Generative Aversarial Networks (Based on slides from Ben Striner)

11785 Deep Learning

Fall 2023

Topics for the week

- Transformers
- GNNs
- VAEs
- GANs
- Connecting the dots

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?

Discriminative vs Generative Models

Given a distribution of inputs X and labels Y.

Discriminative models

- Discriminative models learn conditional distribution P(Y | X)

Generative models

- Generative models learn the
Generative models learn the joint distribution P(Y, X)

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.

Generative models

- Generative models learn the Generative models learn the joint distribution P(Y, X)
- Learns actual probability distribution of data.

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.

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.

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

- 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

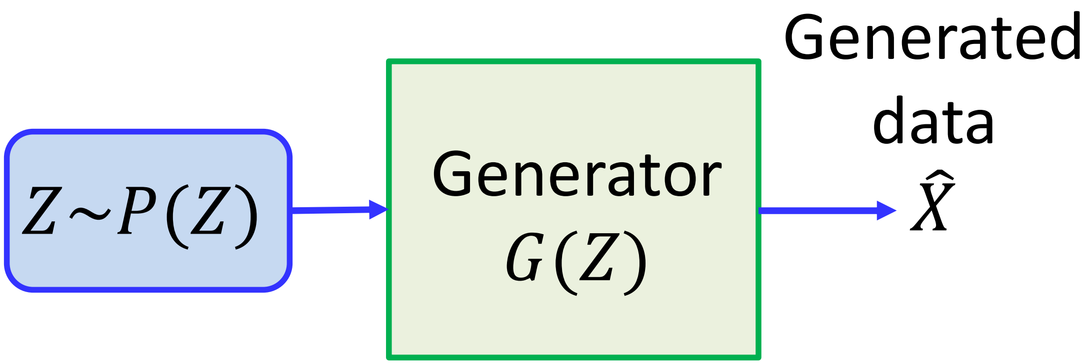
- What is the difference between Discriminative models vs. Generative models
  - **Discriminative models model the decision boundary between classes, whereas Generative models model class distributions**
  - Generative models model the decision boundary between classes, whereas Discriminative models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

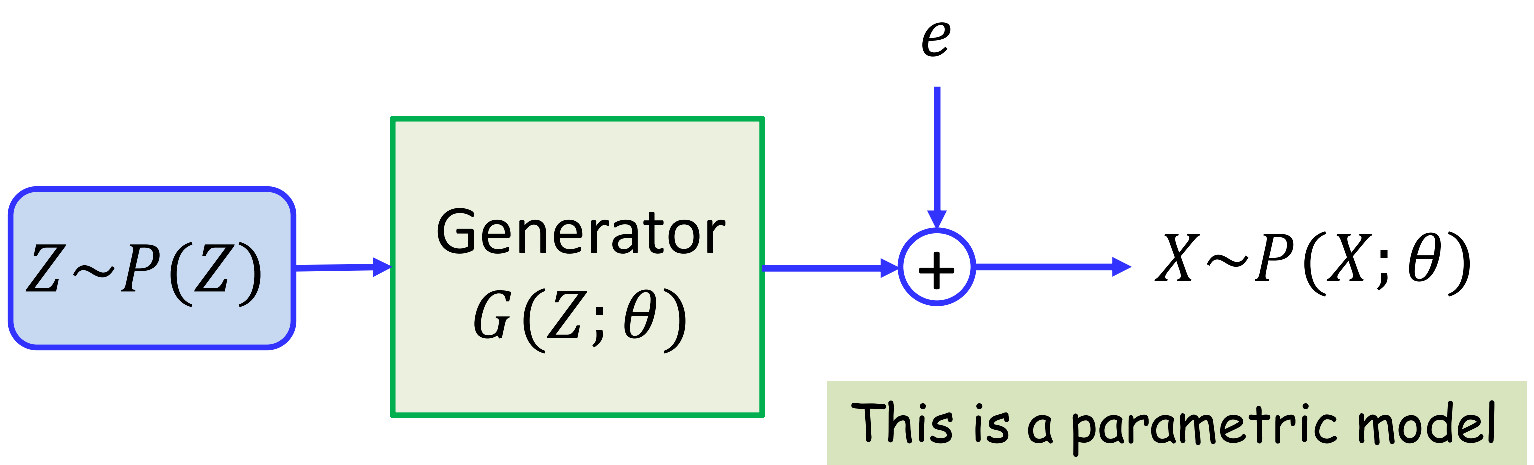
- From a large collection of images of faces, can a network learn to *generate* new portrait
  - Generate samples from the distribution of "face" images
    - How do we even characterize this distribution?

