://twitter.com/goodfellow_ian/status/1084973596236144640?lang=en

StarGAN(2018)

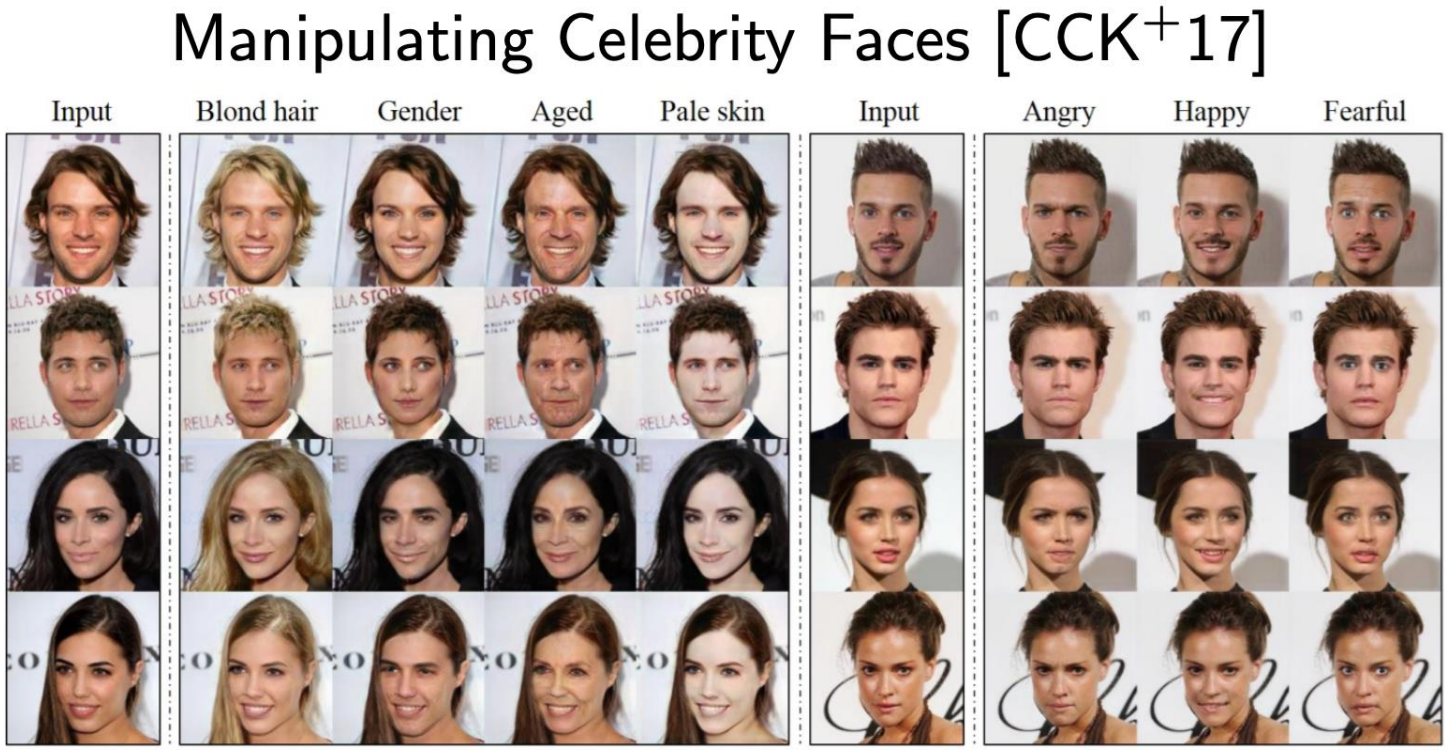


Figure 1. Multi-domain image-to-image translation results on the CelebA dataset via transferring knowledge learned from the RaFD dataset. The first and sixth columns show input images while the remaining columns are images generated by StarGAN. Note that the images are generated by a single generator network, and facial expression labels such as angry, happy, and fearful are from RaFD, not CelebA.

Progressive growing of GANs (2018)

Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

High fidelity natural images (2019)

Generating High-Quality Images [BDS18]

Moving on (slides by Hao Chen)

- Addressing many of the shortcomings of GANs
- Different types of GANs
- GAN applications

Problems of Vanilla GANs

- Vanishing gradients: Generator receives no meaningful gradient for learning
- Mode collapse: the generator distribution collapses to a small set of samples

Problems of Vanilla GANs

- Vanishing gradients: Generator receives no meaningful gradient for learning

Problems of Vanilla GANs

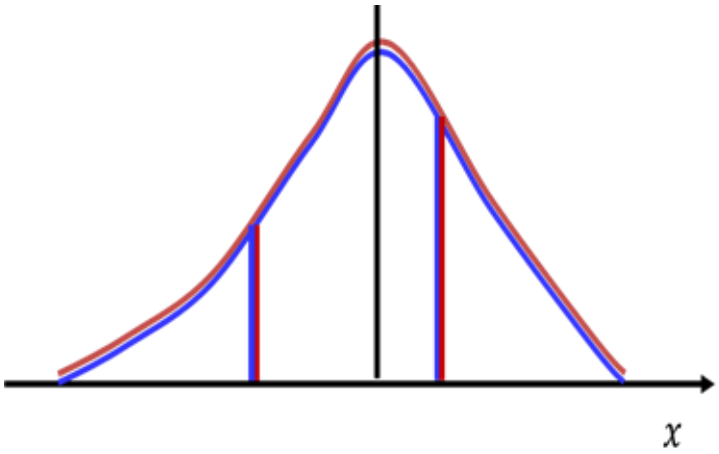
- Vanishing gradients: Generator receives no meaningful gradient for learning

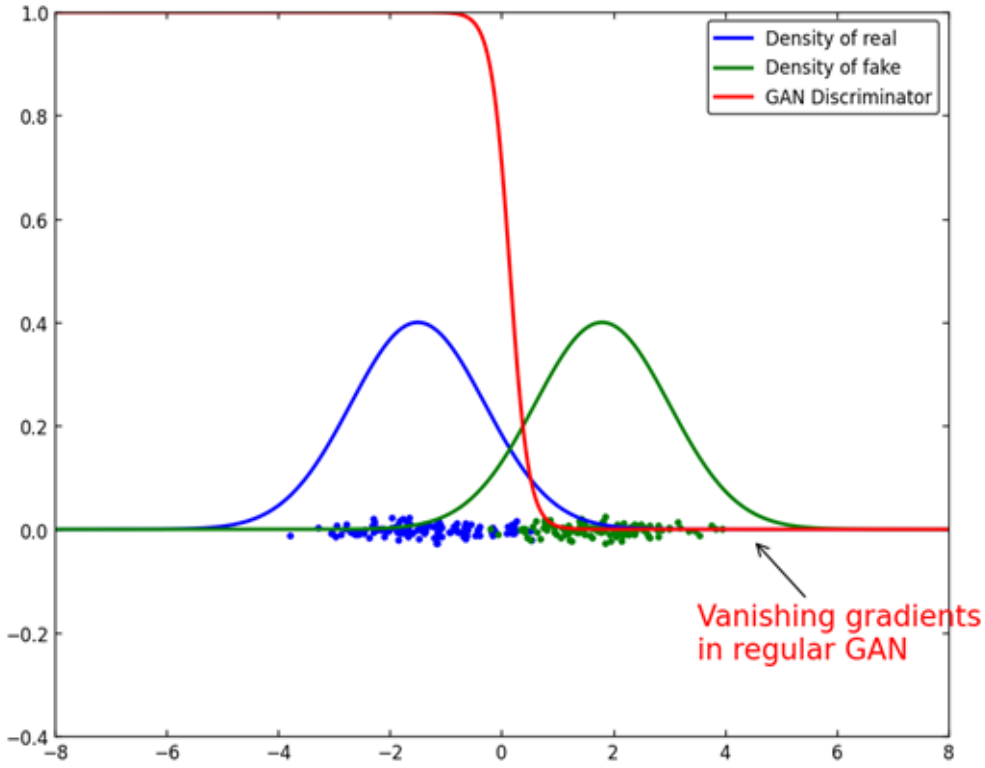
Causes:

Discriminator issue

- too strong → outputs ≈ 0 / 1
- too weak → unreliable feedback

The problem with a strong discriminator





- A strong discriminator perfectly separates real and fake, and does not pass useful gradients to guide the generator.

Problems of Vanilla GANs

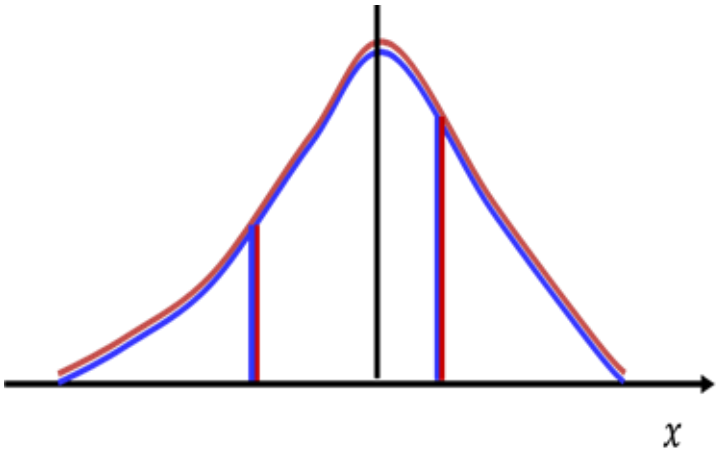
- Vanishing gradients: Generator receives no meaningful gradient for learning

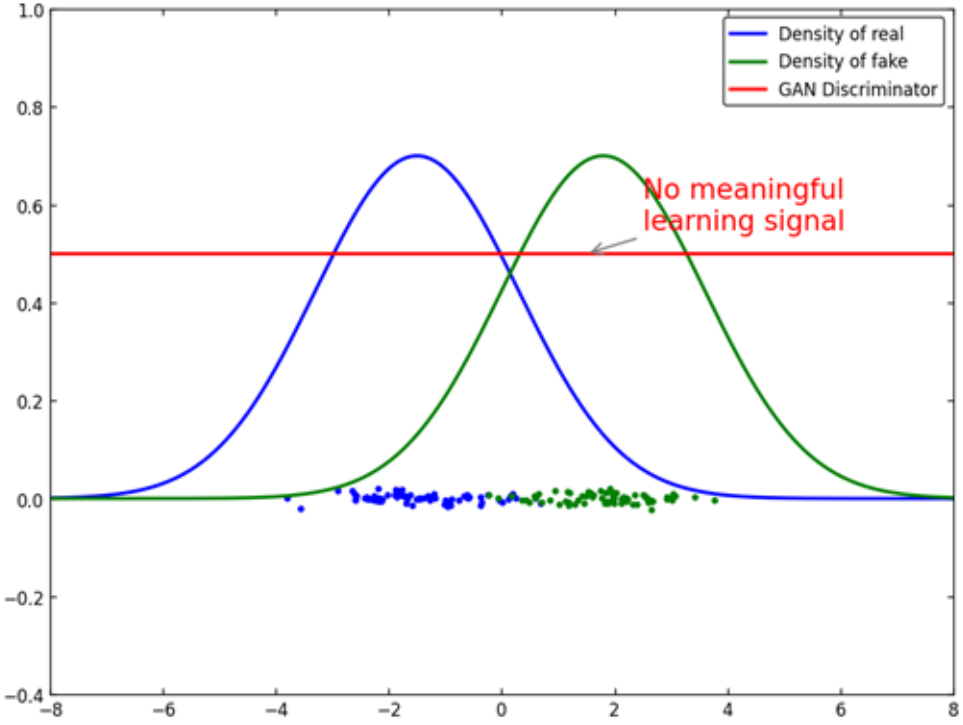
  Causes:

  Discriminator issue

  - too strong → outputs ≈ 0 / 1
  - too weak → unreliable feedback

The problem with a weak discriminator





- A weak discriminator cannot distinguish real from fake, outputs 0.5 everywhere, and thus provides no useful gradients

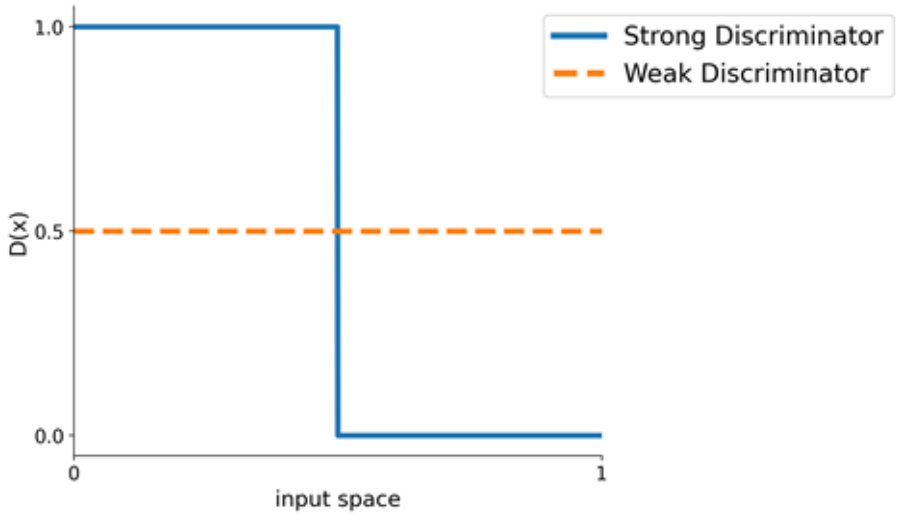
Problems of Vanilla GANs

- Vanishing gradients: Generator receives no meaningful gradient for learning

Causes:

Discriminator issue

- too strong → outputs ≈ 0 / 1
- too weak → unreliable feedback



Problems of Vanilla GANs

- Vanishing gradients: Generator receives no meaningful gradient for learning

Causes:

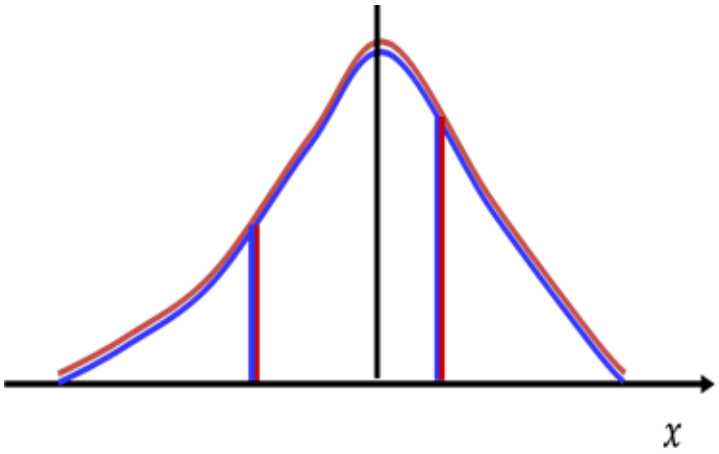
Discriminator issue

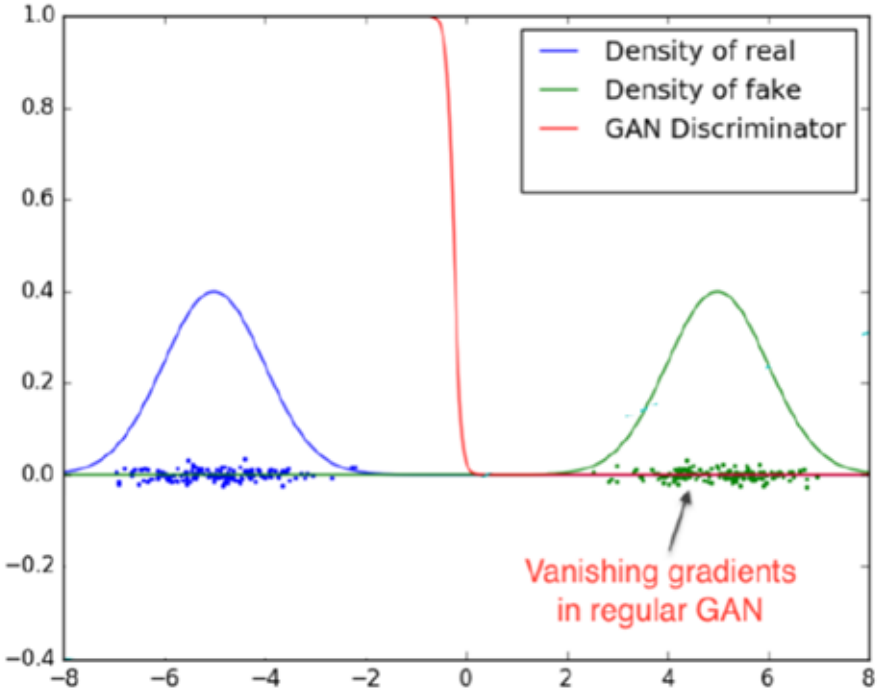
- too strong → outputs ≈ 0 / 1
- too weak → unreliable feedback

Loss issue

- non-overlapping distributions
- sigmoid saturation

Problem with sigmoid loss





- Sigmoid loss leads to saturation, resulting in near-zero gradients

Problems of Vanilla GANs

- Vanishing gradients: Generator receives no meaningful gradient for learning

Causes:

Discriminator issue

- too strong → outputs ≈ 0 / 1
- too weak → unreliable feedback

Loss issue

- non-overlapping distributions
- sigmoid saturation

Divergence issue

Problem with KL divergence

$$KL(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)}$$

- Real data is a point mass at 0
- Generated data is a point mass at θ

Problem with KL divergence

$$KL(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)}$$

Problem with KL divergence

$$KL(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)} \qquad p(x) = \begin{cases} 1 & x = 0 \\ 0 & \text{else} \end{cases} \qquad q(x) = \begin{cases} 1 & x = \theta \\ 0 & \text{else} \end{cases}$$

Case 1: $\theta \neq 0$

$$KL(p\|q) = p(0) \log \frac{p(0)}{q(0)} + p(\theta) \log \frac{p(\theta)}{q(\theta)} = 1 \log \frac{1}{0} + 0 \log \frac{0}{1} = \infty$$

Problem with KL divergence

$$KL(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)} \qquad p(x) = \begin{cases} 1 & x = 0 \\ 0 & \text{else} \end{cases} \qquad q(x) = \begin{cases} 1 & x = \theta \\ 0 & \text{else} \end{cases}$$

Case 1: $\theta \neq 0$

$$KL(p\|q) = p(0) \log \frac{p(0)}{q(0)} + p(\theta) \log \frac{p(\theta)}{q(\theta)} = 1 \log \frac{1}{0} + 0 \log \frac{0}{1} = \infty$$

Case 2: $\theta = 0$

$$KL(p\|q) = p(0) \log \frac{p(0)}{q(0)} = 1 \log \frac{1}{1} = 0$$

Problem with KL divergence

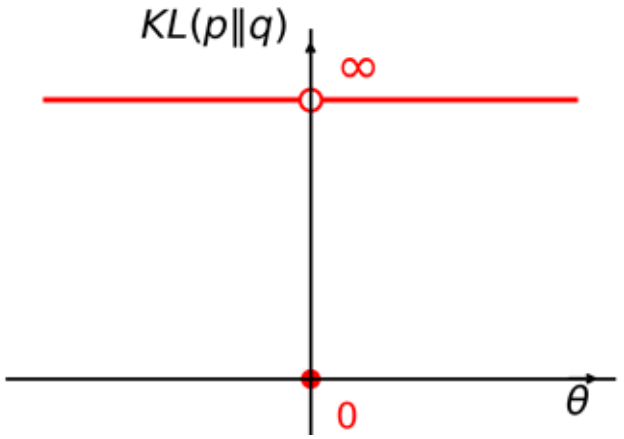
$$KL(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)}$$

- Real data is a point mass at 0
- Generated data is a point mass at θ
- If $\theta \neq 0$, $p(0) \log \frac{p(0)}{q(0)} = 1 \log \frac{1}{0} = \infty$
- If $\theta = 0$, $1 \log \frac{1}{1} = 0$
- Not differentiable w.r.t. θ

Problem with KL divergence

$$KL(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)}$$

- Real data is a point mass at 0
- Generated data is a point mass at θ
- If $\theta \neq 0$, $p(0) \log \frac{p(0)}{q(0)} = 1 \log \frac{1}{0} = \infty$
- If $\theta = 0$, $1 \log \frac{1}{1} = 0$
- Not differentiable w.r.t. θ



Problem with JS divergence

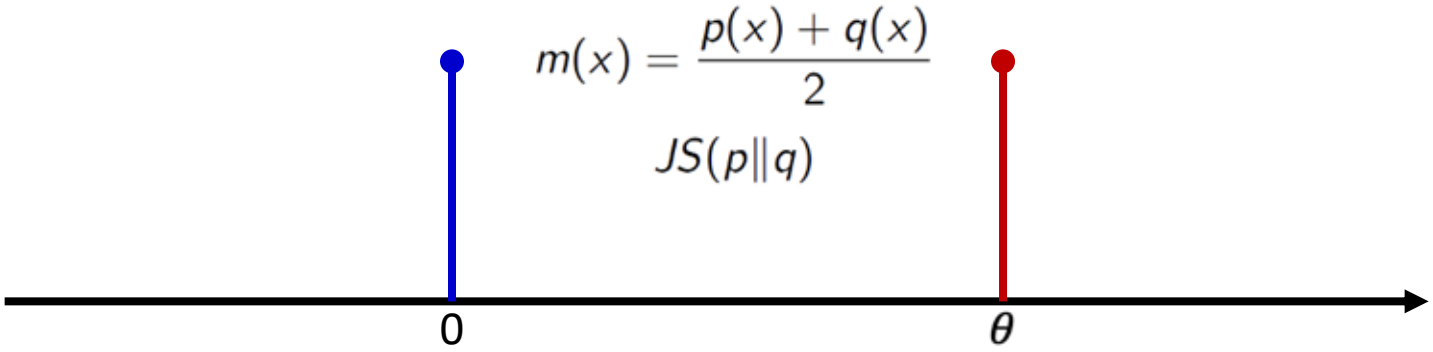
Calculate the average distribution and calculate the average KL to the average.

$$m(x) = \frac{p(x) + q(x)}{2}$$

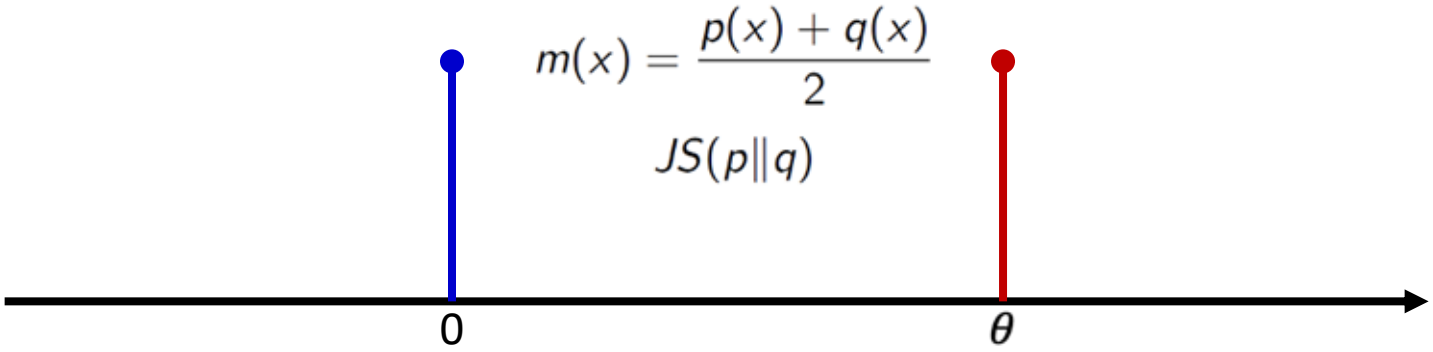
$$JS(p\|q)$$

- Real data is a point mass at 0
- Generated data is a point mass at θ

Problem with JS divergence



Problem with JS divergence

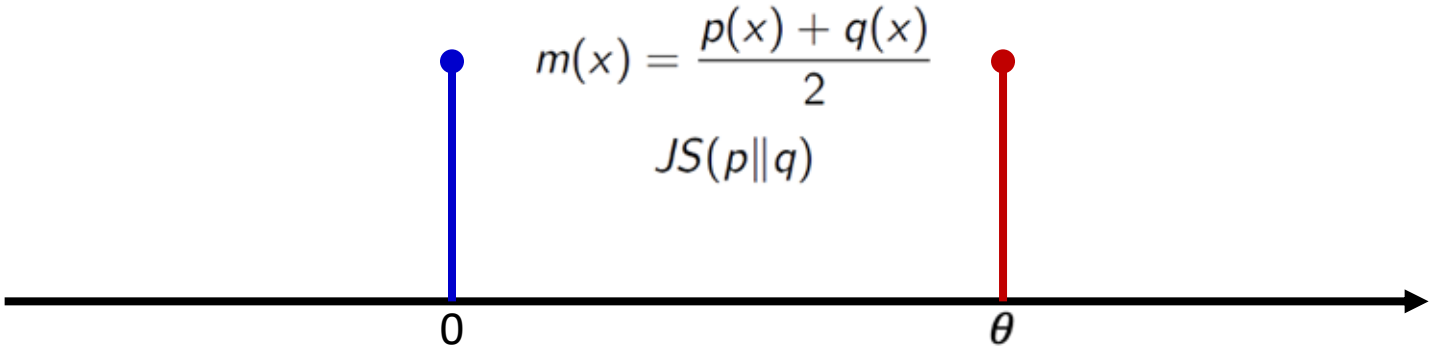


Case 1: $\theta \neq 0$

$$M = \frac{p+q}{2}, \quad M(0) = 0.5,\ M(\theta) = 0.5$$

$$JS(p\|q) = \frac{1}{2}KL(p\|M) + \frac{1}{2}KL(q\|M) = \frac{1}{2}\left(1 \log \frac{1}{0.5} + 0 \log \frac{0}{0.5}\right) + \frac{1}{2}\left(0 \log \frac{0}{0.5} + 1 \log \frac{1}{0.5}\right) = \log 2$$

Problem with JS divergence



Case 1: $\theta \neq 0$

$$M = \frac{p+q}{2}, \quad M(0) = 0.5,\ M(\theta) = 0.5$$

$$JS(p\|q) = \frac{1}{2}KL(p\|M) + \frac{1}{2}KL(q\|M) = \frac{1}{2}\left(1\log\frac{1}{0.5} + 0\log\frac{0}{0.5}\right) + \frac{1}{2}\left(0\log\frac{0}{0.5} + 1\log\frac{1}{0.5}\right) = \log 2$$

Case 2: $\theta = 0$

$$M = p = q, \quad M(0) = 1$$

$$JS(p\|q) = \frac{1}{2}KL(p\|M) + \frac{1}{2}KL(q\|M) = \frac{1}{2}\left(1\log\frac{1}{1}\right) + \frac{1}{2}\left(1\log\frac{1}{1}\right) = 0$$

Problem with JS divergence

Calculate the average distribution and calculate the average KL to the average.

$$m(x) = \frac{p(x) + q(x)}{2}$$

$$JS(p\|q)$$

- Real data is a point mass at 0
- Generated data is a point mass at θ
- If $\theta \neq 0$, $\frac{1}{2}\left[1 \log \frac{1}{0.5} + 1 \log \frac{1}{0.5}\right] = \log 2$
- If $\theta = 0$, $\frac{1}{2}\left[1 \log \frac{1}{1} + 1 \log \frac{1}{1}\right] = 0$
- Not differentiable w.r.t. θ

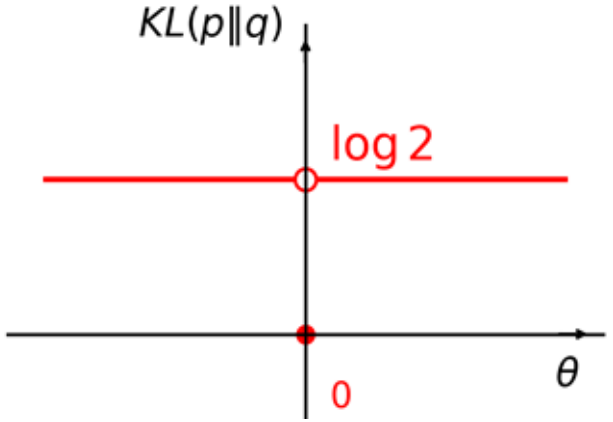
Problem with JS divergence

Calculate the average distribution and calculate the average KL to the average.

$$m(x) = \frac{p(x) + q(x)}{2}$$

$$JS(p\|q)$$

- Real data is a point mass at 0
- Generated data is a point mass at θ
- If $\theta \neq 0$, $\frac{1}{2}\left[1 \log \frac{1}{0.5} + 1 \log \frac{1}{0.5}\right] = \log 2$
- If $\theta = 0$, $\frac{1}{2}\left[1 \log \frac{1}{1} + 1 \log \frac{1}{1}\right] = 0$
- Not differentiable w.r.t. θ



Least-Squares GAN

- Use an L_2 loss instead of cross-entropy
- Generator tries to minimize L_2 loss

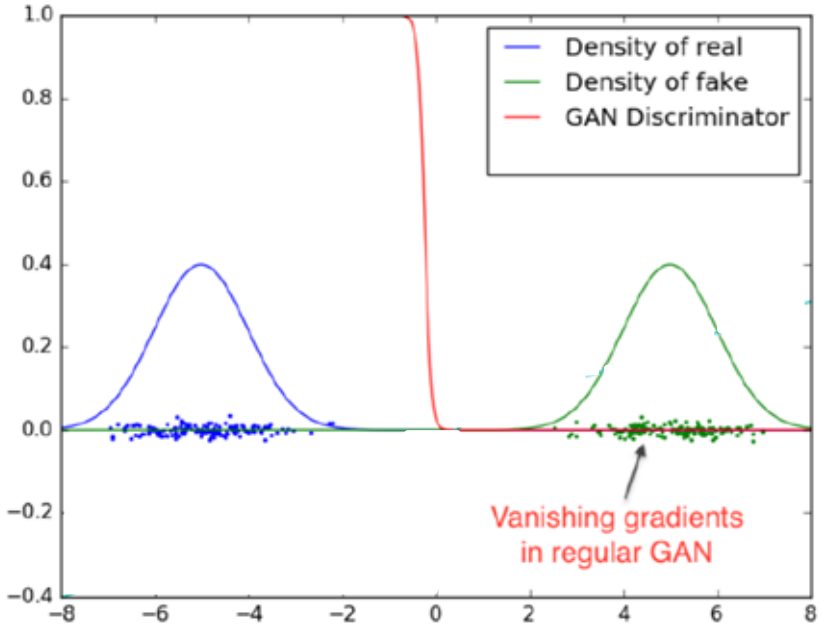
$$E_{z\sim P(z)}\left[\left(D\left(G(z)\right)-1\right)^2\right]$$

- Discriminator tries to minimize L_2 loss

$$E_{x\sim P(x)}[(D(x)-1)^2]+E_{z\sim P(z)}\left[\left(D\left(G(z)\right)\right)^2\right]$$

- Less squashing and fewer large values than cross-entropy

Least-squares GAN



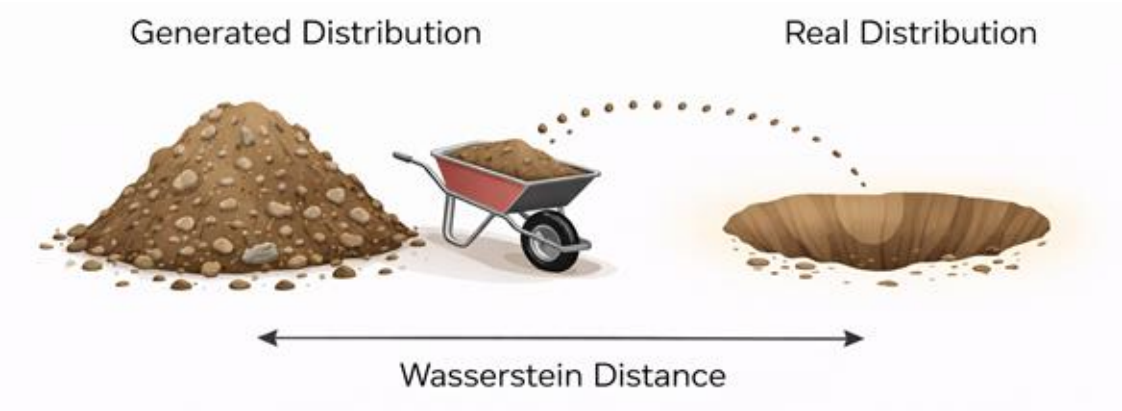
- The loss is better behaved than KL
- But the discriminator itself can continue to be misbehaved

- Can we do better?

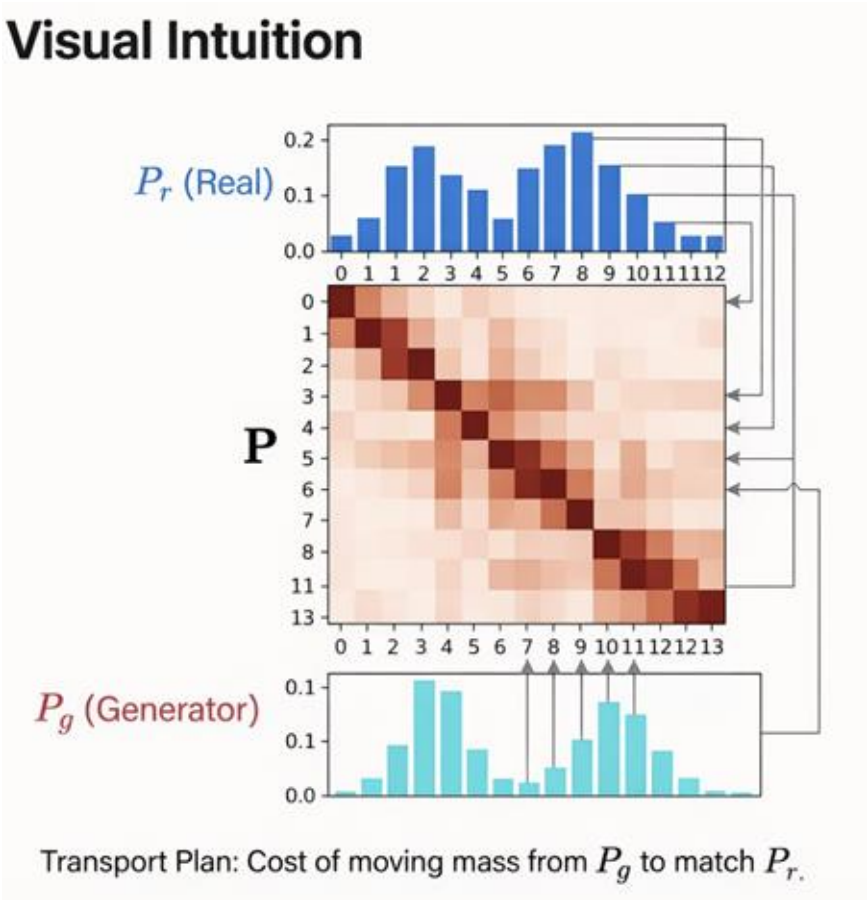
Wasserstein Distance

The total $\sum$ mass $\times$ distance required to transform one distribution to another.