What we have seen: VAE



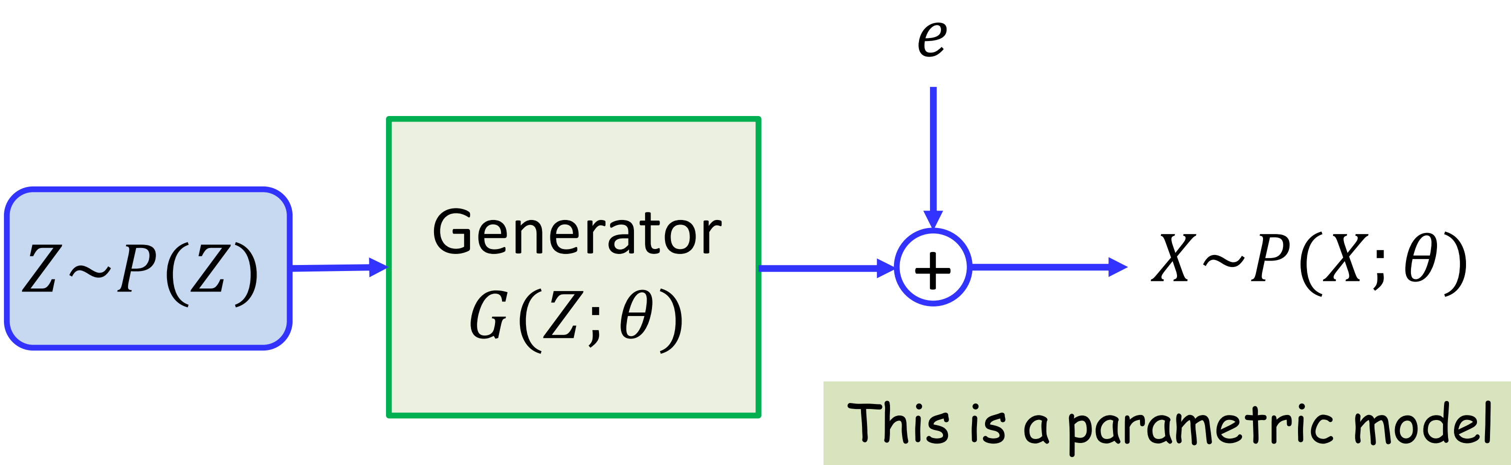
- Generator is a decoder of a VAE

What we have seen: VAE



- Generator is a decoder of a VAE

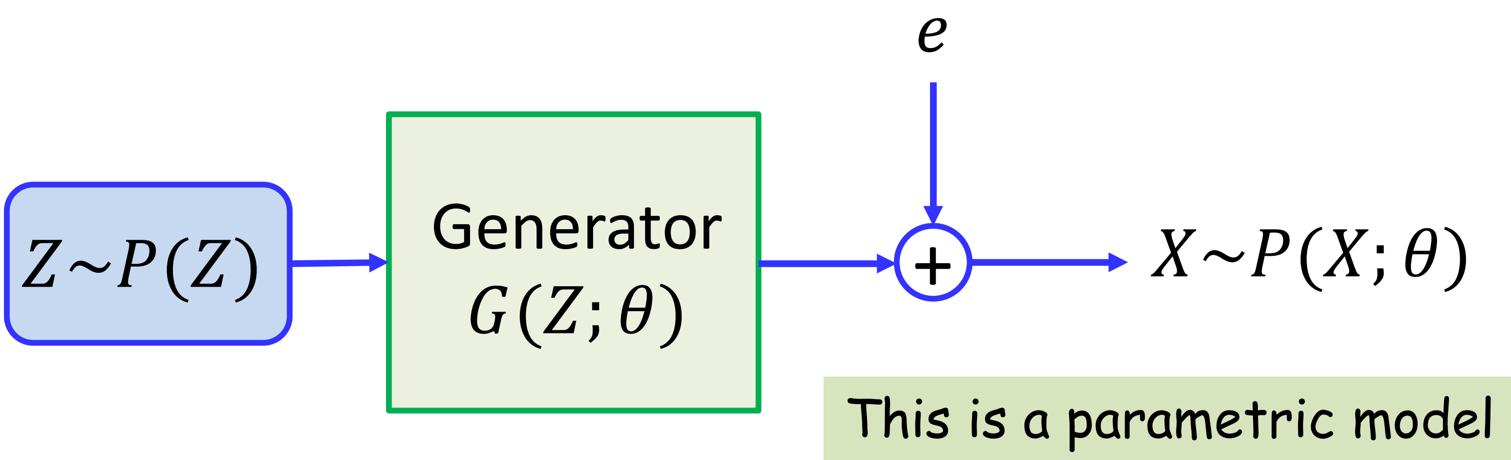
What we have seen: VAE



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

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

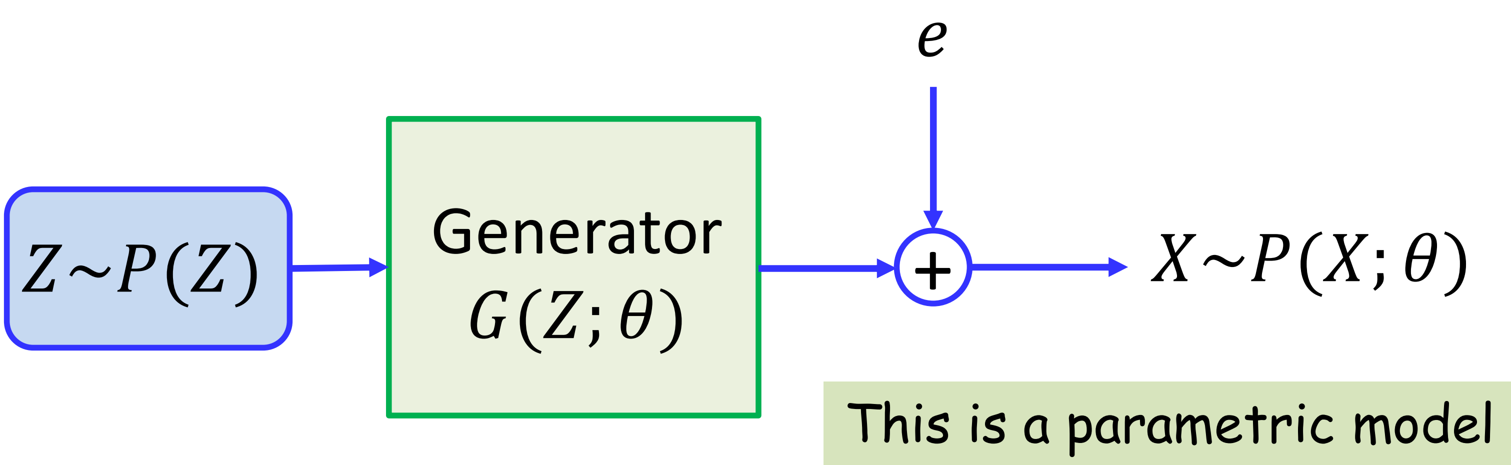
What we have seen: VAE



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

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

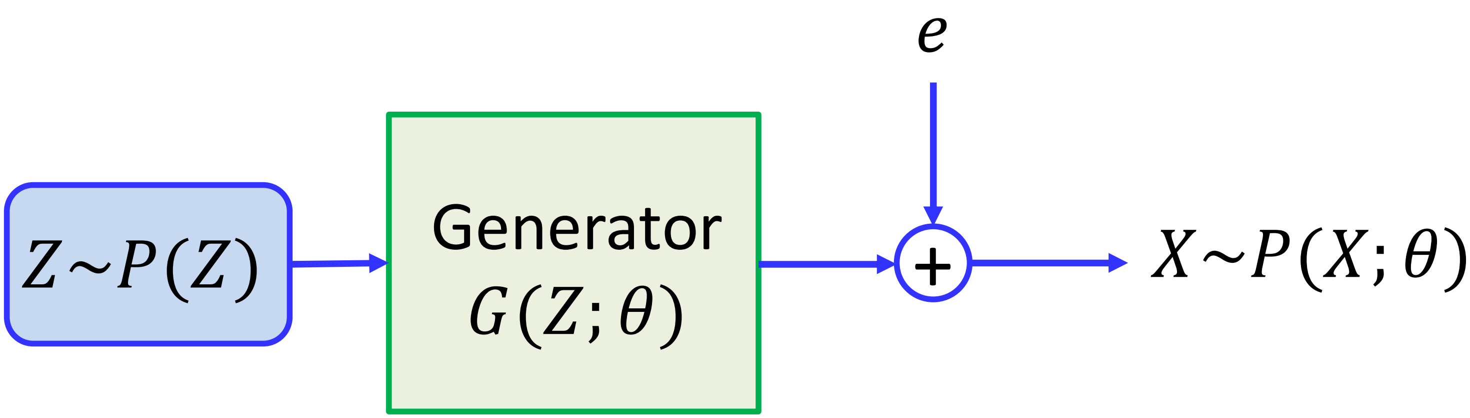
What we have seen: VAE



- 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

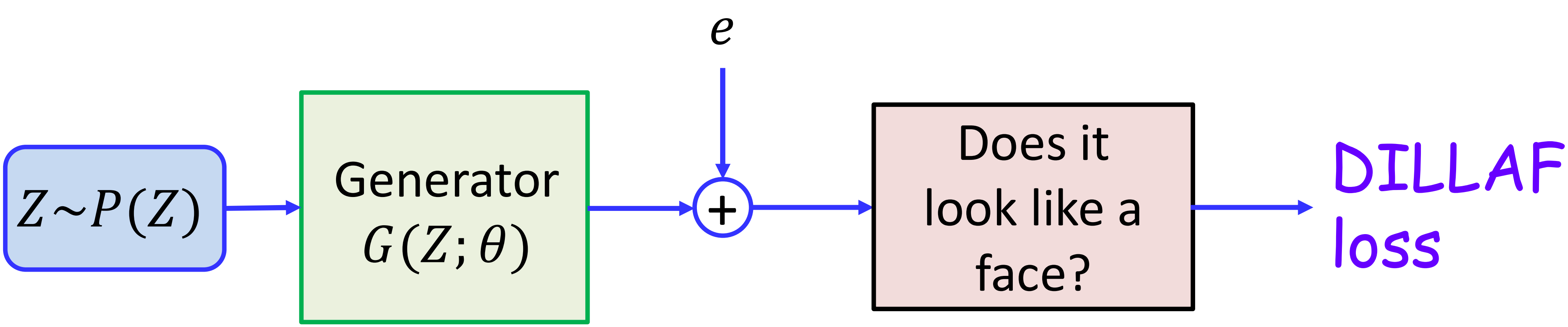
$$\theta^* = \arg\min_\theta \quad -\log P(X;\theta)$$

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

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

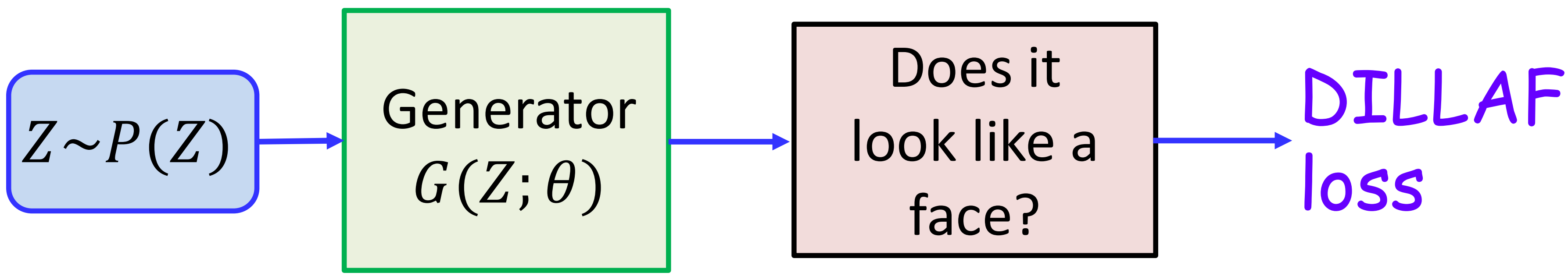
- Why would likelihood maximization not result in a model that produces more face-like outputs (for a face-generating VAE)?
  - The model can maximize the likelihood of training data without any assurance about what other (non-training) samples look like
  - The model is more likely to run into poor 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

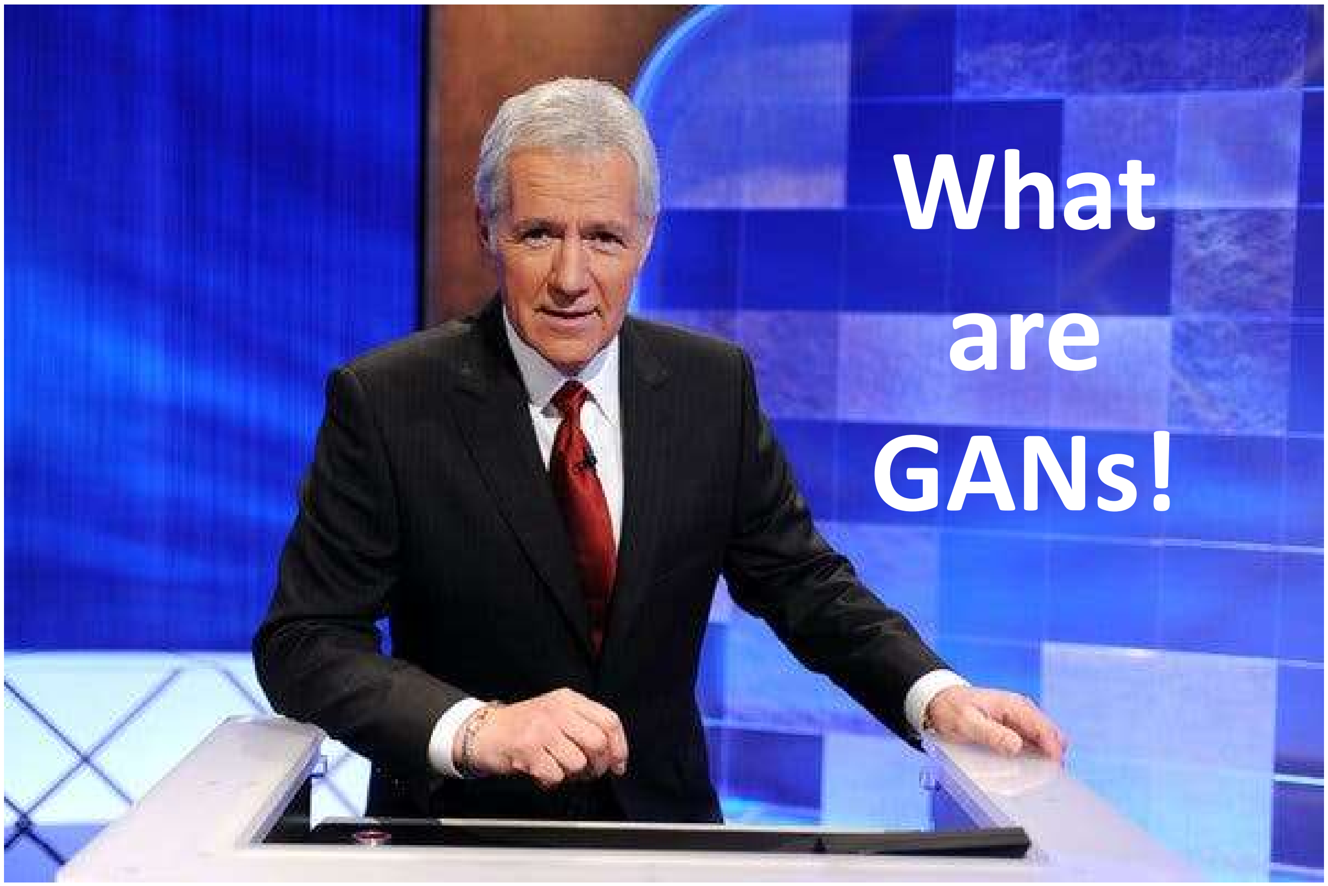
Poll 2

- 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

Generative Adversarial Networks

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

What are GANs

Generative Adversarial Networks

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 **Adversarial** **Networks**