- Intuitive, symmetric measure of divergence
- Hard to calculate because requires solving "optimal transport"
- You have some distribution of products at some warehouses and you need to make some other distribution of products at those warehouses
- Move the products to the target distribution minimizing $\sum$ mass $\times$ distance
- Creating the optimal "transport plan" can be NP-hard



Wasserstein Distance



Definition & Logic

The Wasserstein Distance (W) measures the minimum cost to transport probability mass from one distribution to another.

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

Slide credit: https://youtu.be/G-Ed5wbKiU8?si=zmQUK6NFzdUUvN-M

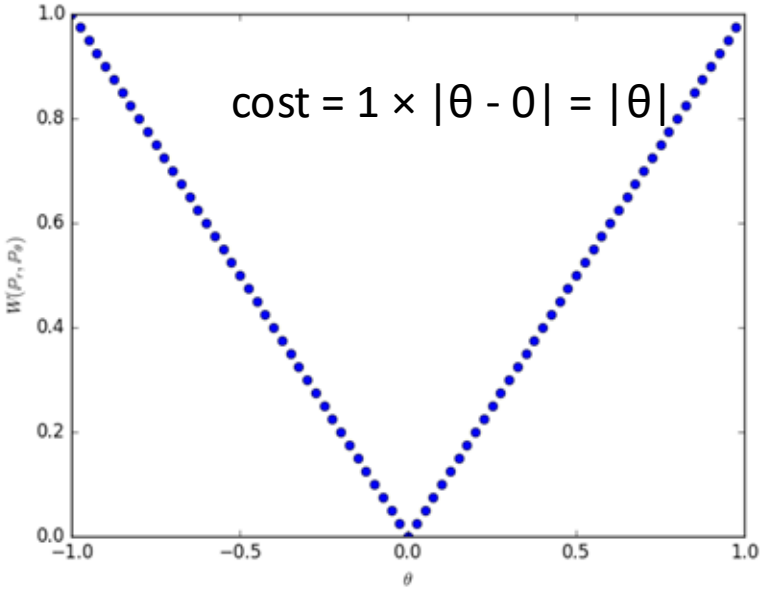
Wasserstein Distance

- Real data is a point mass at 0
- Generated data is a point mass at θ

Wasserstein GANs: Two-mass case

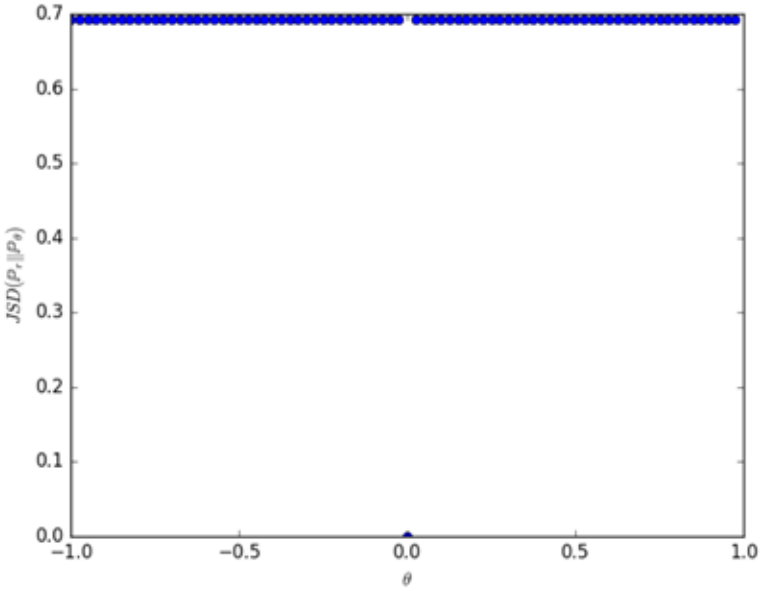
- Uses **Earth-mover (Wasserstein-1) distance** to measure the distance between two distributions

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



Wasserstein-1

Differentiable w.r.t θ Continuous and useful gradient everywhere

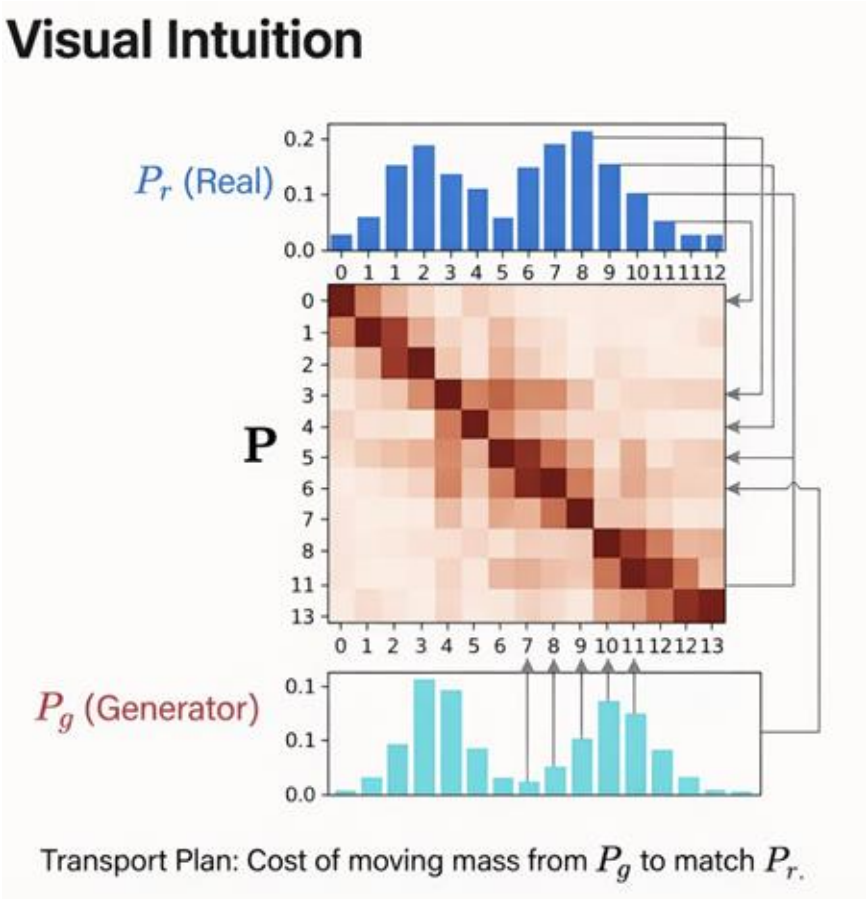


JS

Not Continuous

Arjovsky et al. Wasserstein GAN. 2017.

Wasserstein Distance



Definition & Logic

The Wasserstein Distance (W) measures the minimum cost to transport probability mass from one distribution to another.

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

Why it works:

- Unlike JS divergence, W-distance is continuous and differentiable almost everywhere.
- Even when distributions are disjoint, the distance reflects how far apart they are (linear gradient).
- It provides a meaningful gradient flow for the generator to “push” the mass towards the real data.

Slide credit: https://youtu.be/G-Ed5wbKiU8?si=zmQUK6NFzdUUvN-M

Kantorovich-Rubinstein Duality

Since the infimum in the primal form is intractable to compute directly, we use the dual form.

$$W(P_r, P_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_g}[f(x)]$$

Slide credit: https://youtu.be/G-Ed5wbKiU8?si=zmQUK6NFzdUUvN-M

Kantorovich-Rubinstein Duality

Since the infimum in the primal form is intractable to compute directly, we use the dual form.

The Critic. Replaces the Discriminator. Outputs a raw score (realness), not a probabiity.

$$W(P_r, P_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_g}[f(x)]$$

Slide credit: https://youtu.be/G-Ed5wbKiU8?si=zmQUK6NFzdUUvN-M

Kantorovich-Rubinstein Duality

Since the infimum in the primal form is intractable to compute directly, we use the dual form.

The Critic. Replaces the Discriminator. Outputs a raw score (realness), not a probabiity.

$$W(P_r, P_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_g}[f(x)]$$

Maximization. We train the critic to maximize the difference between scores for real and fake data.

Slide credit: https://youtu.be/G-Ed5wbKiU8?si=zmQUK6NFzdUUvN-M

Kantorovich-Rubinstein Duality

Since the infimum in the primal form is intractable to compute directly, we use the dual form.

The Critic. Replaces the Discriminator. Outputs a raw score (realness), not a probabiity.

$$W(P_r, P_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_g}[f(x)]$$

Maximization. We train the critic to maximize the difference between scores for real and fake data.

1-Lipschitz Constraint. The function f cannot change too quickly. The gradient norm must be $\leq$ 1everywhere.

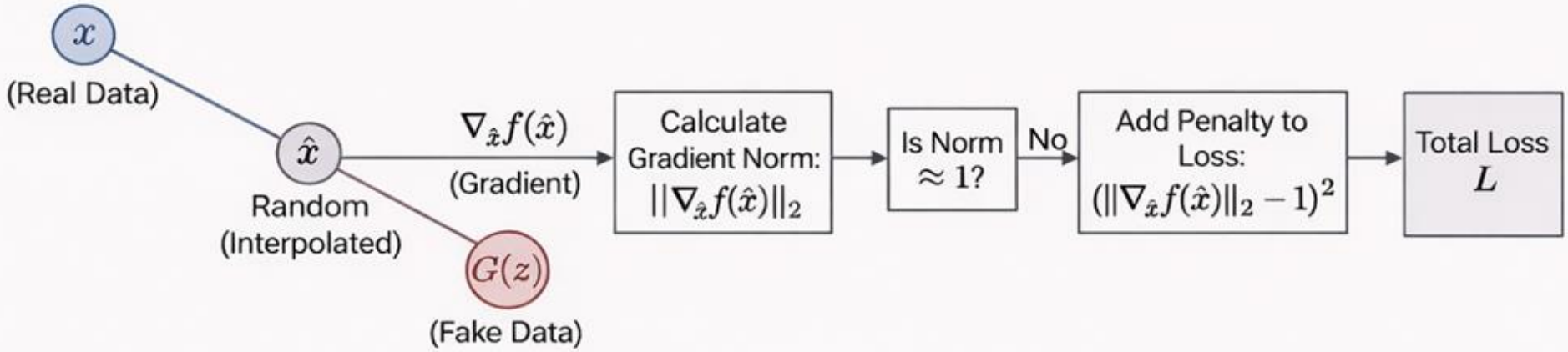
Without the Lipschitz constraint, the critic would assign $-\infty$ to fake samples and $+\infty$ to real samples, causing the loss to explode.

Slide credit: https://youtu.be/G-Ed5wbKiU8?si=zmQUK6NFzdUUvN-M

WGAN with Gradient Penalty (WGAN-GP)

To enforce the Lipschitz constraint, we avoid weight clipping (which leads to optimization issues) and instead penalize the gradient norm directly.

$$L = \underbrace{\mathbb{E}[f(G(z))] - \mathbb{E}[f(x)]}_{\text{Original Wasserstein Estimate}} + \underbrace{\lambda \mathbb{E}_{\hat{x}}\left[(\|\nabla_{\hat{x}} f(\hat{x})\|_2 - 1)^2\right]}_{\text{Gradient Penalty}}$$



Slide credit: https://youtu.be/G-Ed5wbKiU8?si=zmQUK6NFzdUUvN-M

Wasserstein GAN

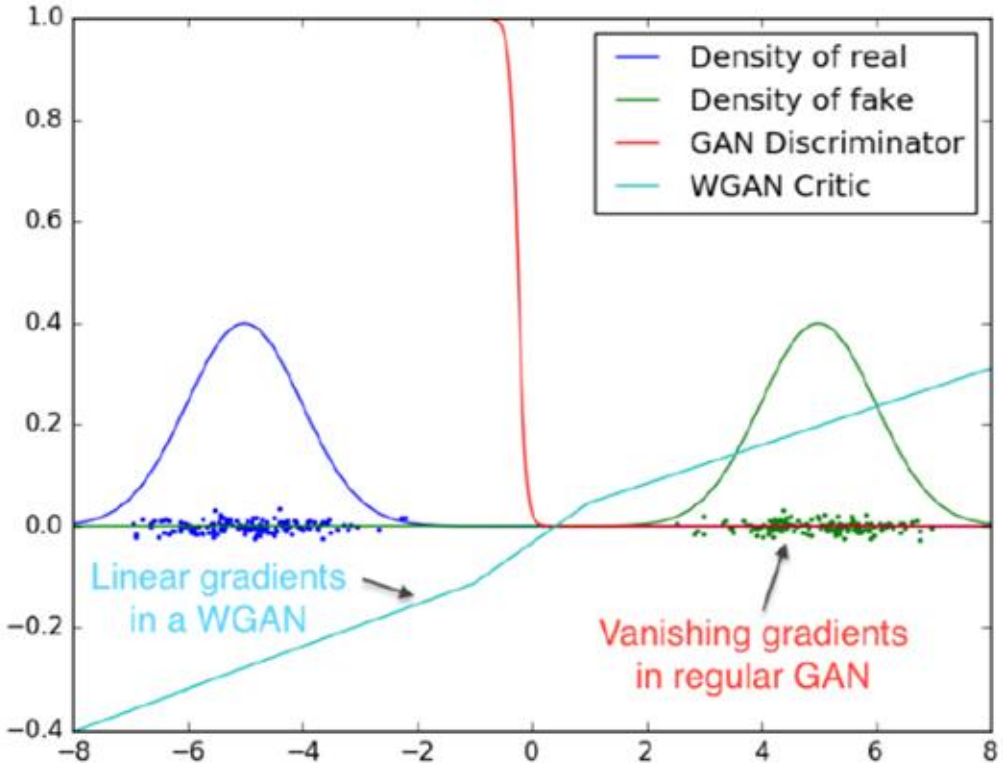


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

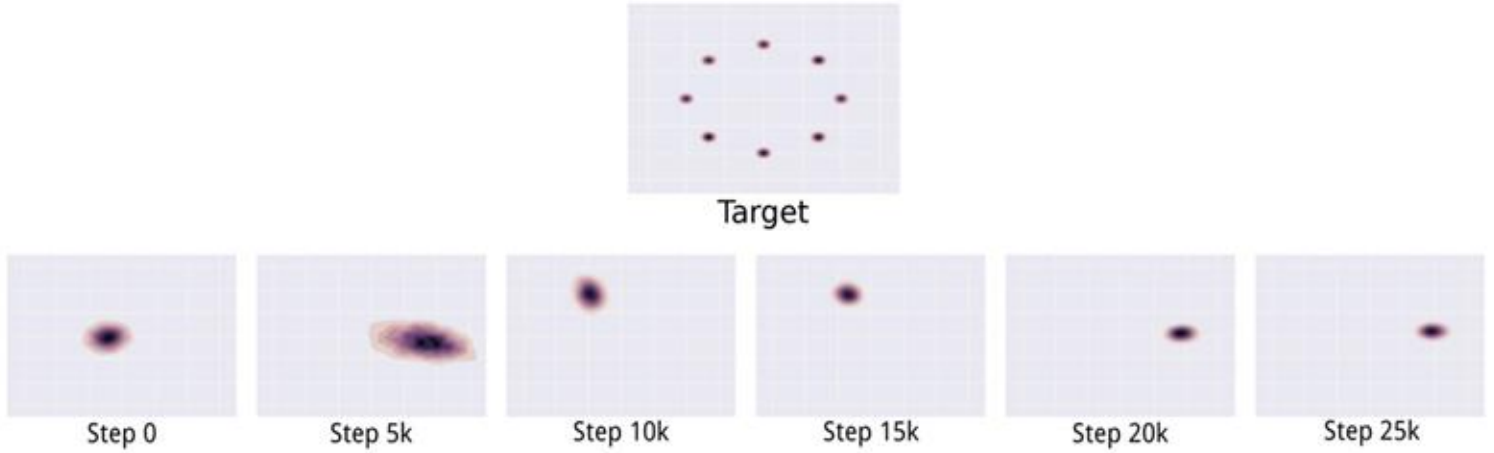
Wasserstein GAN vs regular GAN which is which?

Miyato et al. Spectral Normalization of Generative Adversarial Networks. 2018.

Problems of Vanilla GANs

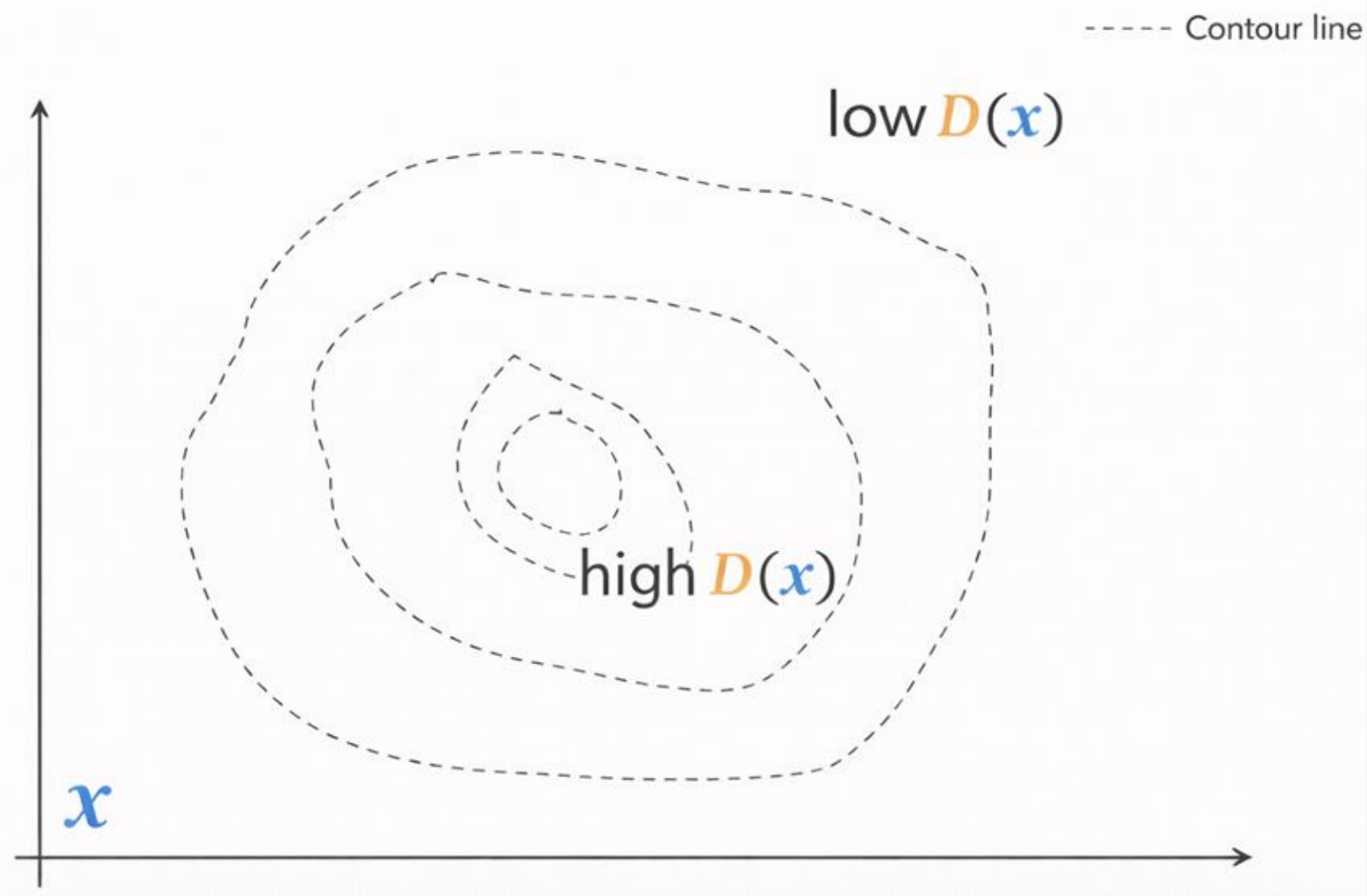
- Vanishing gradients: Generator receives no meaningful gradient for learning
- Mode collapse: the generator distribution collapses to a small set of samples

Mode Collapse in Generator



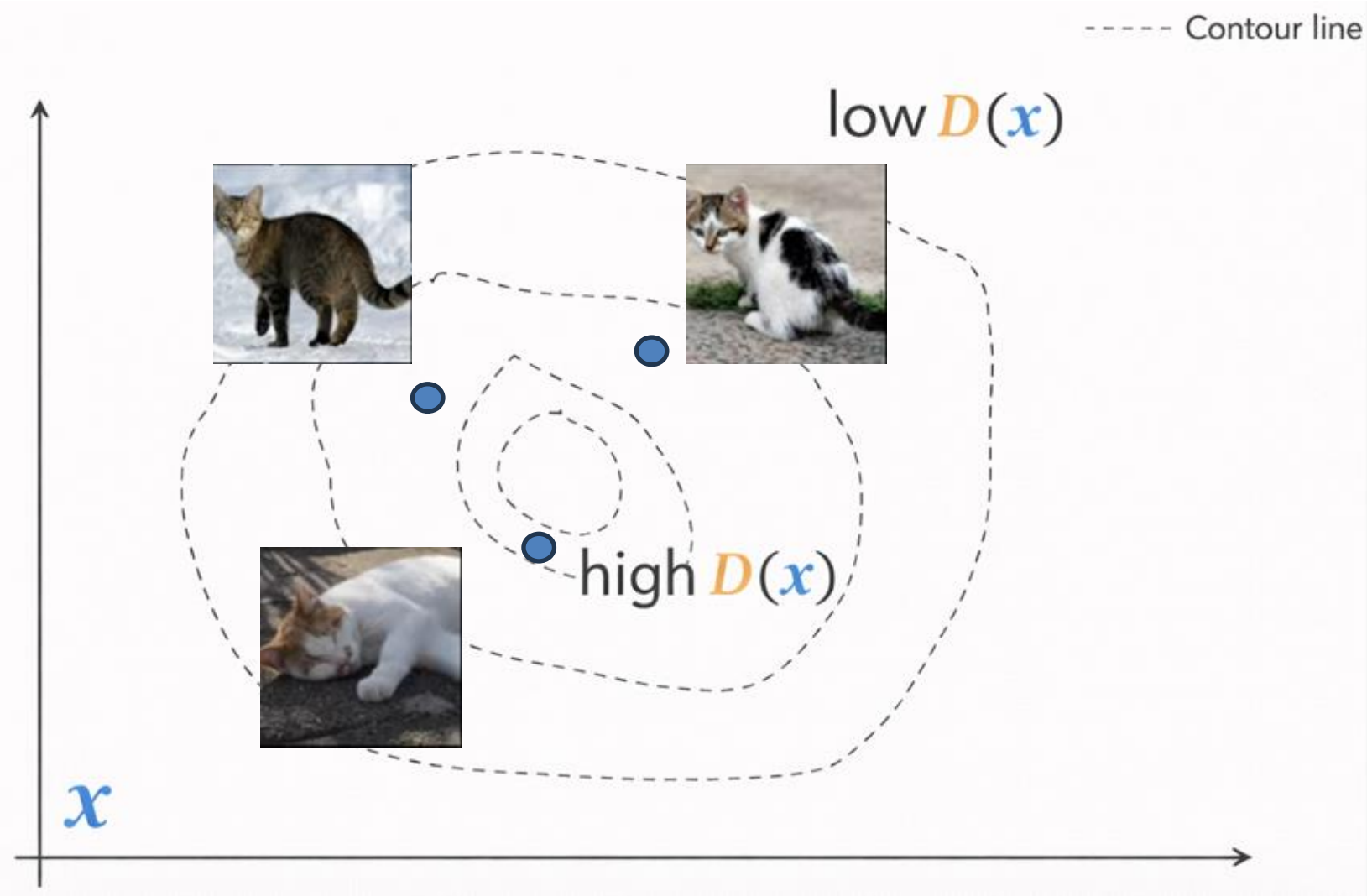
- For different z, the generator produces similar output
- The generator learns to fool the discriminator perfectly by only capturing a subset of the data

Mode Collapse in Generator



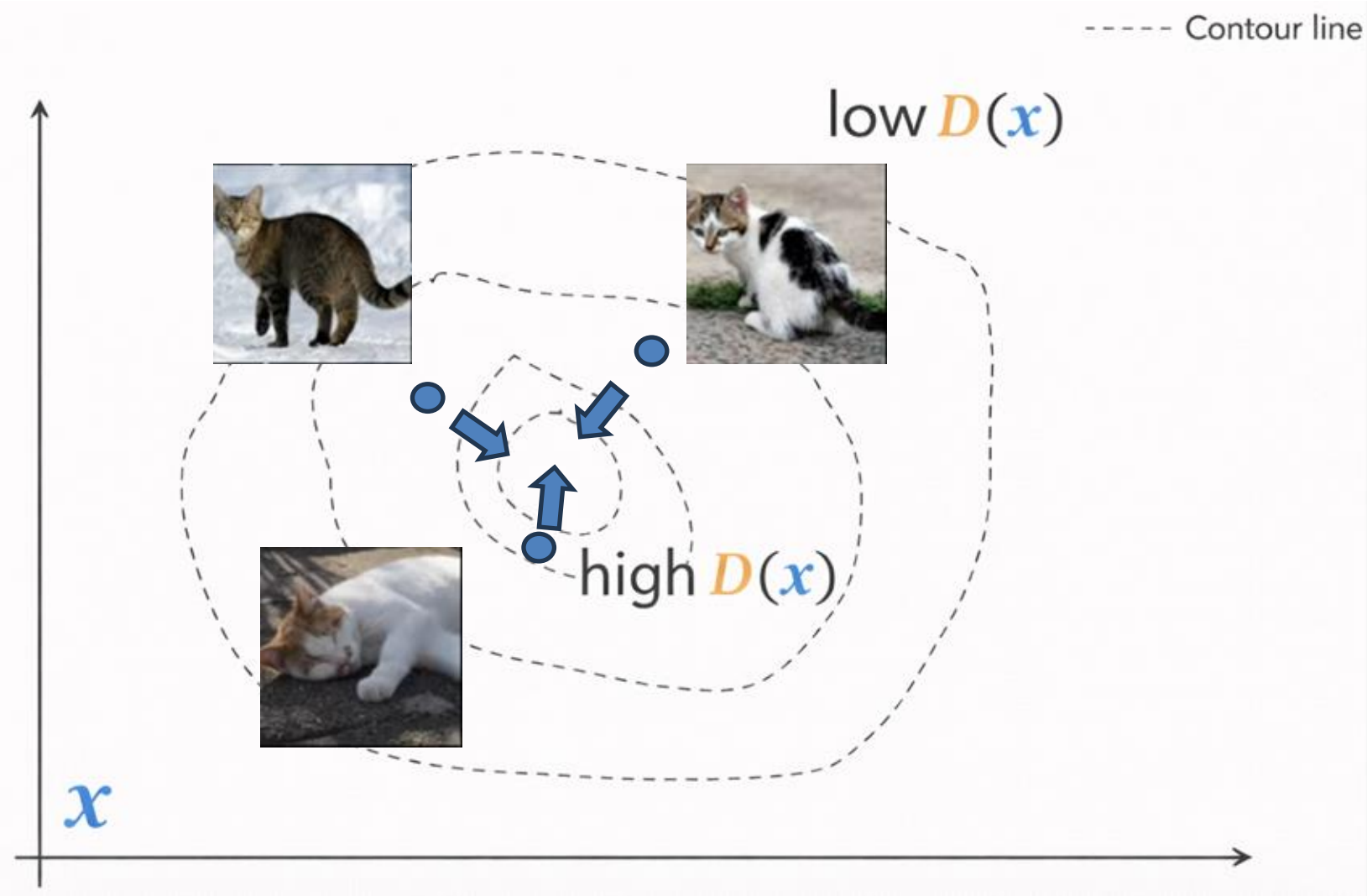
Slide credit: https://youtu.be/RAa55G-oEuk?si=lF2t_t9BPjlQOhiT

Mode Collapse in Generator



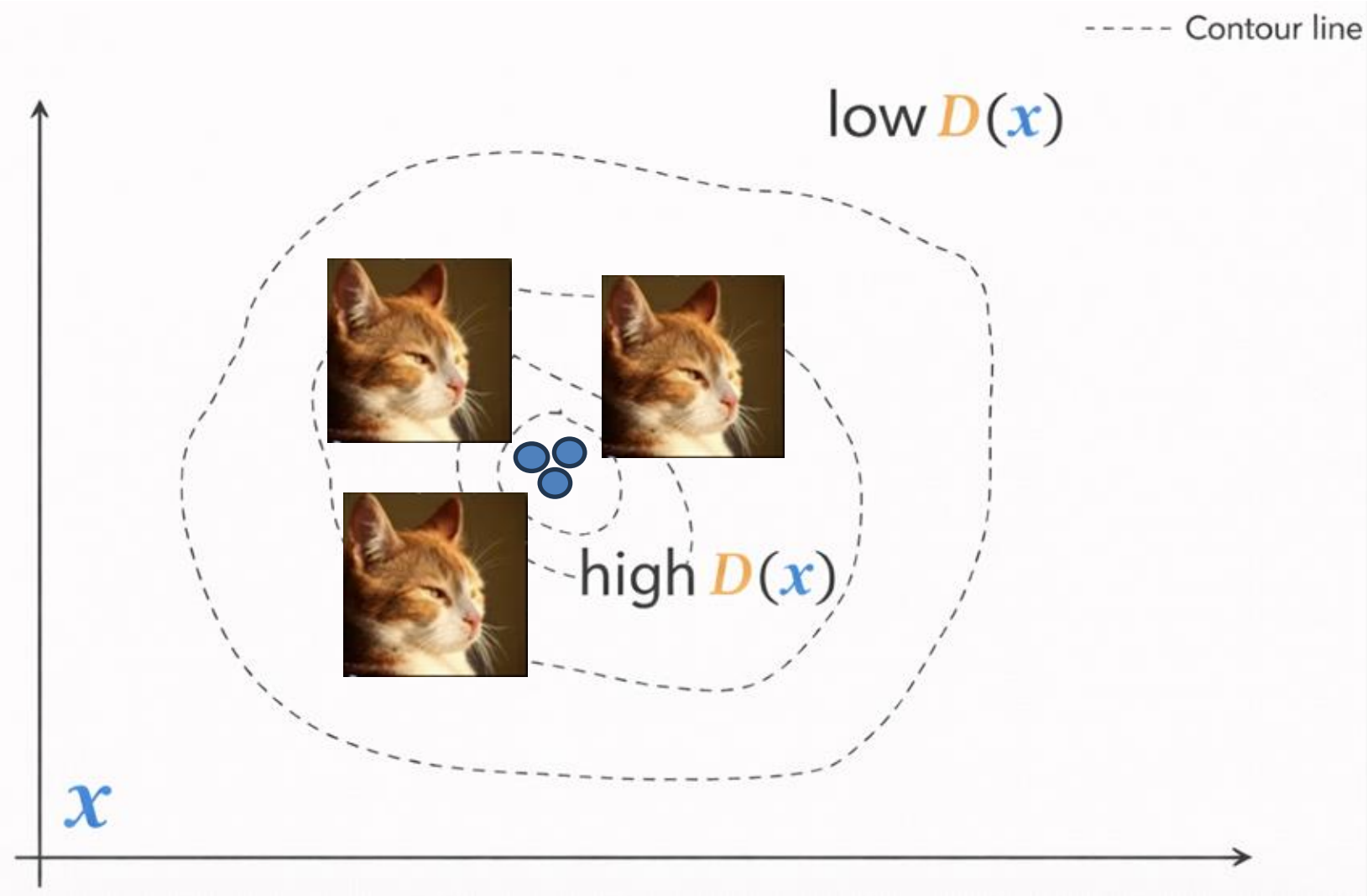
Slide credit: https://youtu.be/RAa55G-oEuk?si=lF2t_t9BPjlQOhiT

Mode Collapse in Generator



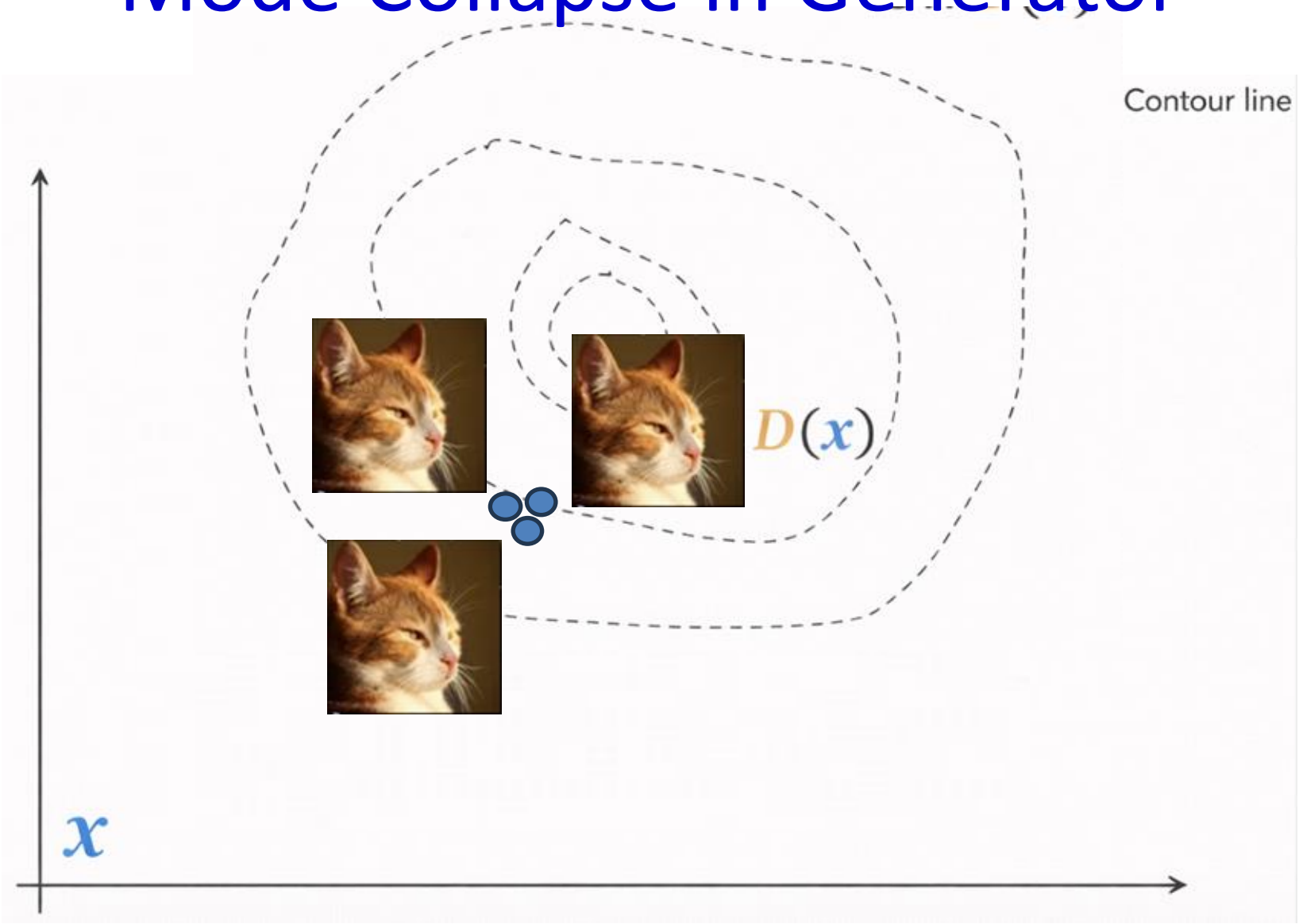
Slide credit: https://youtu.be/RAa55G-oEuk?si=lF2t_t9BPjlQOhiT