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

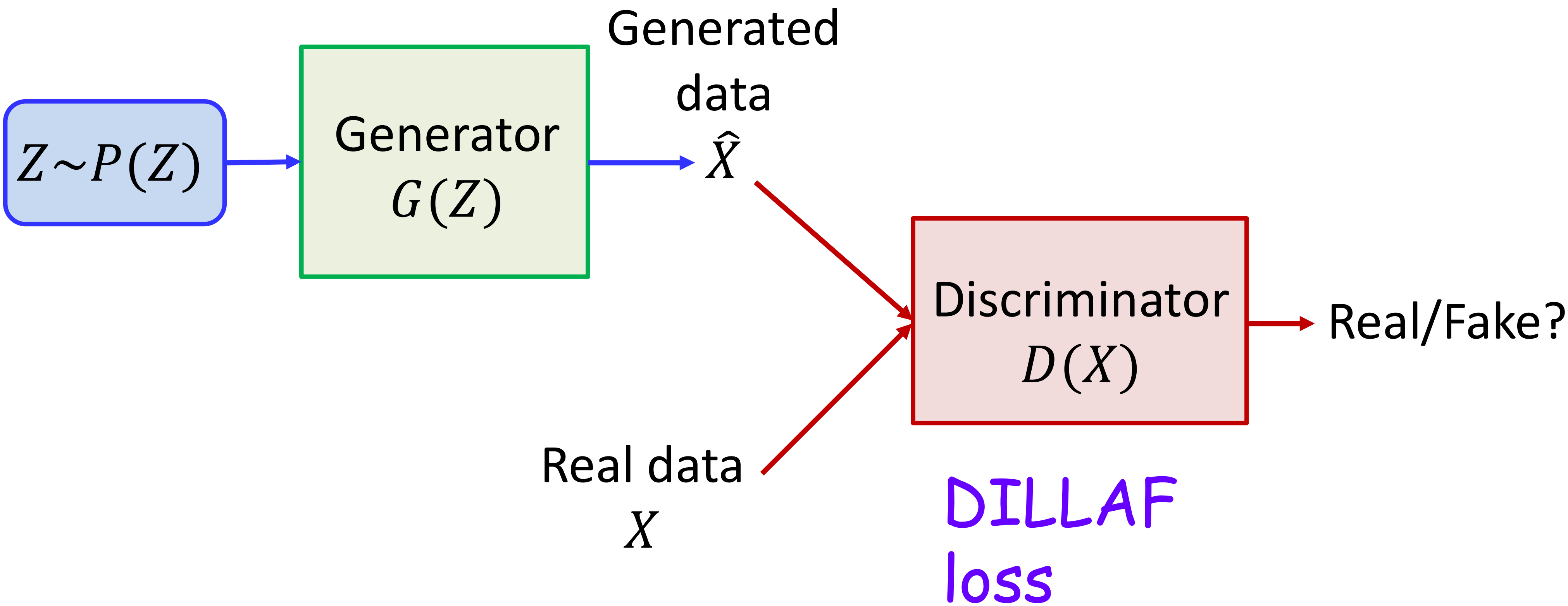
Neural Networks

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

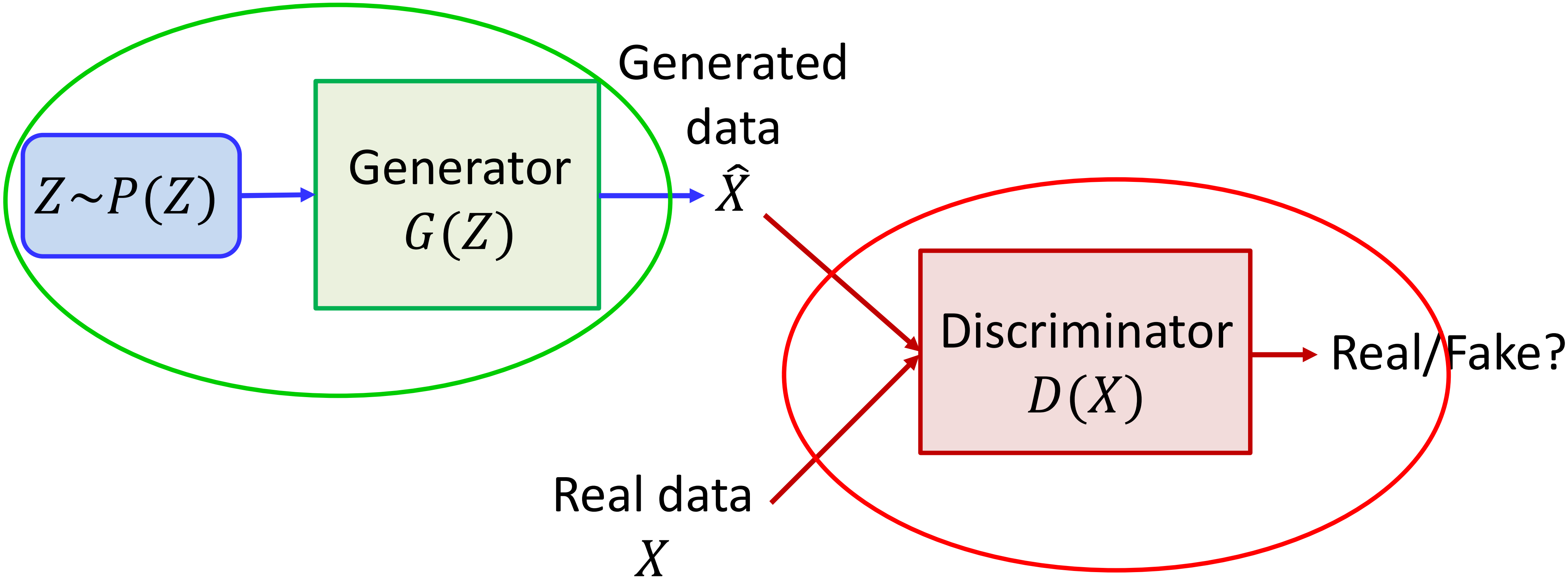
Generative Adversarial Networks

- 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!!

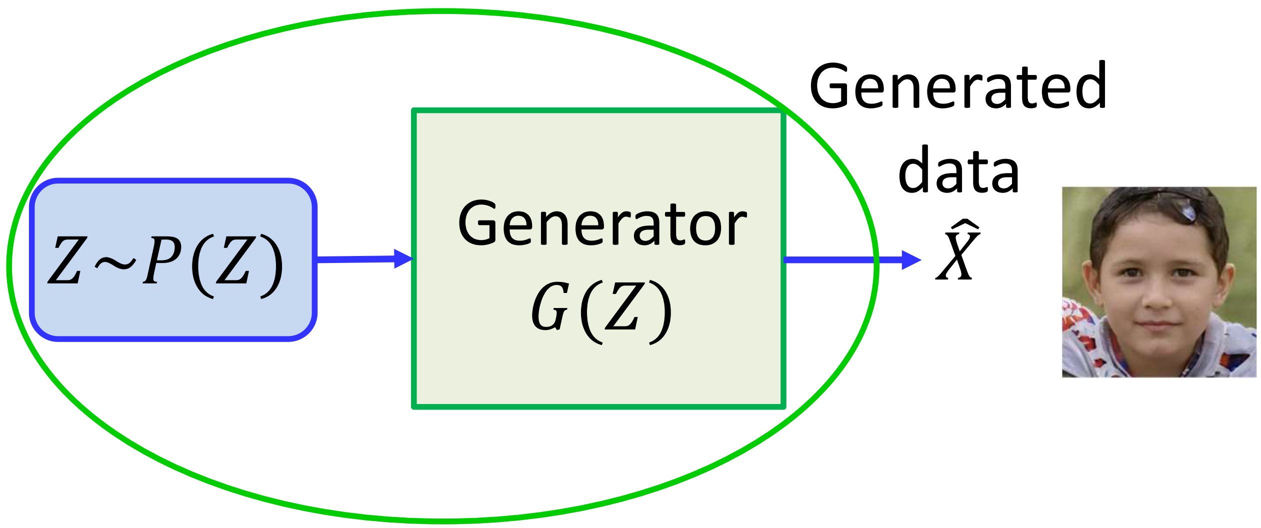
What are GANs?



What are GANs?



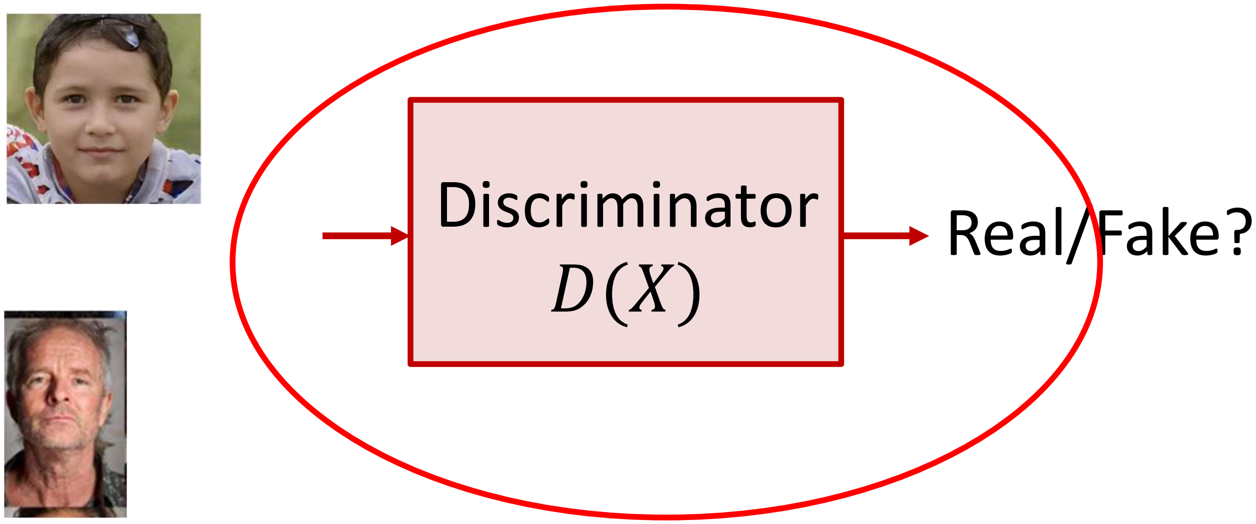
The Generator



- **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

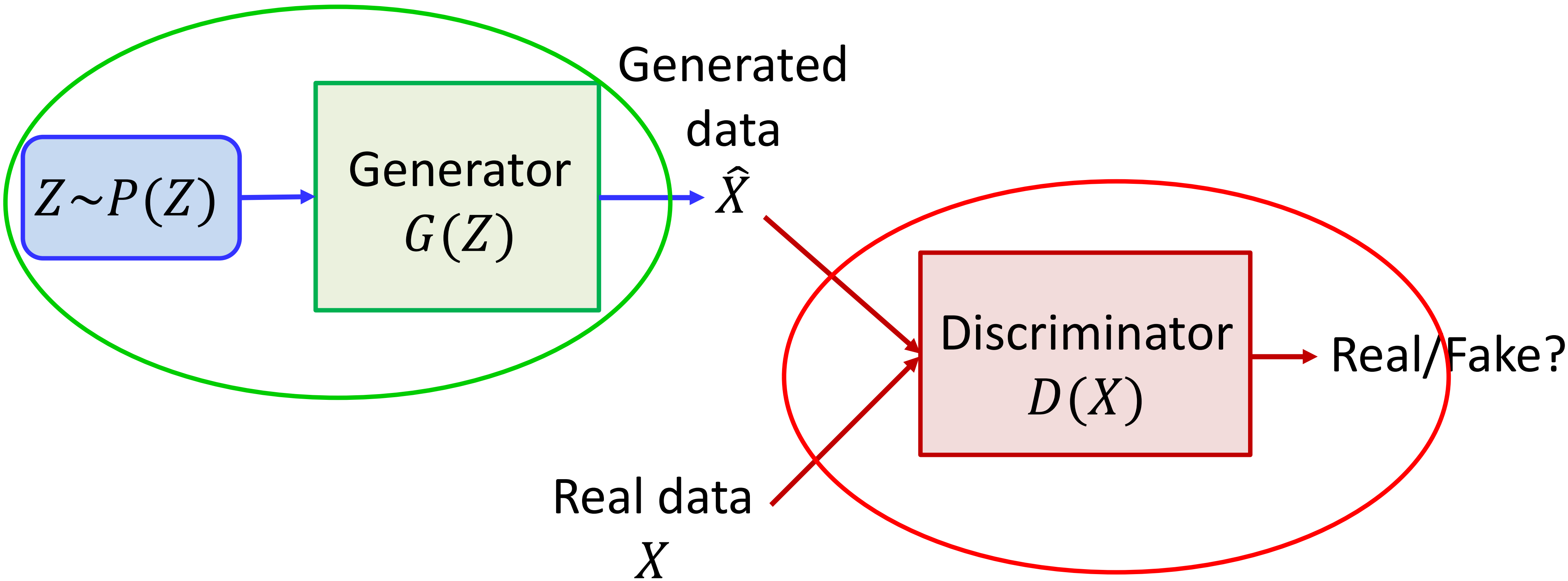
- **Goal**: generated distribution, $P_G(X)$ matches the true data distribution $P_X(X)$
  - $P_G(X)$ is the more "memorable" notation for $P_{\hat{X}}(X)$, the probability that a generated sample $\hat{X}$ takes the value X