Mode Collapse in Generator



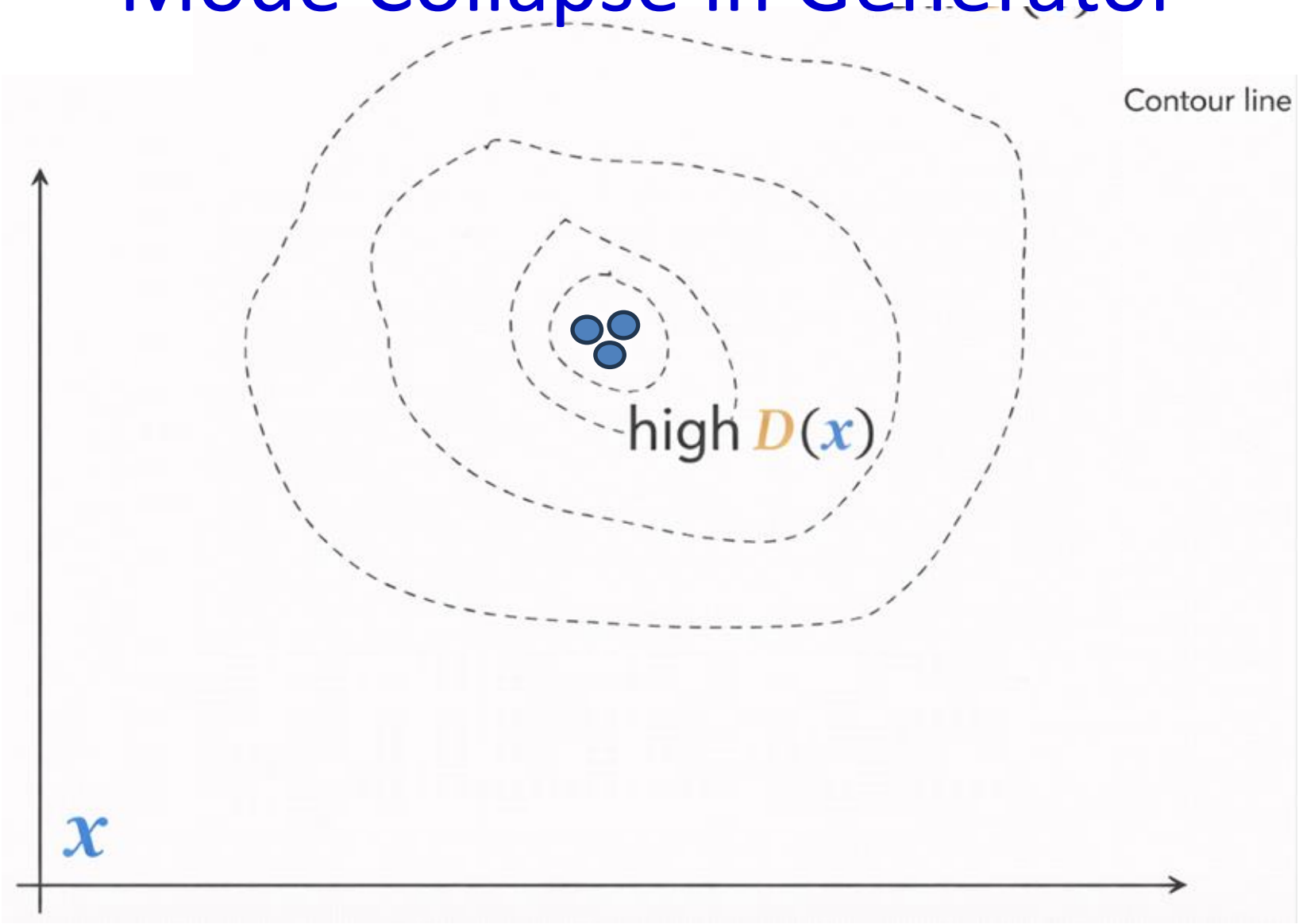
Slide credit: https://youtu.be/RAa55G-oEuk?si=lF2t_t9BPjlQOhiT

Mode Collapse in Generator



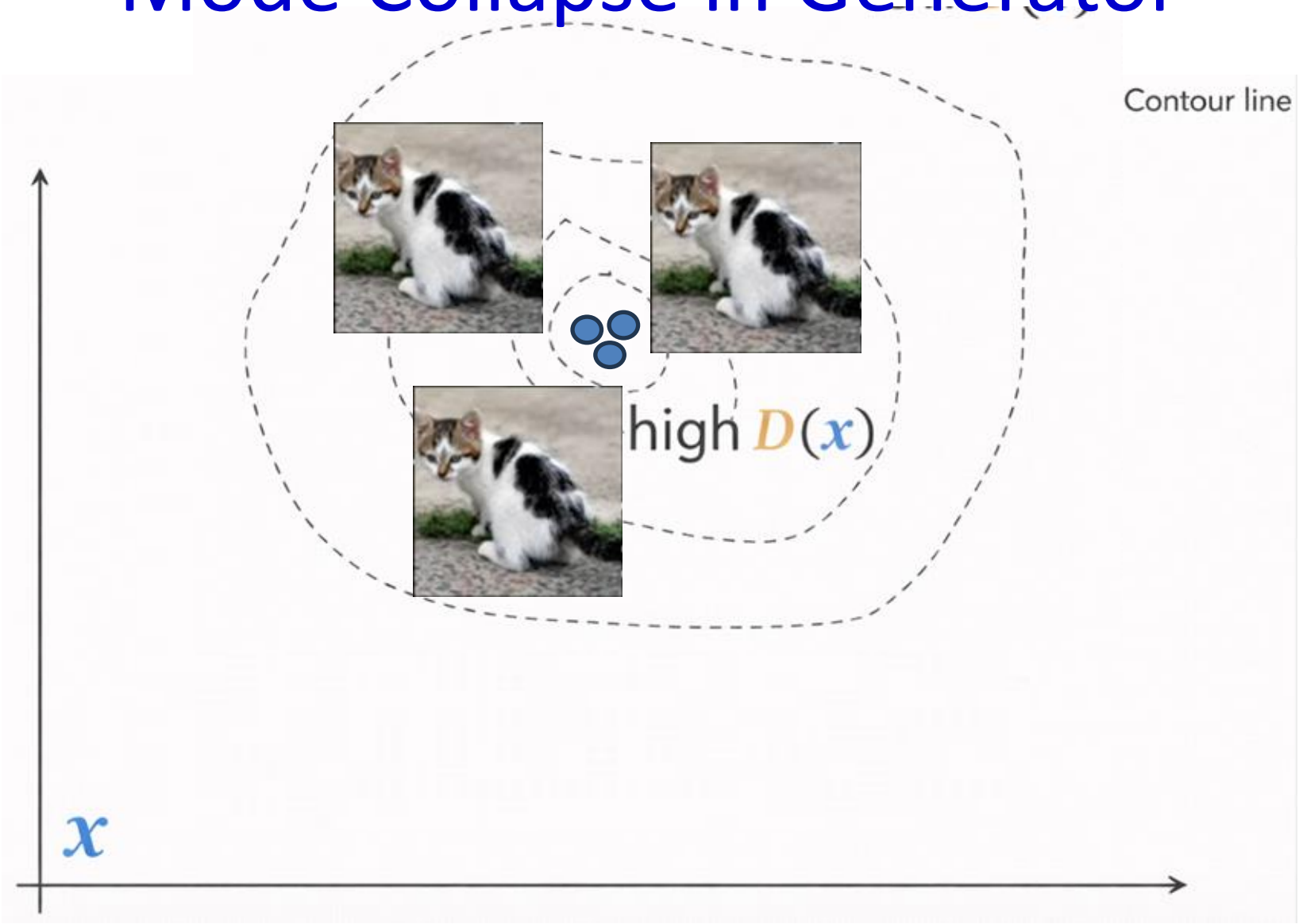
Slide credit: https://youtu.be/RAa55G-oEuk?si=lF2t_t9BPjlQOhiT

Mode Collapse in Generator



Slide credit: https://youtu.be/RAa55G-oEuk?si=lF2t_t9BPjlQOhiT

Mode Collapse in Generator



Slide credit: https://youtu.be/RAa55G-oEuk?si=lF2t_t9BPjlQOhiT

Mode Collapse in Generator

Mode collapse is a persistent problem in GANs

limited diversity and repetitive outputs

needs attention in practice

Mode Collapse in Generator

Mode collapse is a persistent problem in GANs

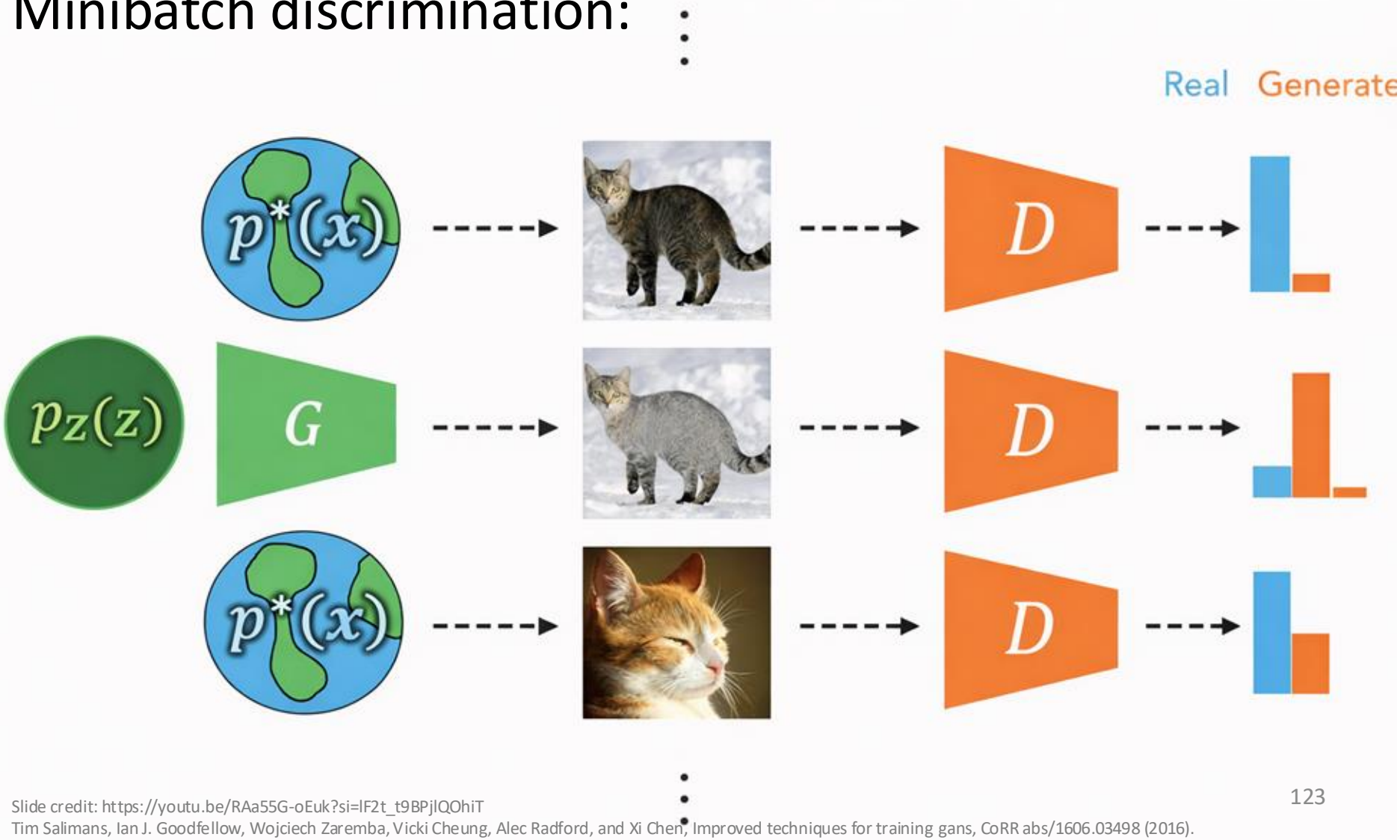
limited diversity and repetitive outputs

needs attention in practice

- **Solutions**
  - Minibatch discrimination: detect similarity across samples

Mode Collapse in Generator

Minibatch discrimination:

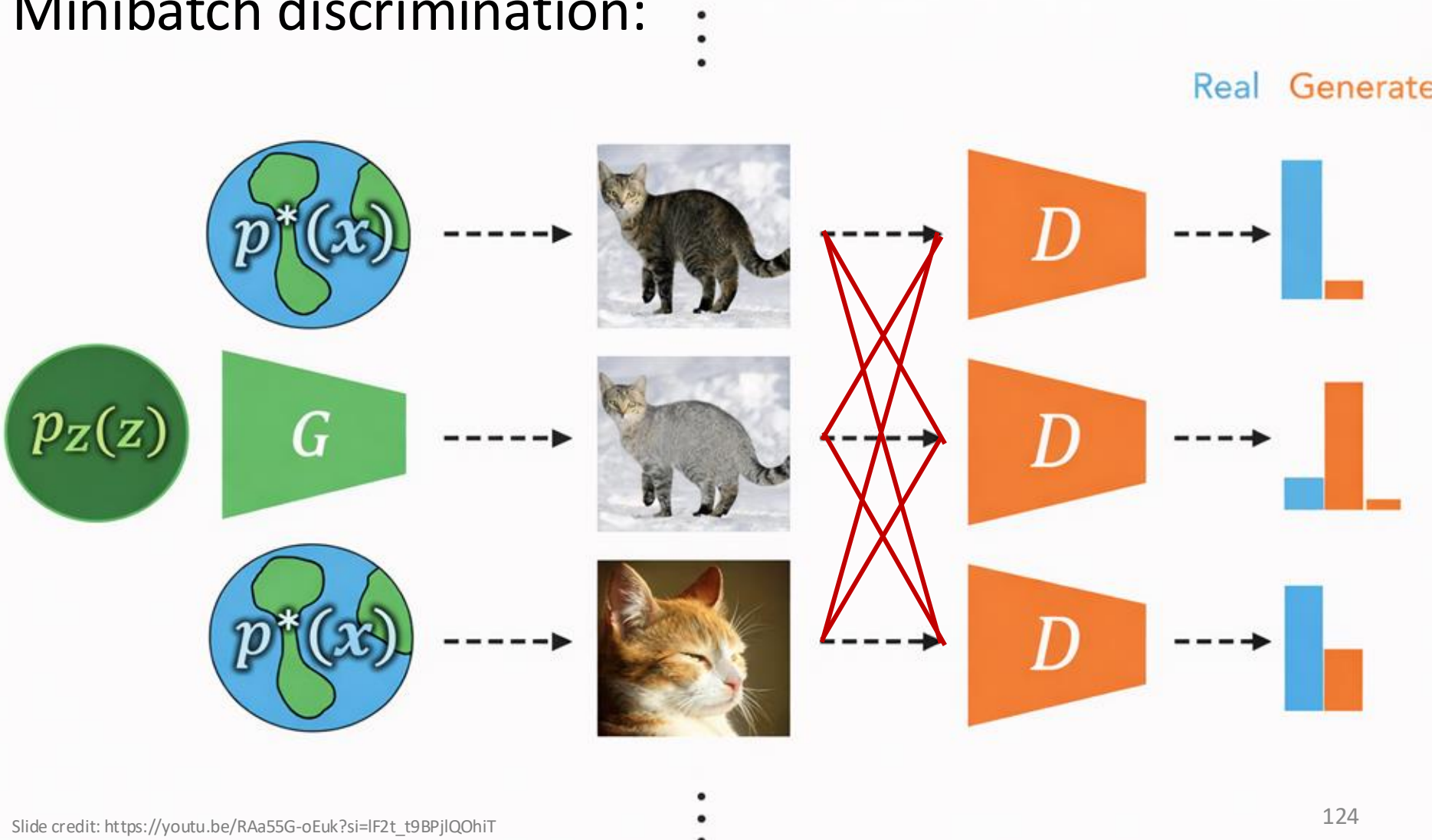


Slide credit: https://youtu.be/RAa55G-oEuk?si=lF2t_t9BPjlQOhiT

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen, Improved techniques for training gans, CoRR abs/1606.03498 (2016).

Mode Collapse in Generator

Minibatch discrimination:



Slide credit: https://youtu.be/RAa55G-oEuk?si=lF2t_t9BPjlQOhiT

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen, Improved techniques for training gans, CoRR abs/1606.03498 (2016).

Mode Collapse in Generator

Mode collapse is a persistent problem in GANs

limited diversity and repetitive outputs

needs attention in practice

- **Solutions**
  - Minibatch discrimination: detect similarity across samples
  - Wasserstein GAN: distribution-level loss encourages diversity

Solutions to Problems of Vanilla GANs

- Vanishing gradients: Generator receives no meaningful gradient for learning
  - Least-Squares GAN (LSGAN)
  - Wasserstein GAN (WGAN/WGAN-GP)
  - Non-saturating loss
  - Hinge loss
  - f-GAN

  …
- Mode collapse: the generator distribution collapses to a small set of samples
  - Minibatch discrimination
  - WGAN
  - Feature matching
  - Unrolled GAN
  - Instance noise

  …

Improvements and Applications of GANs

- Problems of Vanilla GANs and their solutions
- Architecture of GANs
  - Conditional GANs
  - Progressive GANs
  - StyleGAN
- Image-to-Image Translation GANs
- Text-to-Image GANs

Architecture Evolution of GANs

Conditional and disentangled GAN variants (2014–2016)

Image-to-image translation GANs (2017)

Scaling up GANs: Self-attention and large-scale models (2018–2019)

Style-based generators and high-resolution GANs (2018–2020)

Transformer-based and hybrid generative models (2020–present)

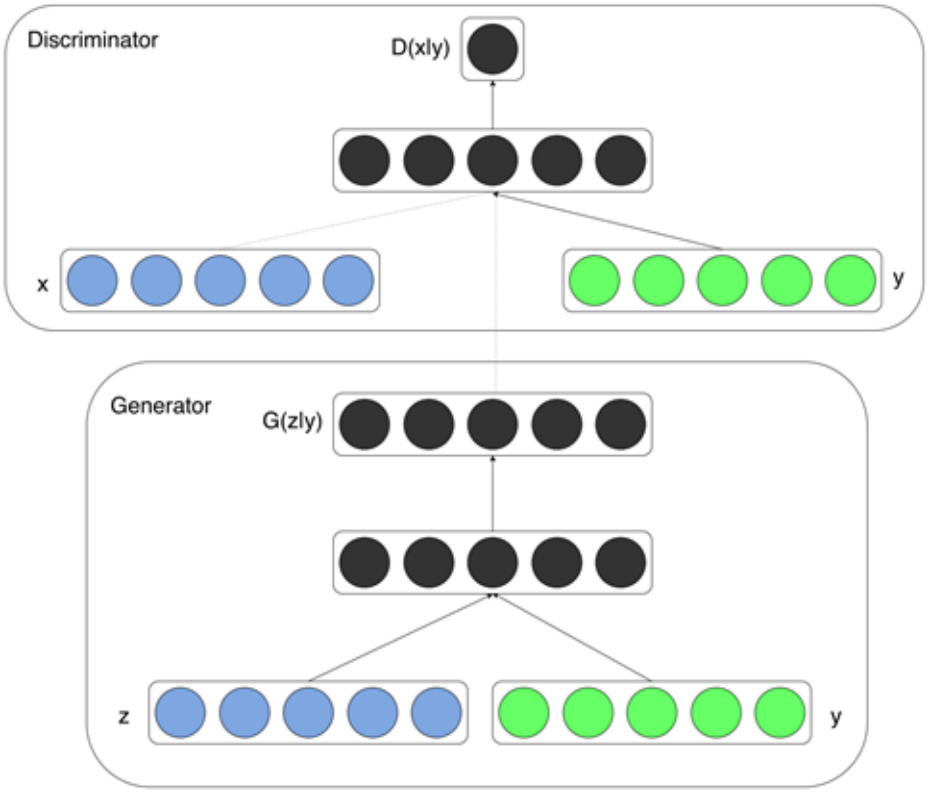
Adapted from: Al-Yaari & Deng (2026), *Generative Adversarial Networks: A Comprehensive Survey*

Controlled Generation: Conditional GAN

Learning to generate data conditioned on labels or inputs.

- Generator is conditioning on y

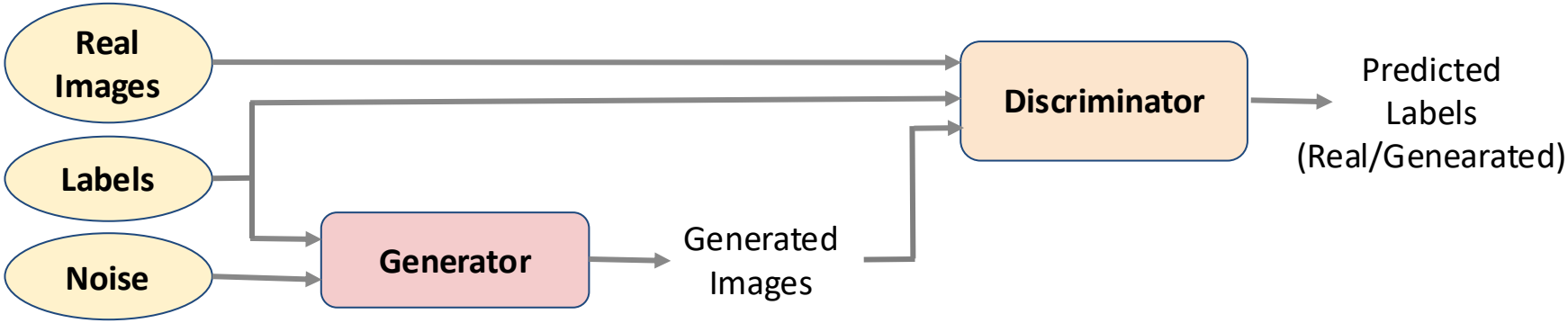
- Discriminator predicts both
  - Real vs Fake
  - Class of the images

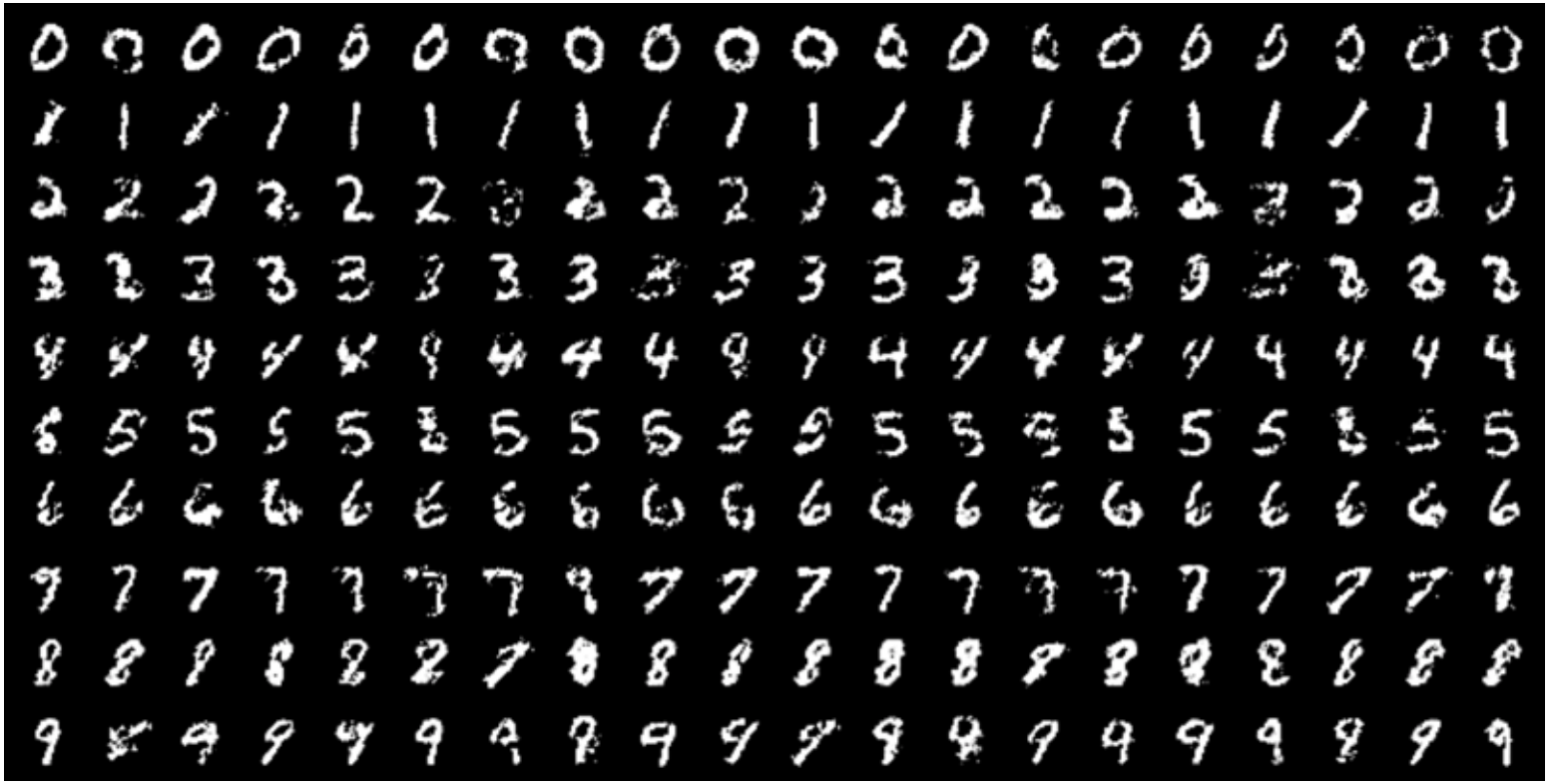


Mirza et al. Conditional Generative Adversarial Nets. 2014.

Controlled Generation: Conditional GAN

Learning to generate data conditioned on labels or inputs.



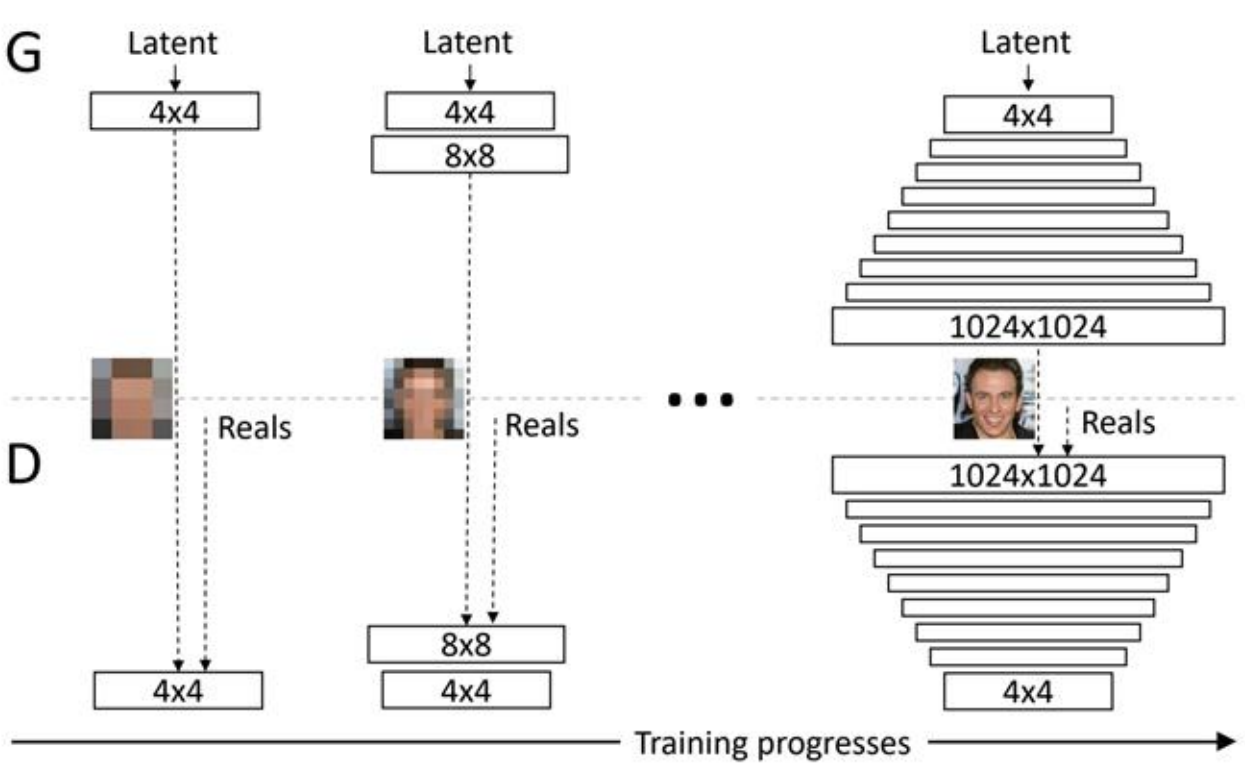


Mirza et al. Conditional Generative Adversarial Nets. 2014.

High-Resolution Generation: Progressive GAN

Growing the model and resolution progressively for stable training.

- Grow both generator and discriminator progressively
  - Speed up and stabilize training
  - High-resolution image generation



Karras et al. Progressive Growing of GANs for Improved Quality, Stability, and Variation. 2017.

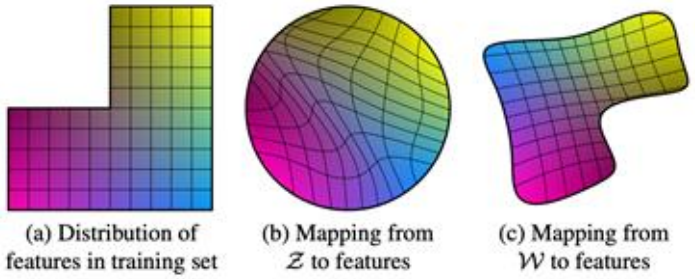
Disentangled Generation: StyleGAN

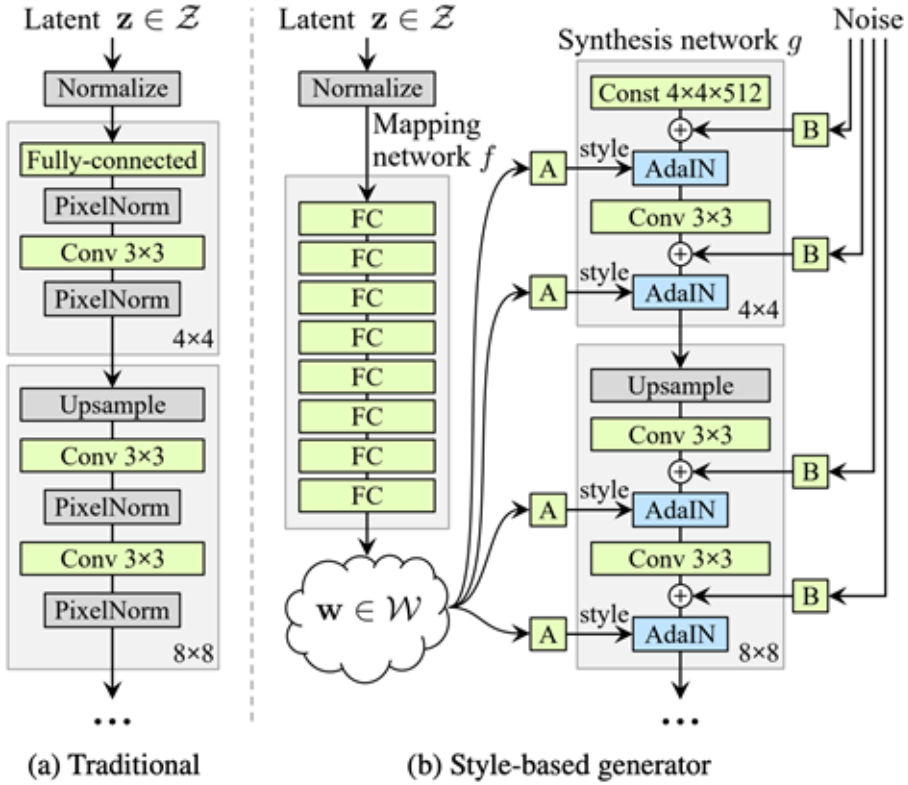
Separating structure and randomness via style modulation.

- AdaIN

$$\mathrm{AdaIN}\left(\mathbf{x}_i, \mathbf{y}\right) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu\left(\mathbf{x}_i\right)}{\sigma\left(\mathbf{x}_i\right)} + \mathbf{y}_{b,i}$$

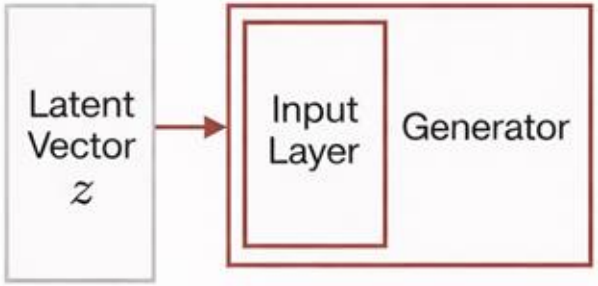
- A better latent space





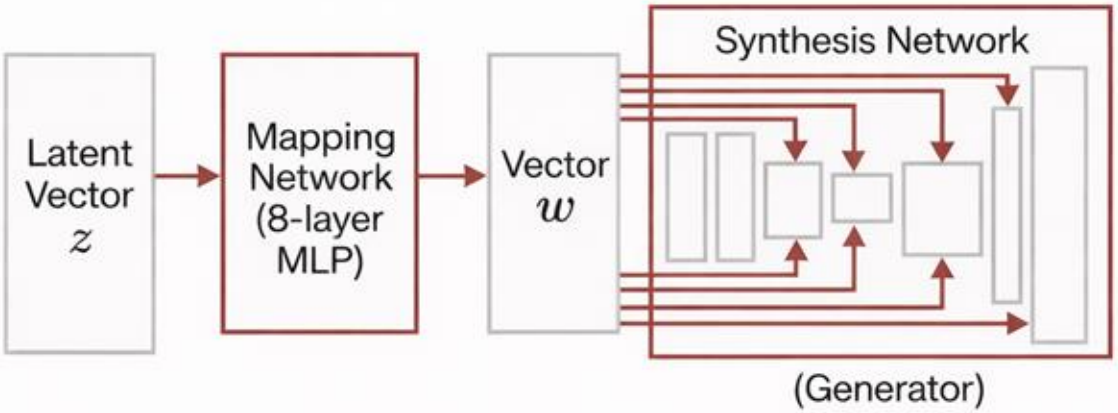
StyleGAN: Disentangling Latent Space

Traditional GAN



Entangled: Factors of variation are warped to fit the Gaussian prior.