The Discriminator



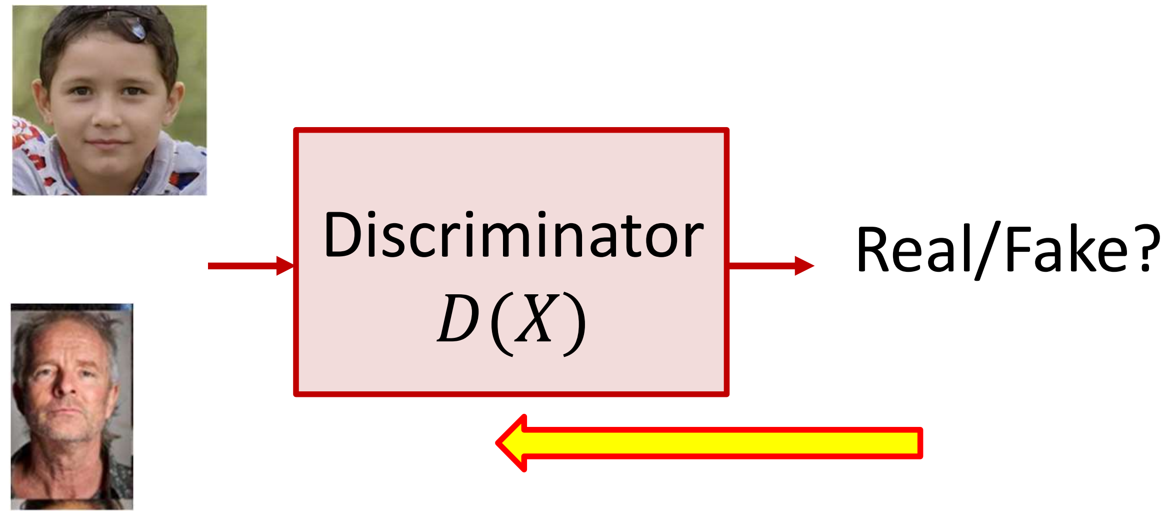
- Discriminator $D(X)$ is trained to tell the difference between real and generated (fake) data
  - Specifically, data produced by the generator
  - If a perfect discriminator is fooled, the generated data cannot be distinguished from real data

Training a GAN



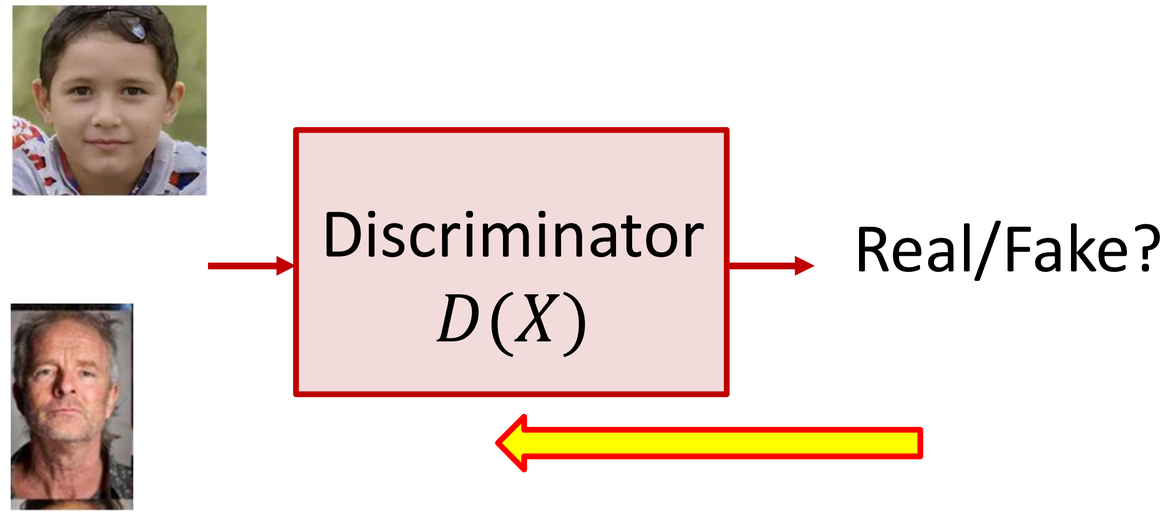
- Both, the generator and discriminator must be trained

Training the discriminator



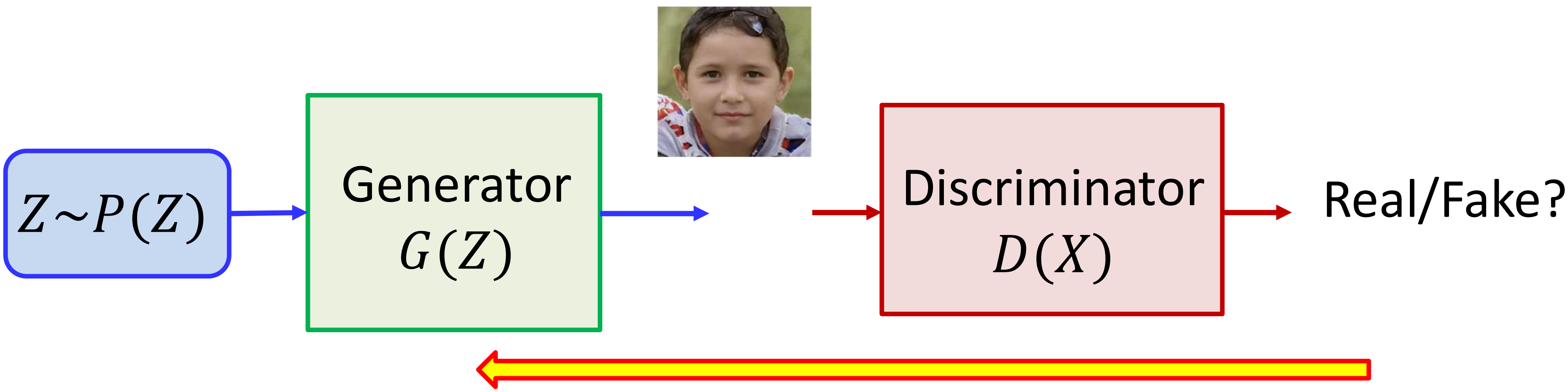
- **Training the discriminator:**
  - The discriminator is provided training examples of real and synthetic faces
  - The discriminator is trained to minimize its classification loss
    - Minimize error between actual and predicted labels
  - Discriminator parameters are trained such that
    - $D(X)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

Training the discriminator



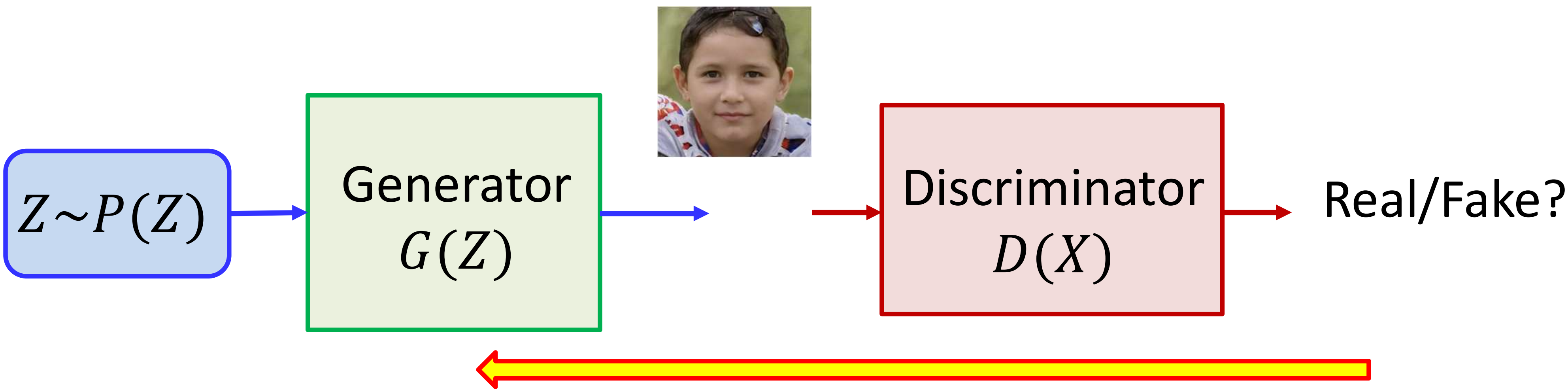
- **Training the discriminator:**
  - The discriminator is provided training examples of real and synthetic faces
  - The discriminator is trained to minimize its classification loss
    - Minimize error between actual and predicted labels
  - Discriminator parameters are trained such that
    - Maximize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

Training the generator



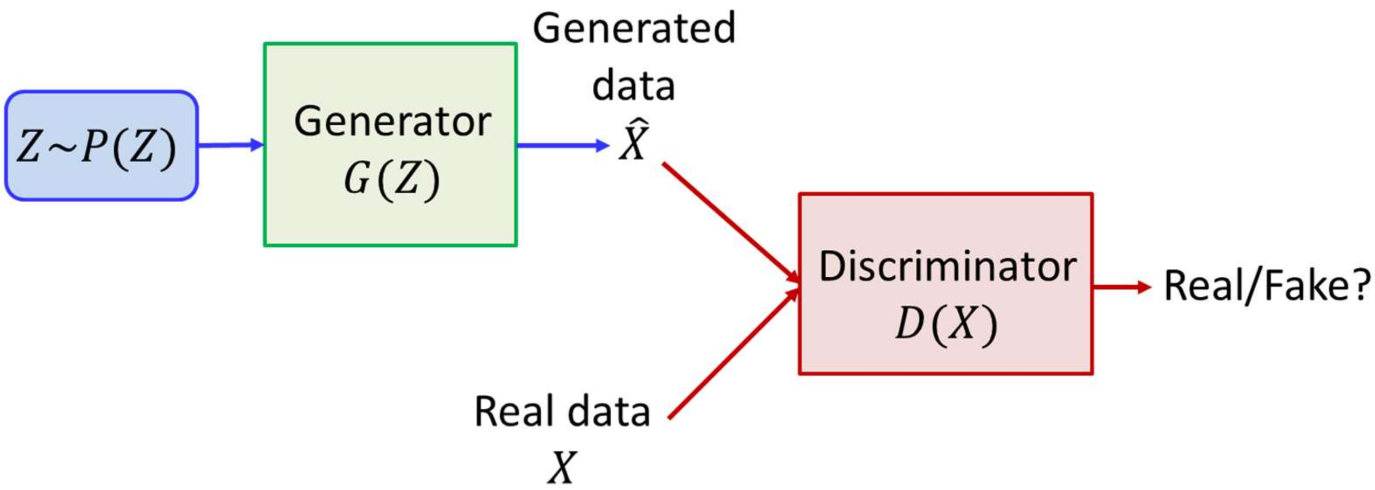
- **Training the generator:**
  - The discriminator's loss is backpropagated to the generator
  - The generator is trained to *maximize* the discriminator loss
    - It is trained to "fool" the discriminator
  - Generator parameters are trained such that
    - $D(G(Z)) = 1$ (i.e. $1 - D(G(Z)) = 0$)