StyleGAN

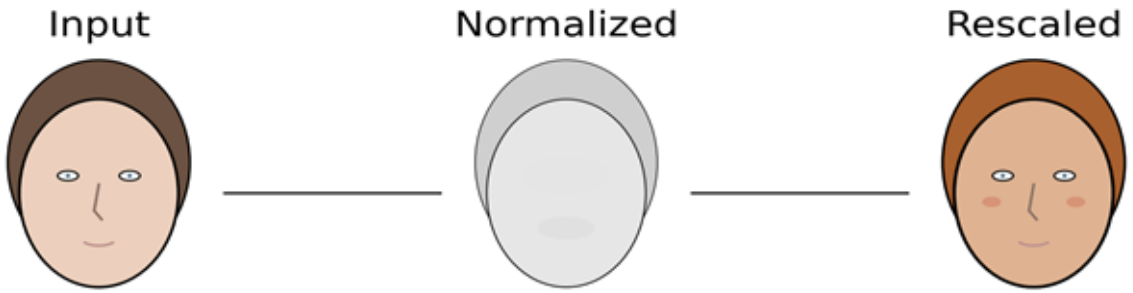


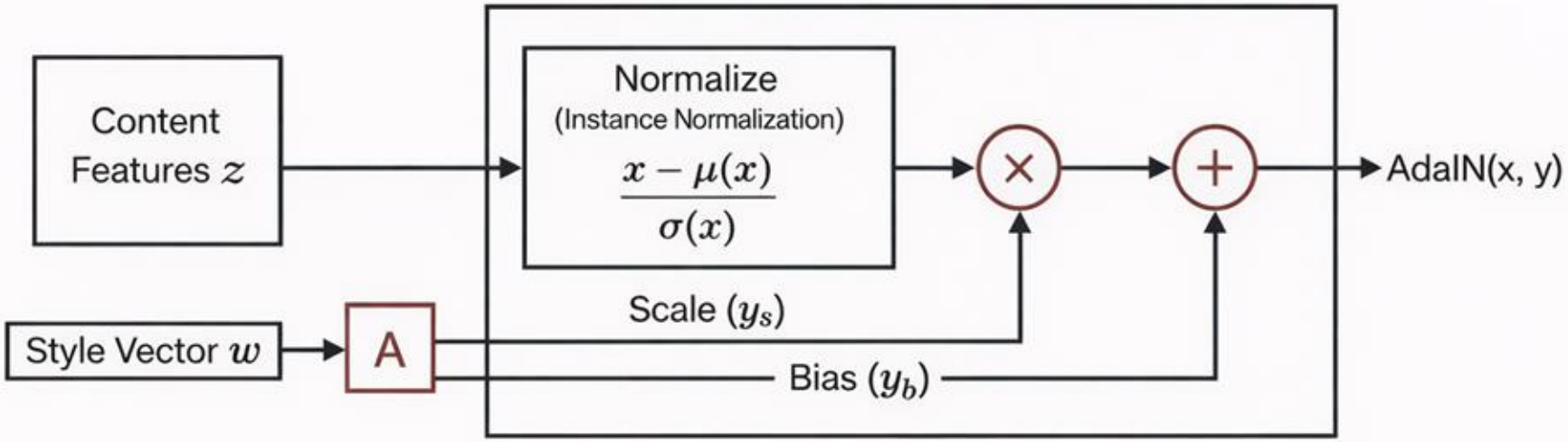
Mapping Network: Transforms $z \rightarrow w$. The intermediate space $\mathcal{W}$ allows factors of variation (pose, lighting, gender) to be linear and disentangled, as it does not need to follow a fixed Gaussian distribution.

Slide credit: https://youtu.be/G-Ed5wbKiU8?si=zmQUK6NFzdUUvN-M

Style Injection via AdaIN

Adaptive Instance Normalization





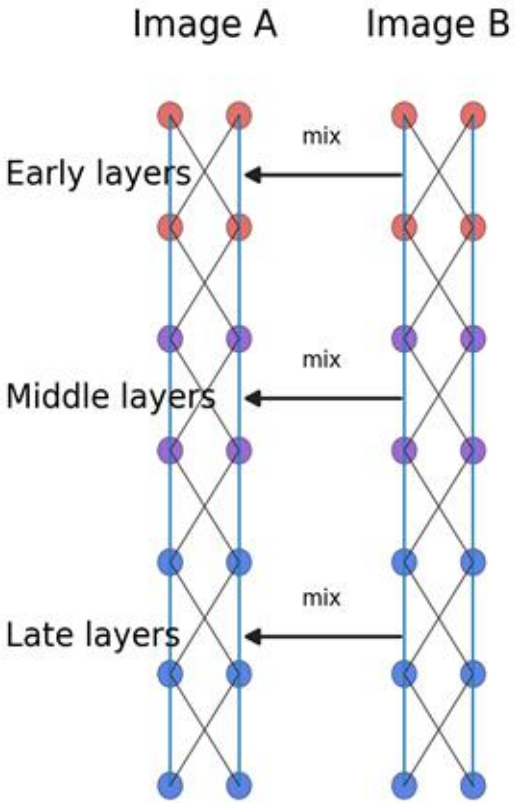
$$\text{AdaIN}(x, y) = y_s \frac{x - \mu(x)}{\sigma(x)} + y_b$$

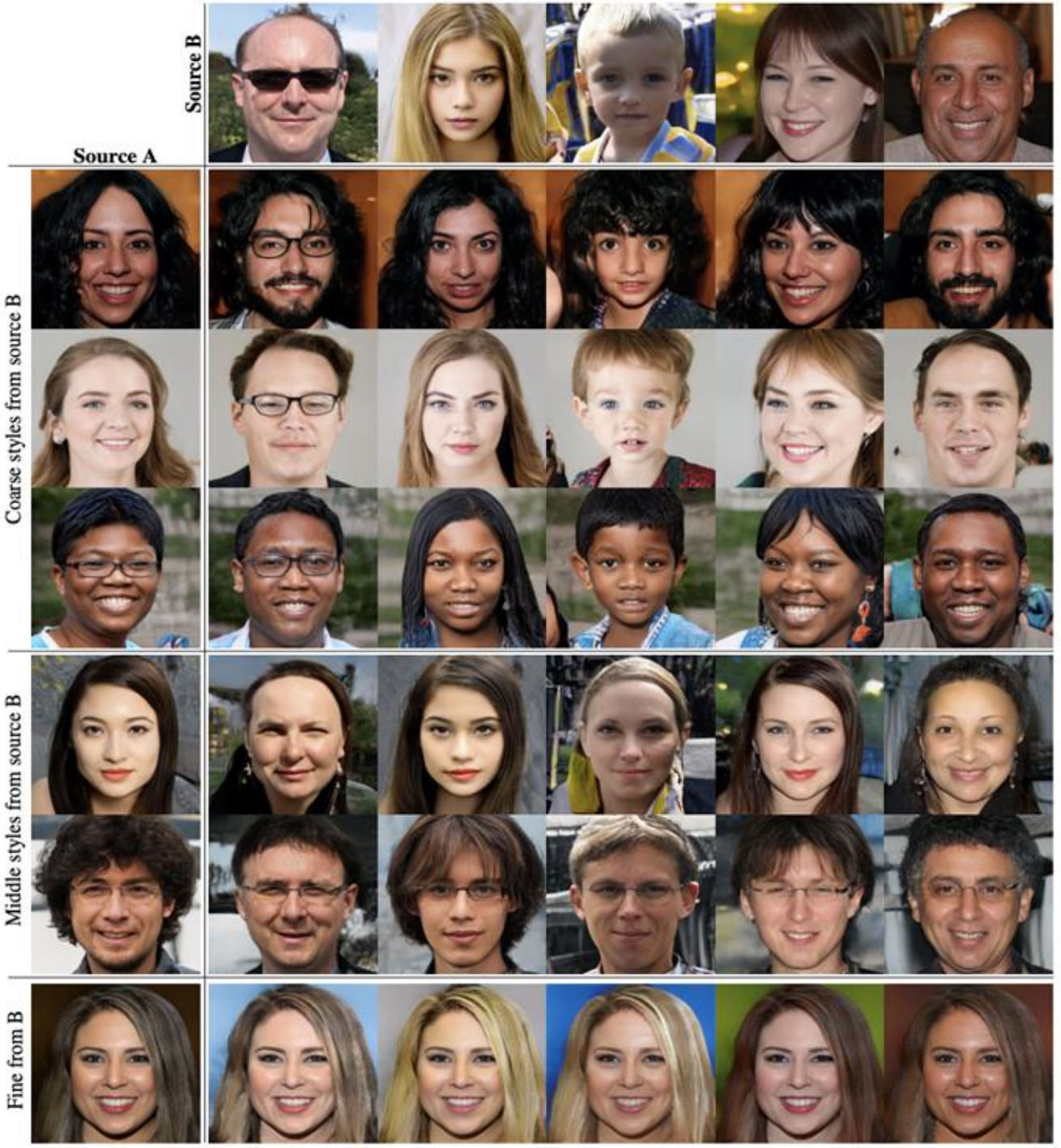
Coarse Styles (4x4 - 8x8):
Pose, Face Shape

Middle Styles (16x16 - 32x32):
Eyes, Facial Features

Fine Styles (64x64+):
Color Scheme, Micro-texture

Slide credit: https://youtu.be/G-Ed5wbKiU8?si=zmQUK6NFzdUUvN-M



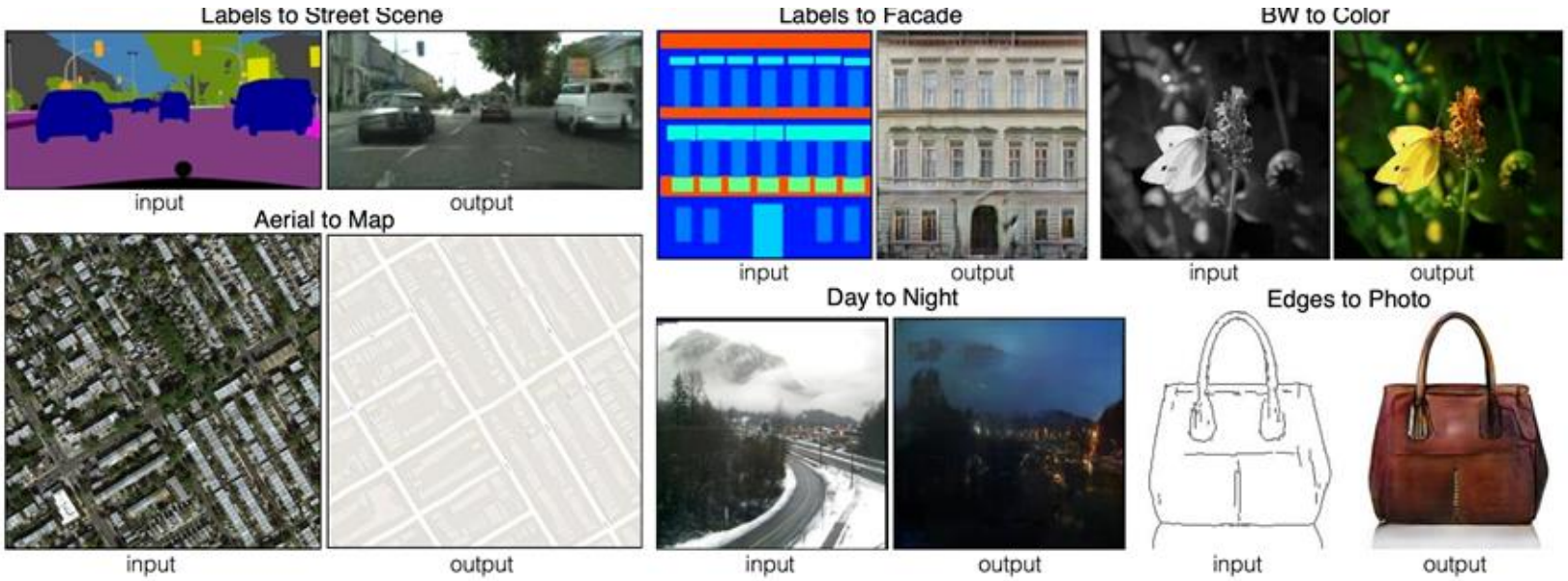


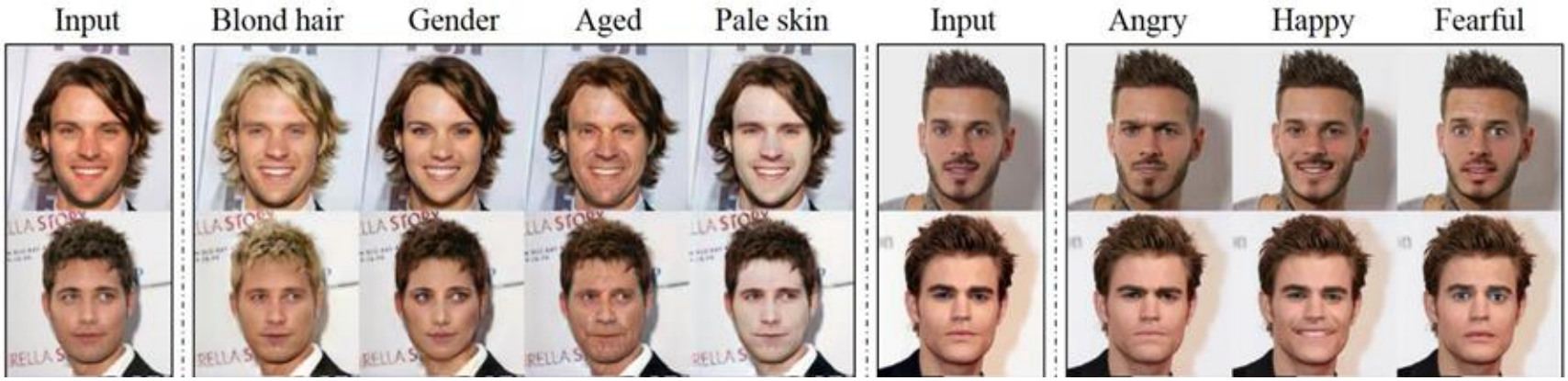
Karras et al. A Style-Based Generator Archi

Improvement and Application of GANs

- Loss functions of GANs
  - LSGAN
  - WGAN (GP)
- Architecture of GANs
  - Conditional GANs
  - Progressive GANs
  - StyleGAN
- Image-to-Image Translation GANs
- Text-to-Image GANs

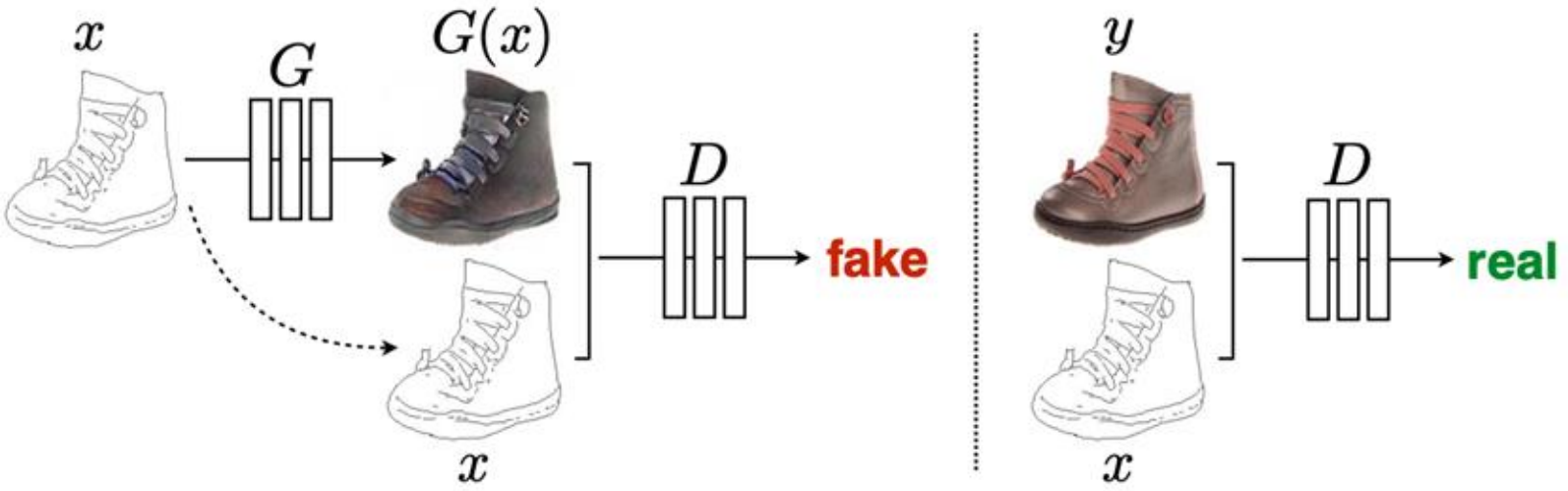
Image-to-Image Translation





Paired Translation: Pix2Pix

Learning direct mappings with paired supervision.



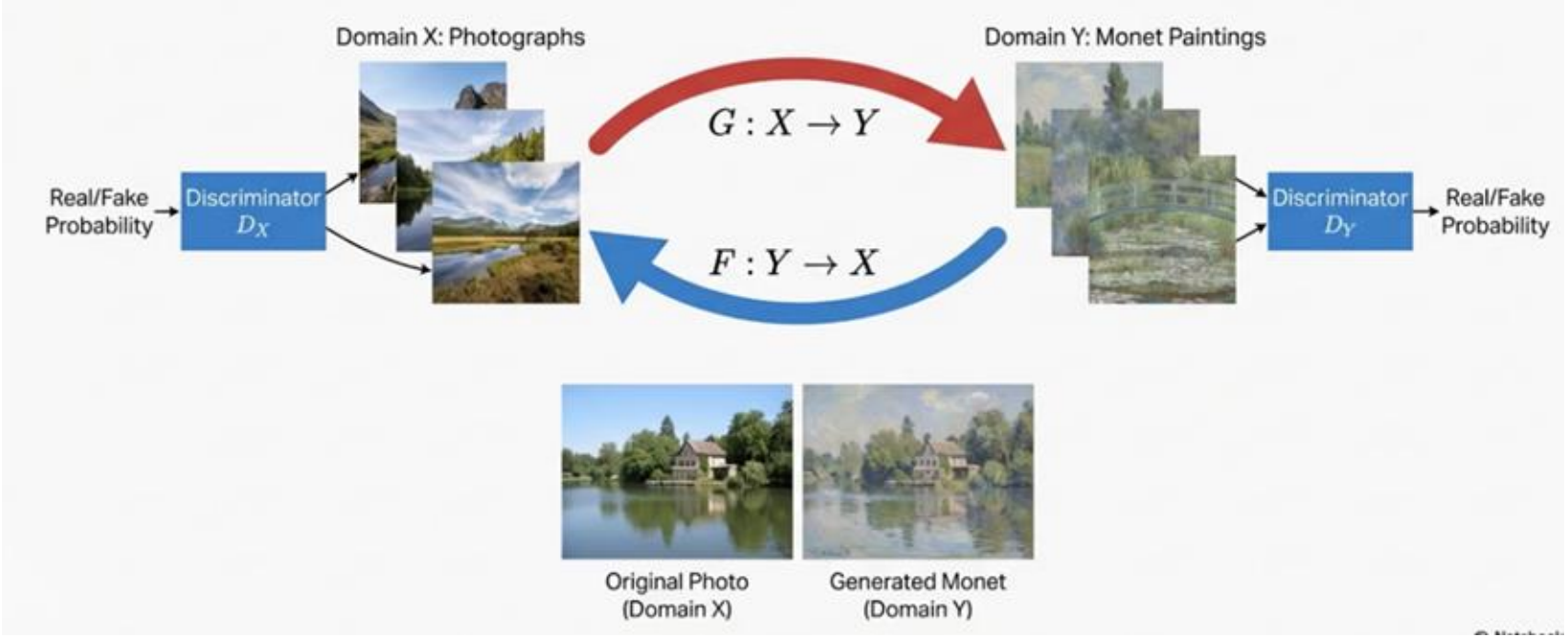
- A conditional GAN, but the condition is source-domain
- PatchGAN discriminator
  - Instead of predicting single real/fake classification
  - Predicting real/fake at feature elements from feature maps

Isola et al. Image-to-Image Translation with Conditional Adversarial Network. 2018.

Unpaired Translation: CycleGAN

Learning mappings without ground truth pairs.

- When we change styles, we typically don't have paired data (with the same data in both styles)



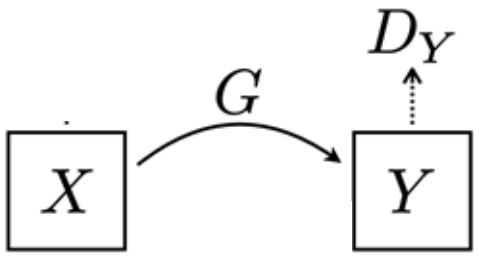
- How can we train image-to-image translation GANs without paired data?

Zhu et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. 2020.

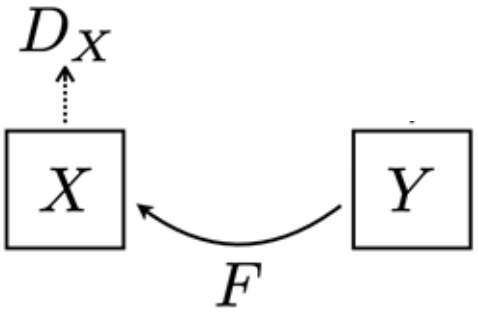
Unpaired Translation: CycleGAN

Learning mappings without ground truth pairs.

- Forward translation X -> Y, need paired data to train



- Backward translation Y -> X, need paired data to train

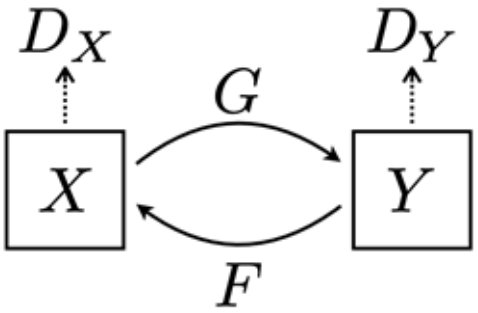


Zhu et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. 2020.

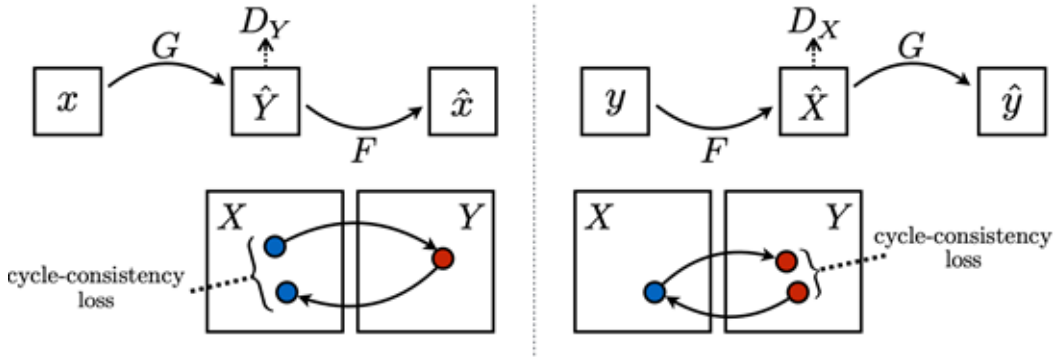
Unpaired Translation: CycleGAN

Learning mappings without ground truth pairs.

- Bi-directional translation: forward and then backward



- Enforcing cycle consistency



Zhu et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. 2020.

Multi-Domain Translation: StarGAN

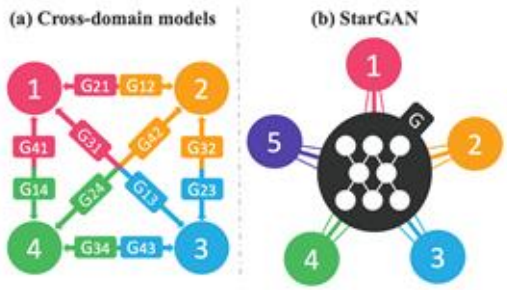
One model for many domain translations.

- CycleGAN was designed for two domains
- What if we have multiple domains?

Multi-Domain Translation: StarGAN

One model for many domain translations.

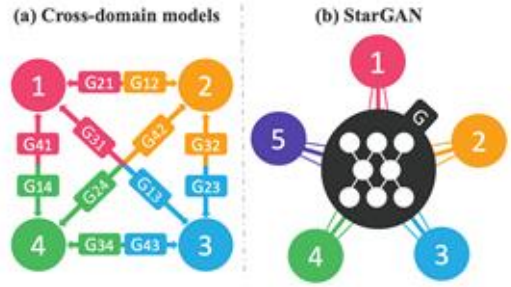
- CycleGAN was designed for two domains
- What if we have multiple domains?
  - Have generators and discriminators for each domain



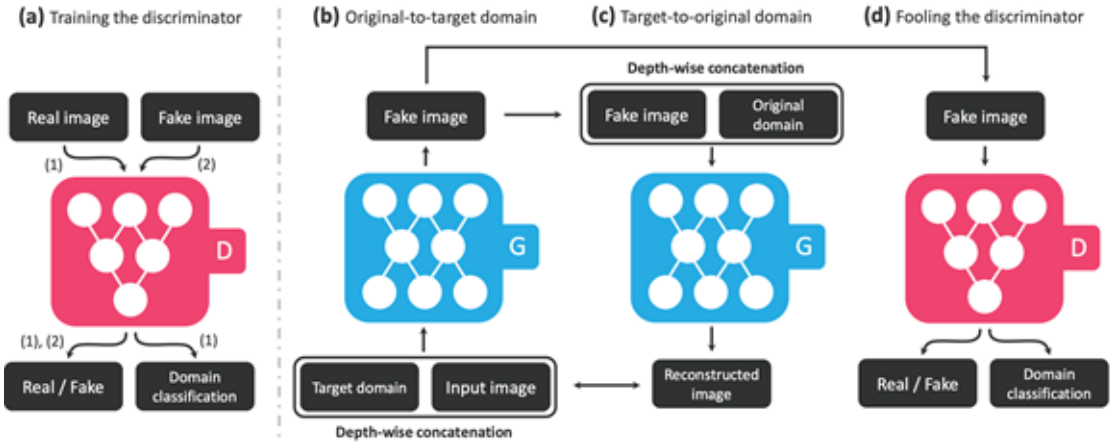
Multi-Domain Translation: StarGAN

One model for many domain translations.

- CycleGAN was designed for two domains
- What if we have multiple domains?
  - Have generators and discriminators for each domain



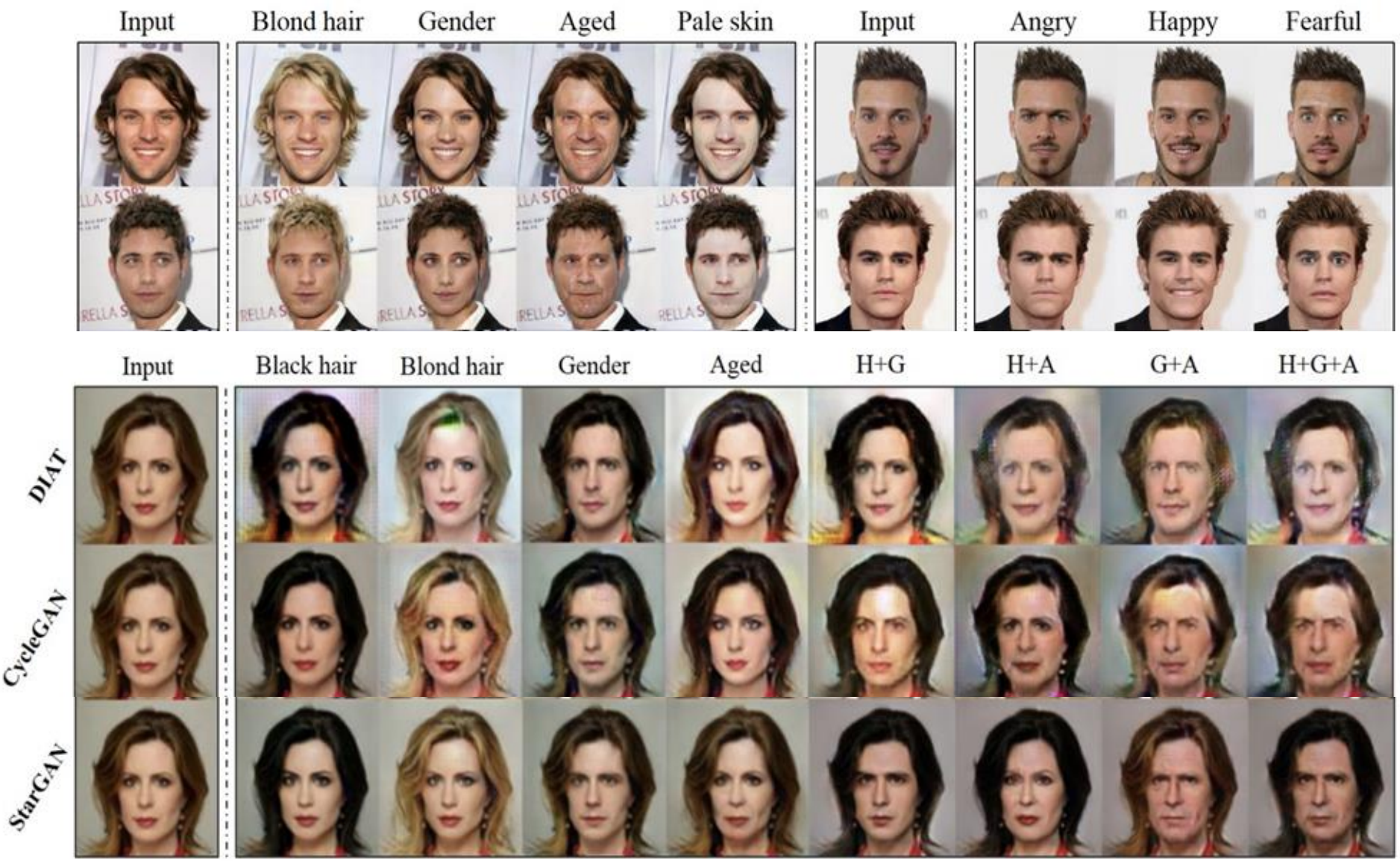
  - During training, sample two domains and do cycleGAN!



Choi et al. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. 2018.

Multi-Domain Translation: StarGAN

One model for many domain translations.



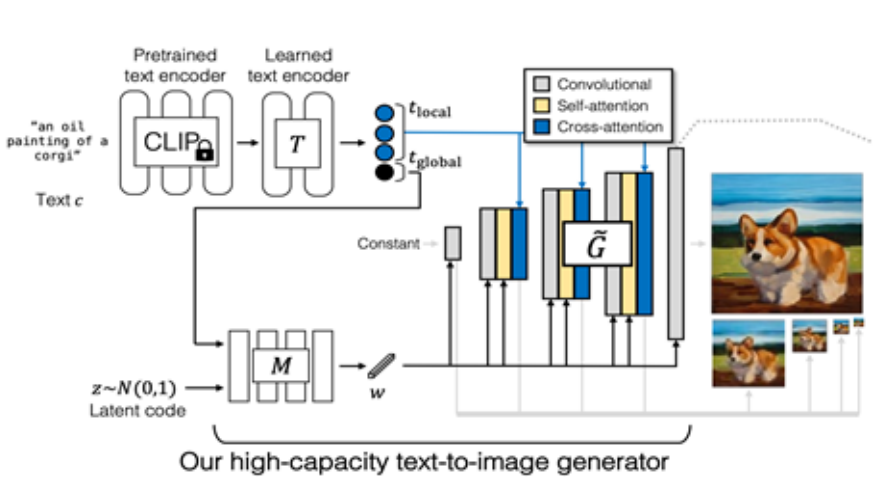
Choi et al. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. 2018.

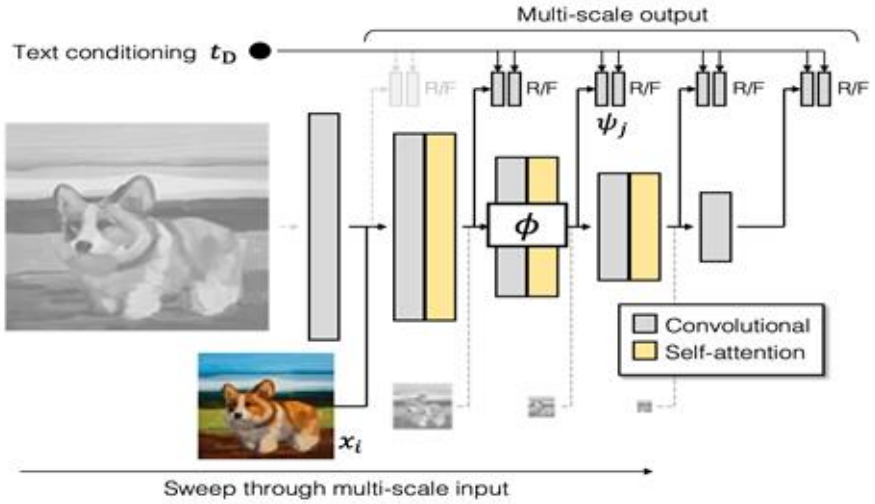
Improvement and Application of GANs

- Loss functions of GANs
  - LSGAN
  - WGAN (GP)
- Architecture of GANs
  - Conditional GANs
  - Progressive GANs
  - StyleGAN
- Image-to-Image Translation GANs
- Text-to-Image GANs

Text-to-Image GANs

- Generator and Discriminator design





Text-to-Image GANs

small white orange black and yellow bird with long grey tarsus and small black beak.

a small yellow bird with thin orange feet and dark yellow wings.

this colorful bird has a black head and throat, black wings with white detail, and a bright yellow body.

A white biplane on grassy area next to buildings.

flat screen television on top of an old tv console

A dark wooden table with a wicker place mat, a banana, spoon and a white bowl filled with food.

In Closing

- GANs were the first breakthrough in high-quality synthesis/generation
  - The amazing power of a DILLAF loss…
- Recent progress focuses more on optimization rather than architecture
  - generative trilemma: trade-offs between speed, quality, and diversity
- The idea of a *discriminator* is very powerful and is useful outside of the purely generation problem