Training the generator



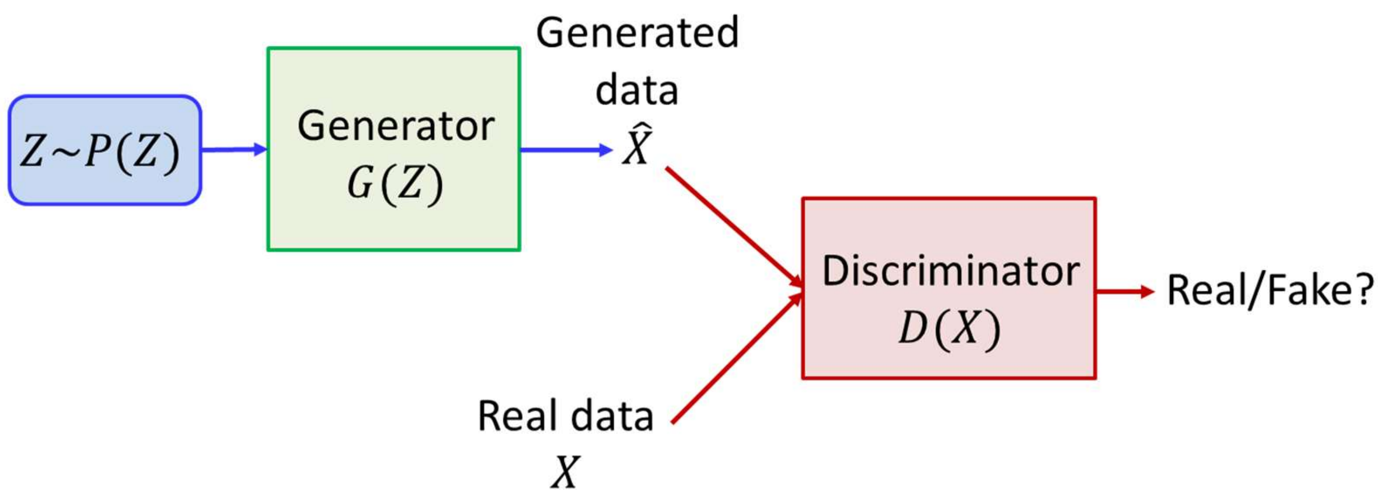
- **Training the generator:**
  - The discriminator's loss is backpropagated to the generator
  - The generator is trained to *maximize* the discriminator loss
    - It is trained to "fool" the discriminator
  - Generator parameters are trained such that
    - Minimize $\log(1 - D(G(Z)))$

The GAN formulation



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

The GAN formulation

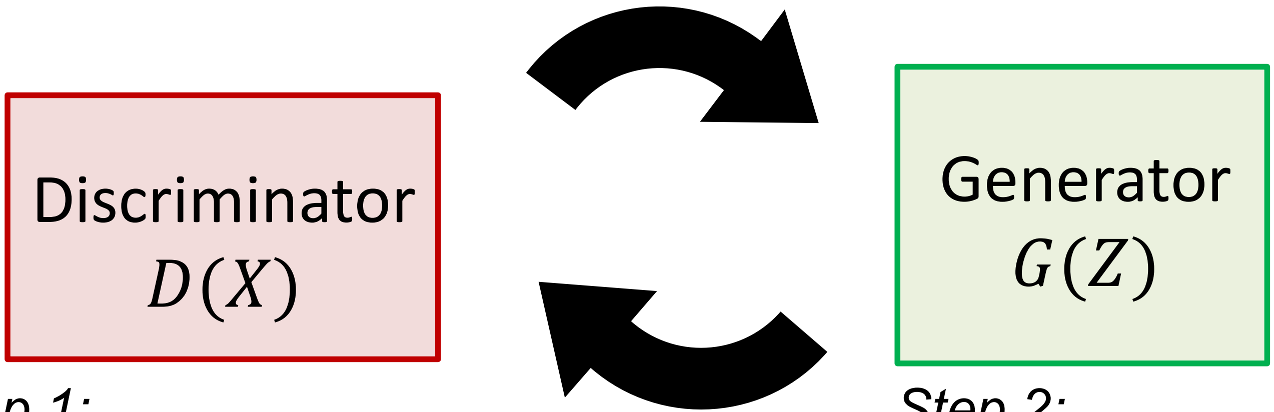


- The original GAN formulation is the following min-max optimization

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

- 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

- When training a GAN, which component must you train first
  - The discriminator
  - The generator

- Which component is updated more frequently
  - The discriminator
  - The generator

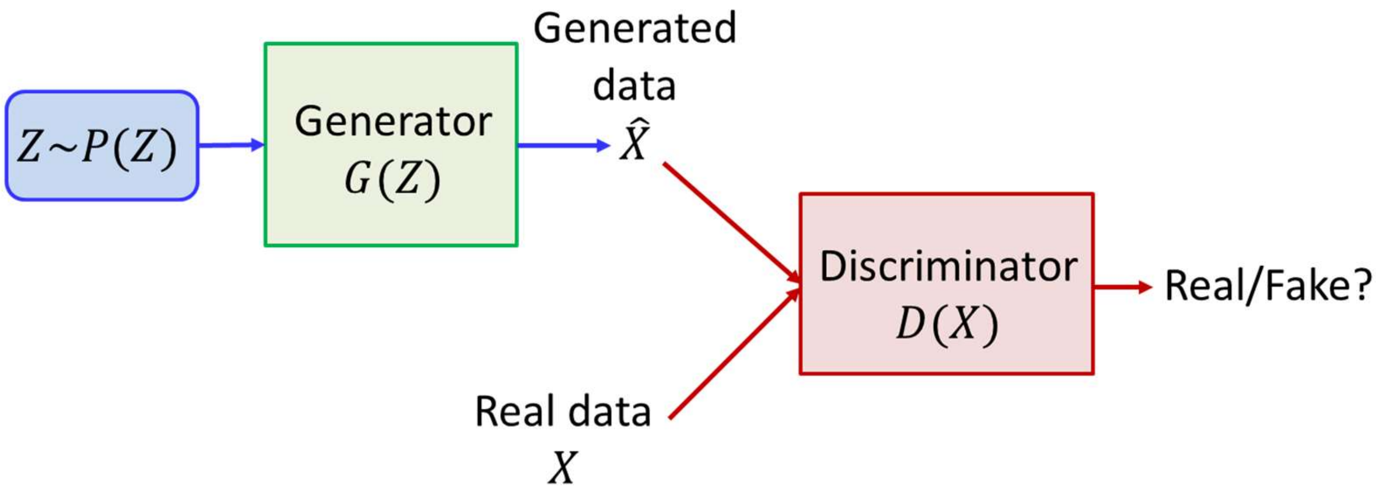
Poll 3

- When training a GAN, which component must you train first
  - **The discriminator**
  - The generator

- Which component is updated more frequently
  - **The discriminator**
  - The generator

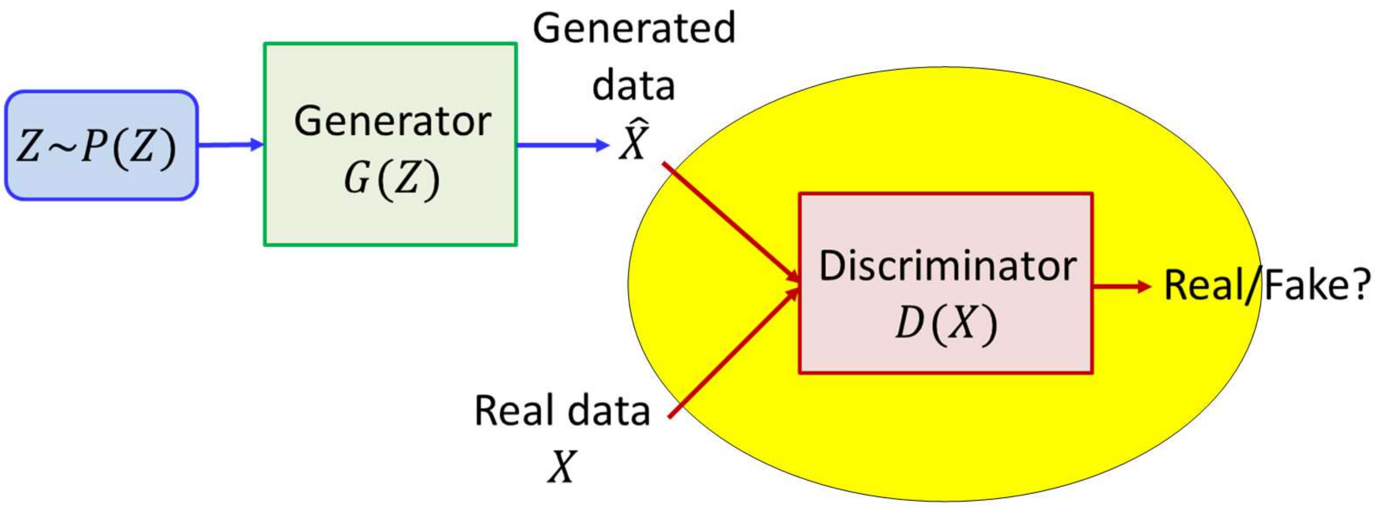
The discriminator is the (DILLAF) loss. Training the loss is more important, since the loss guides the training!

The GAN formulation



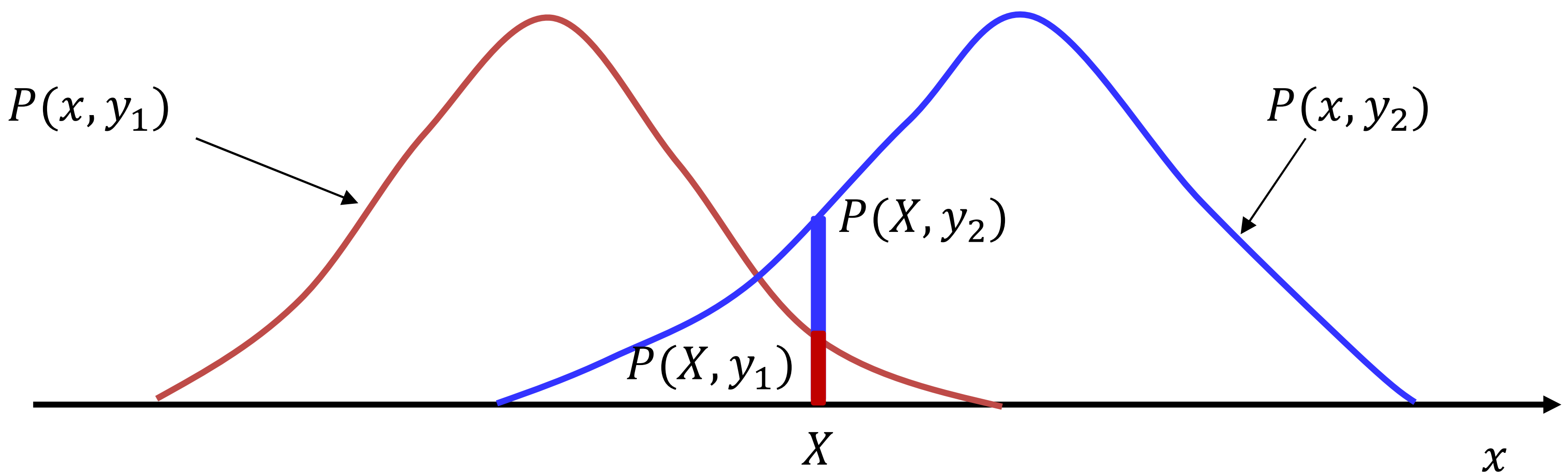
- So how does this behave when each component is optimized…

The GAN formulation



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

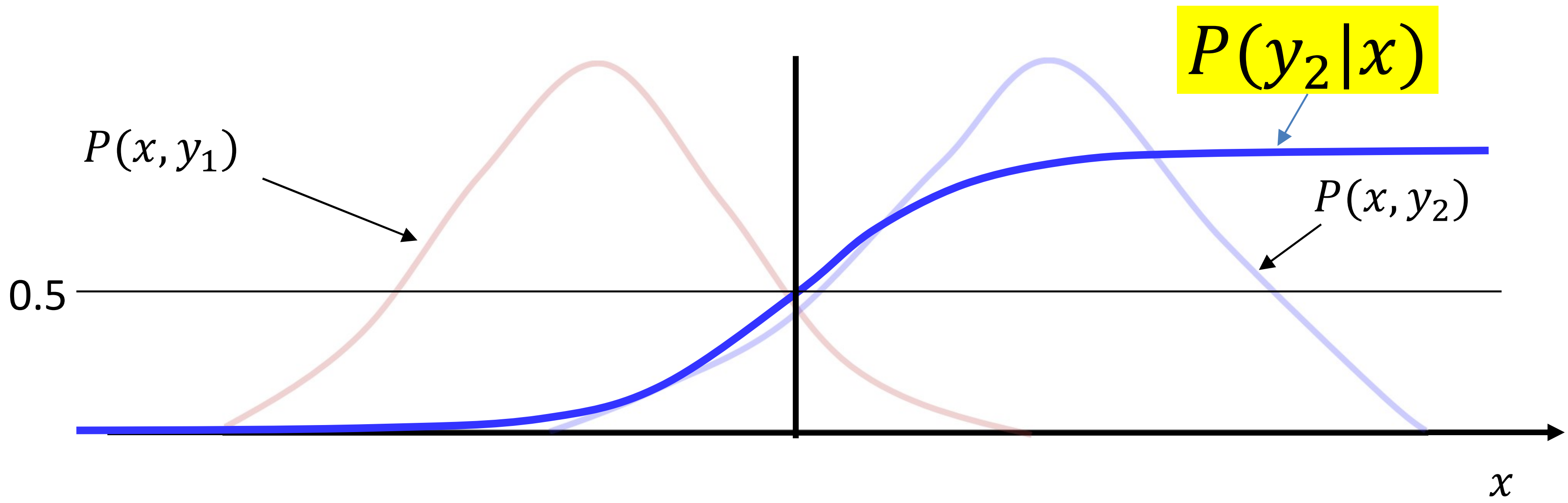
The perfect discriminator: Consider a binary classification problem



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

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

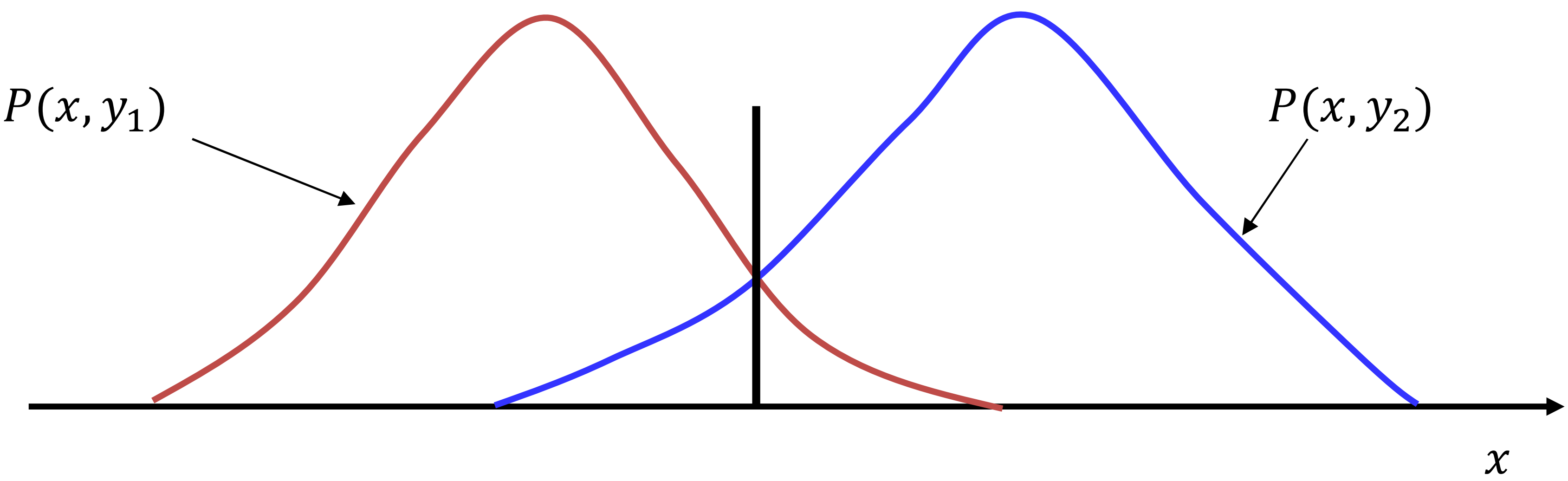
The perfect discriminator: Consider a binary classification problem



- 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

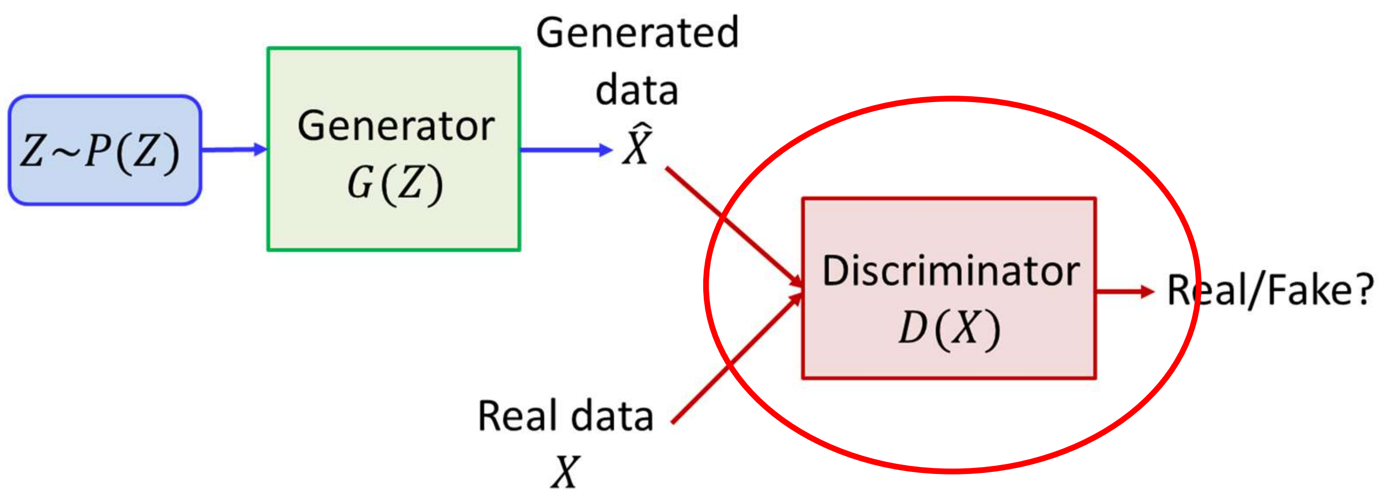


- The a posteriori probability of the classes for any instance $x = X$ is

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

- 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 $P(y_i|X)$

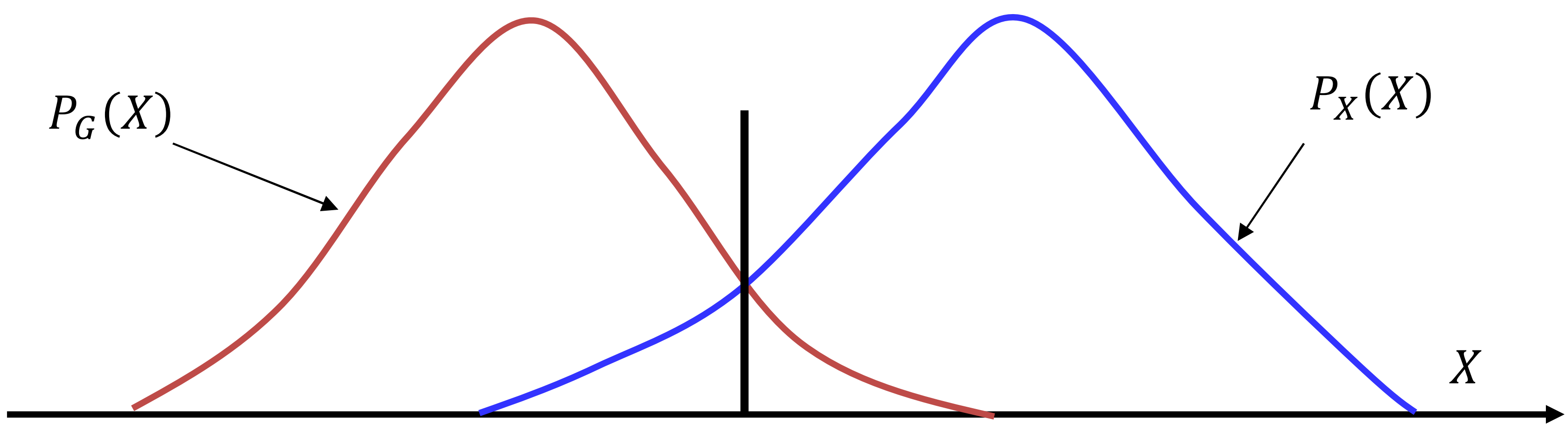
The optimal discriminator



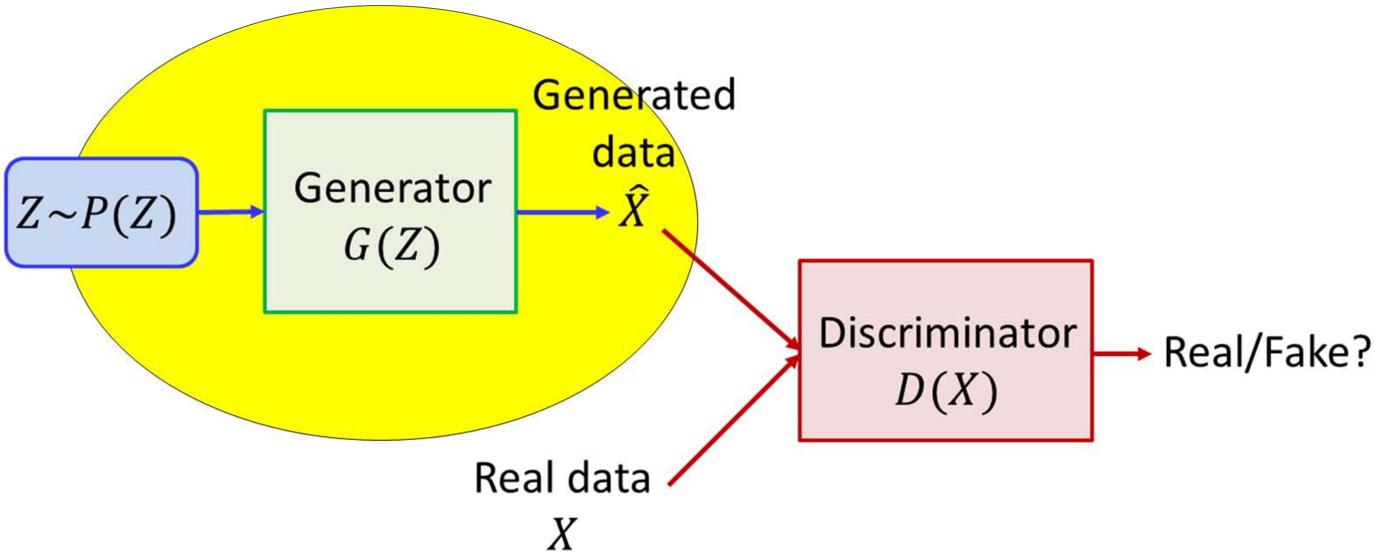
- The **optimal discriminator** would be a Bayesian classifier

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

  – Assuming uniform prior

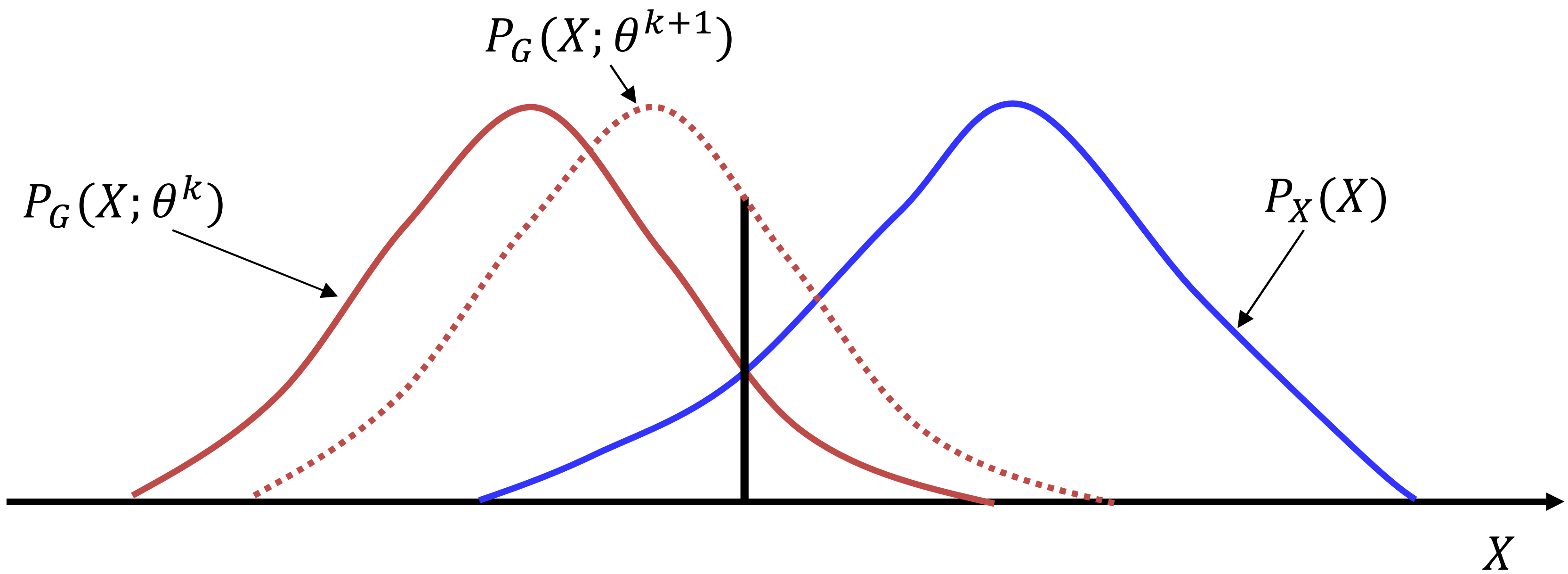


The GAN formulation



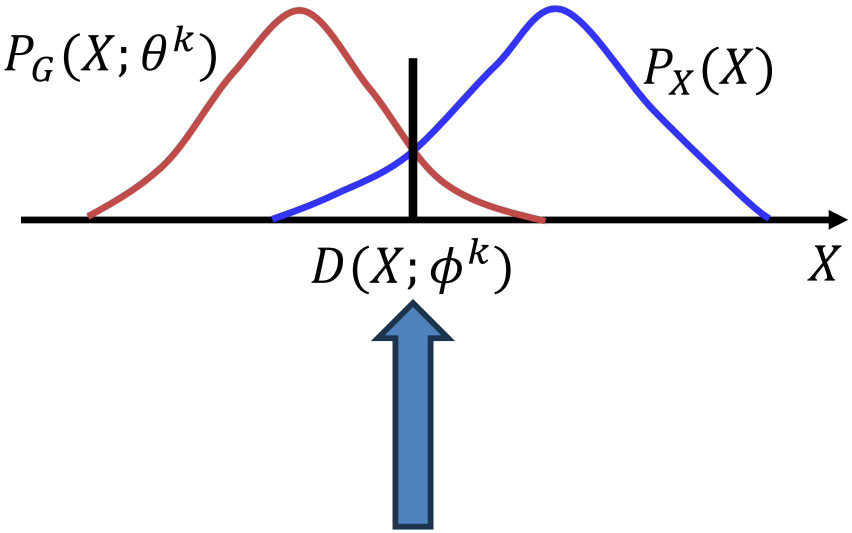
- So how does this behave when each component is optimized…

Updating the Generator: Fooling the perfect discriminator



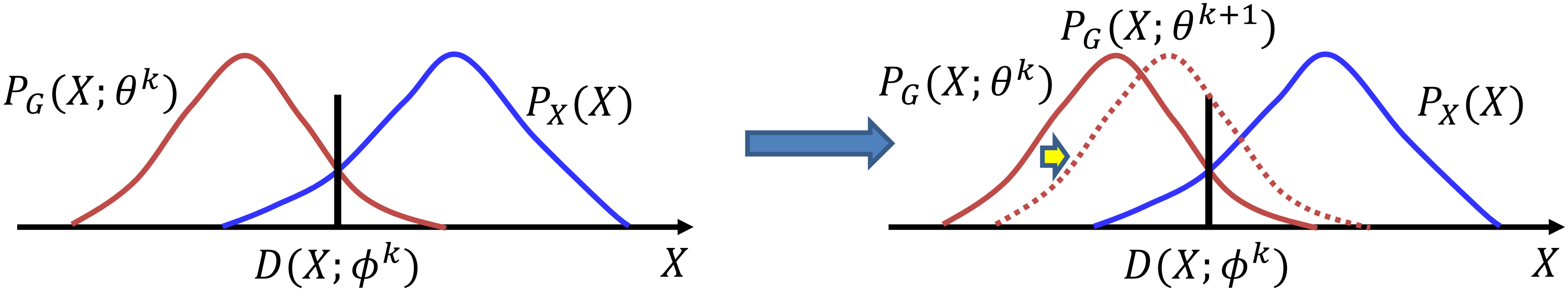
- Relearn generator parameters so that the new distribution of generated data "fools" the discriminator
  - By moving it into the region assigned to the other class by the (perfect) discriminator

The iterated learning



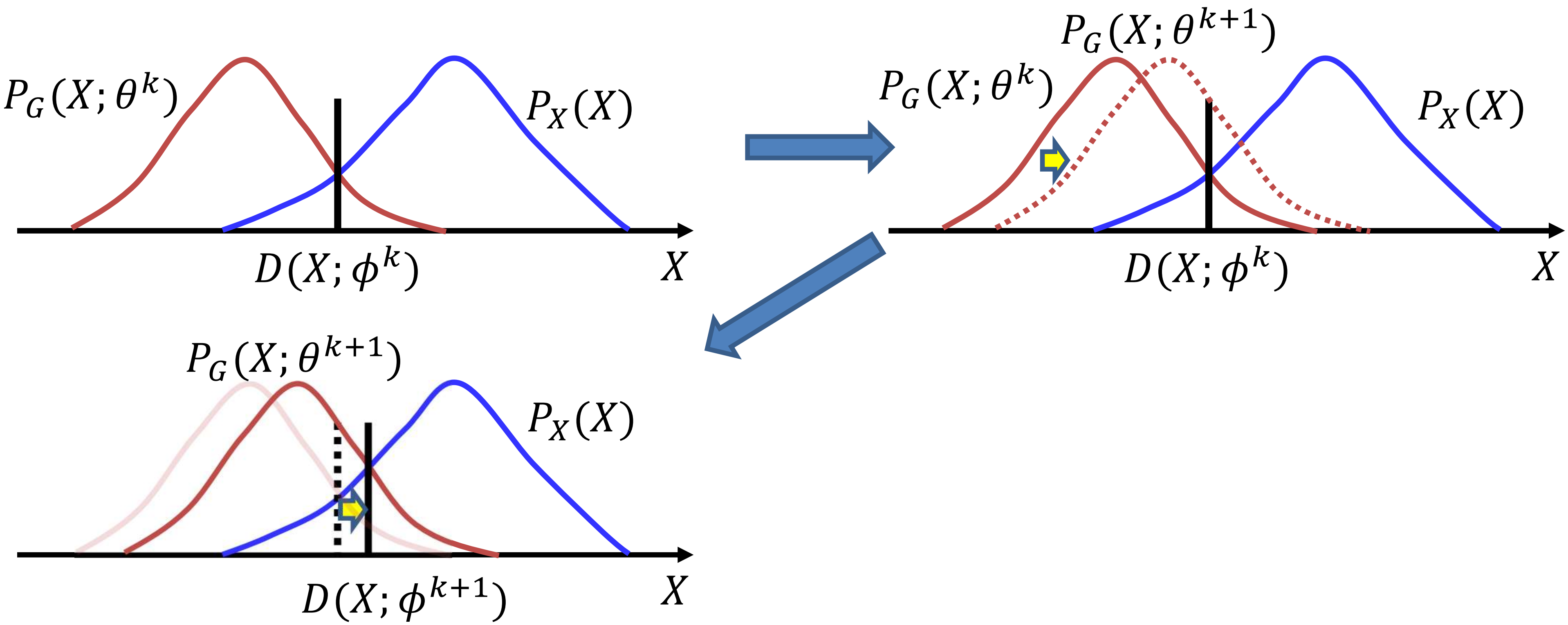
- Discriminator learns perfect boundary

The iterated learning



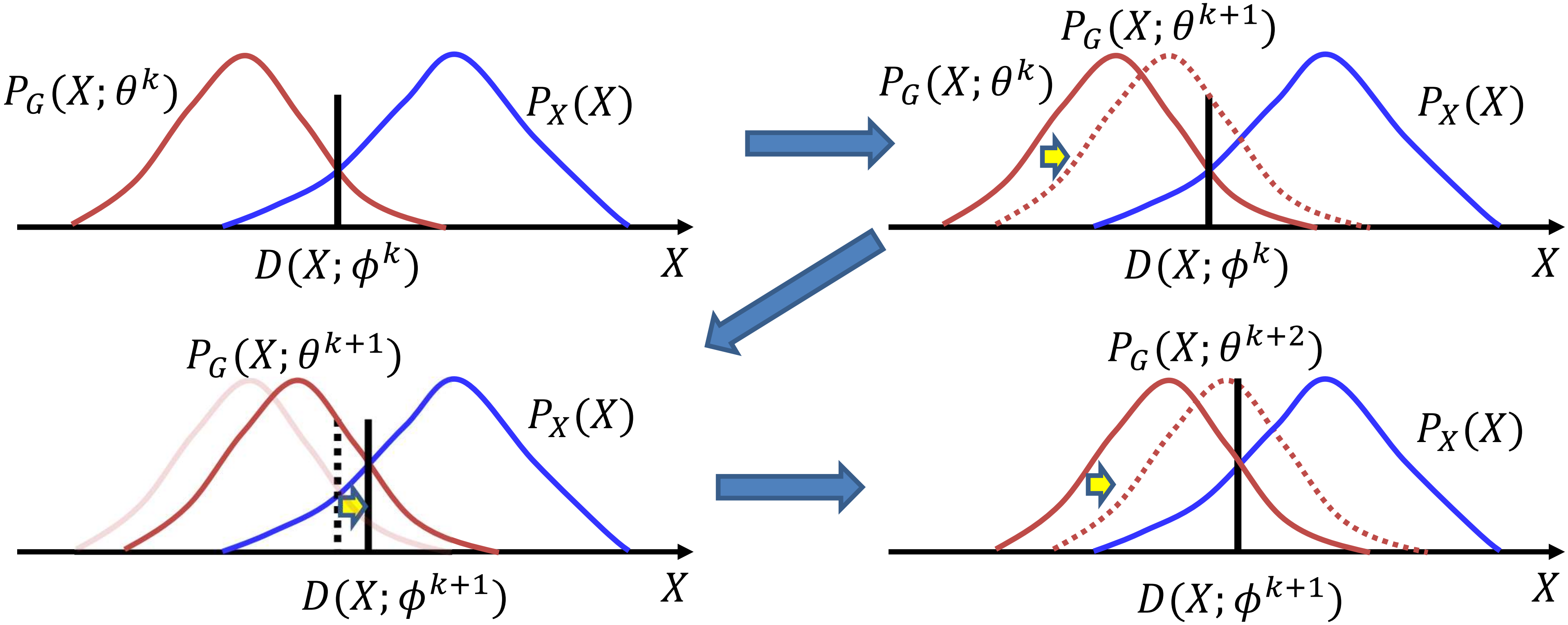
- Discriminator learns perfect boundary
- Generator moves its distribution past the boundary “into” the real distribution

The iterated learning



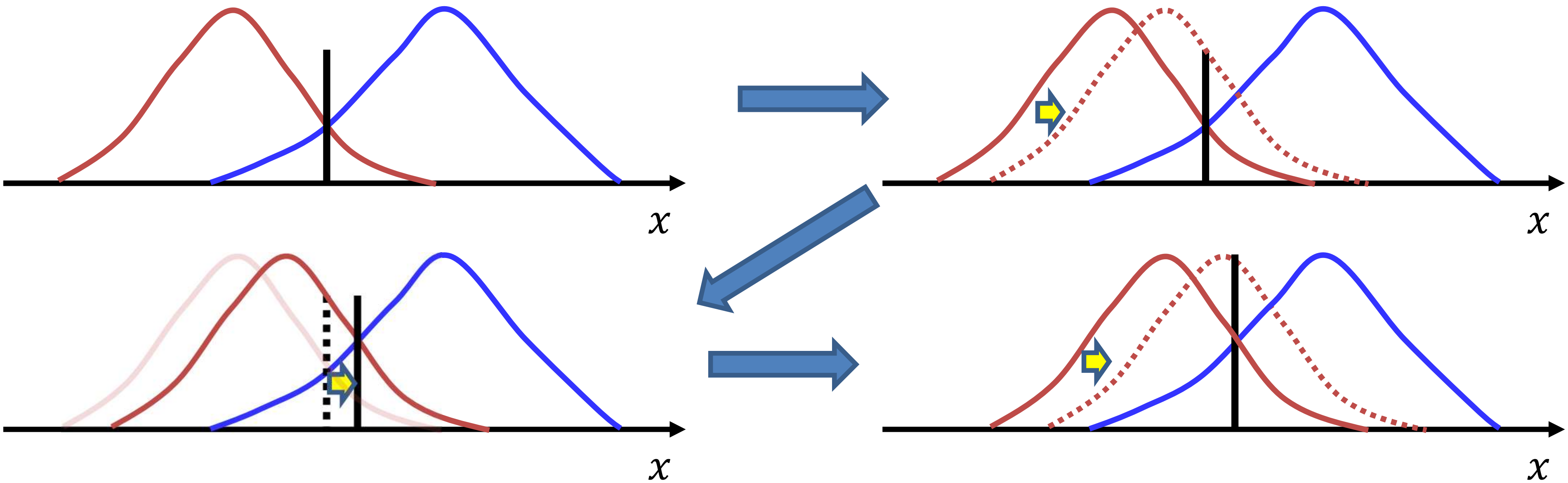
- Discriminator learns perfect boundary
- Generator moves its distribution past the boundary “into” the real distribution
- Discriminator relearns new “perfect” boundary

The iterated learning

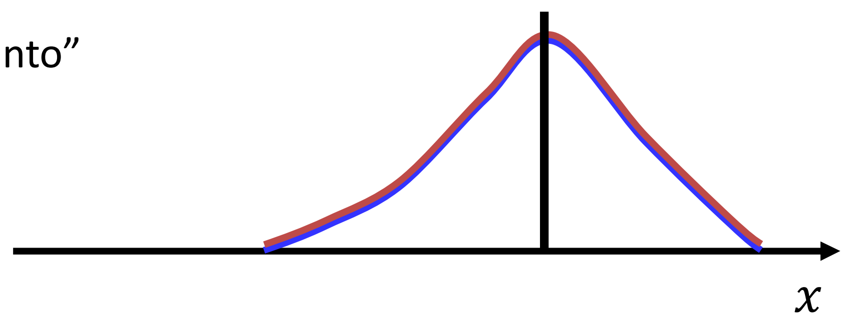


- Discriminator learns perfect boundary
- Generator moves its distribution past the boundary "into" the real distribution
- Discriminator relearns new "perfect" boundary
- Generator shifts distribution past new boundary
- …

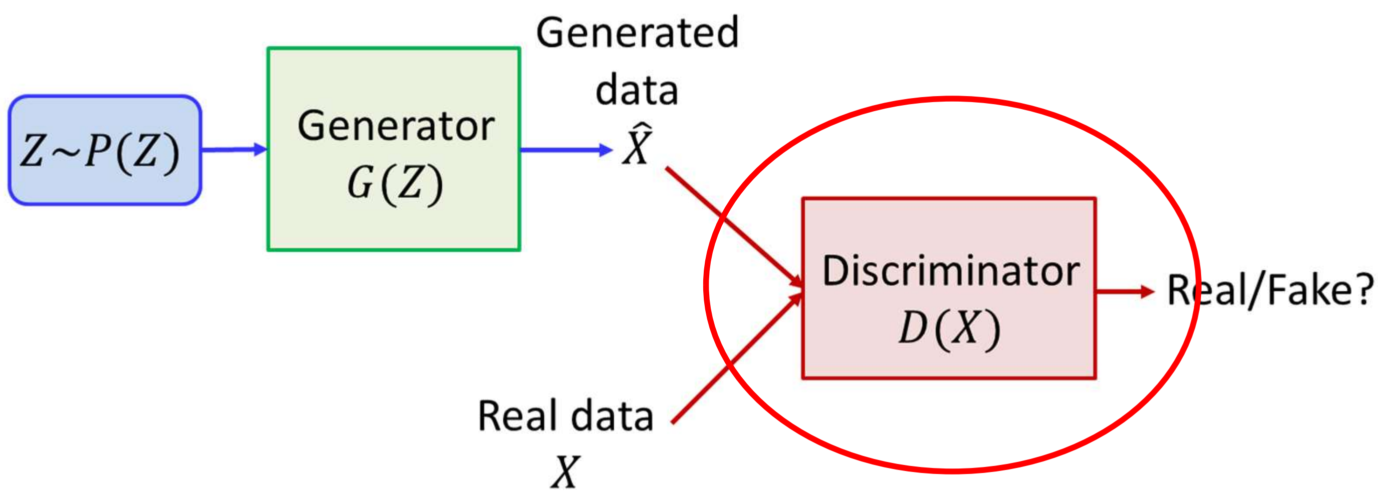
The iterated learning



- 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

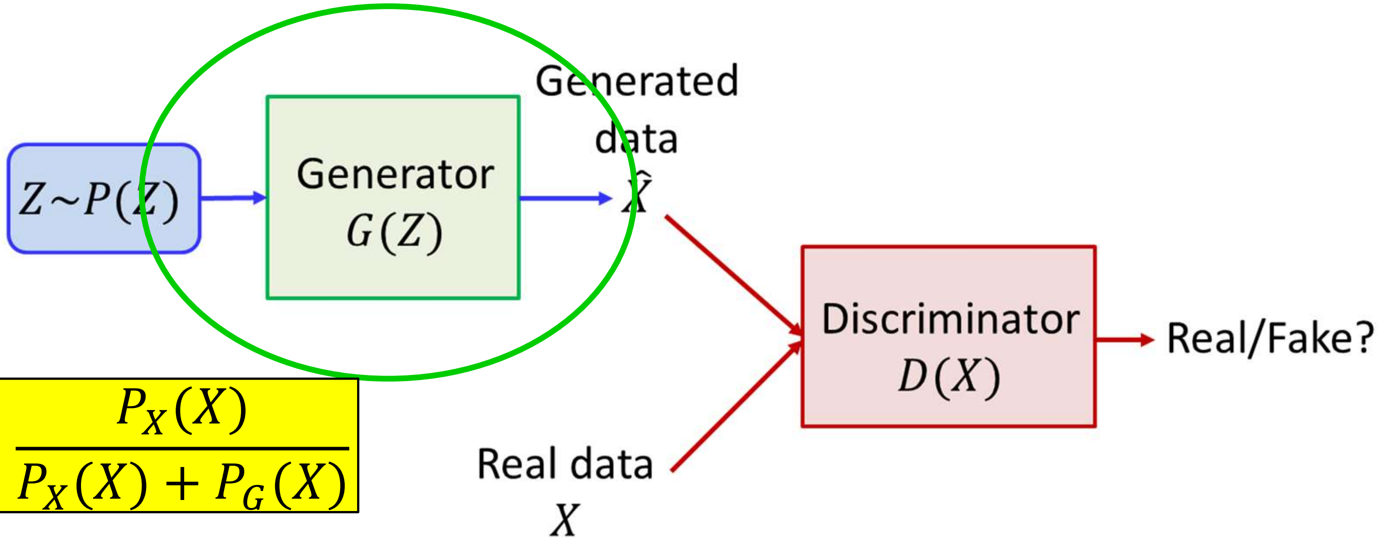


- The **optimal discriminator** would be a Bayesian classifier

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

  – Assuming uniform prior

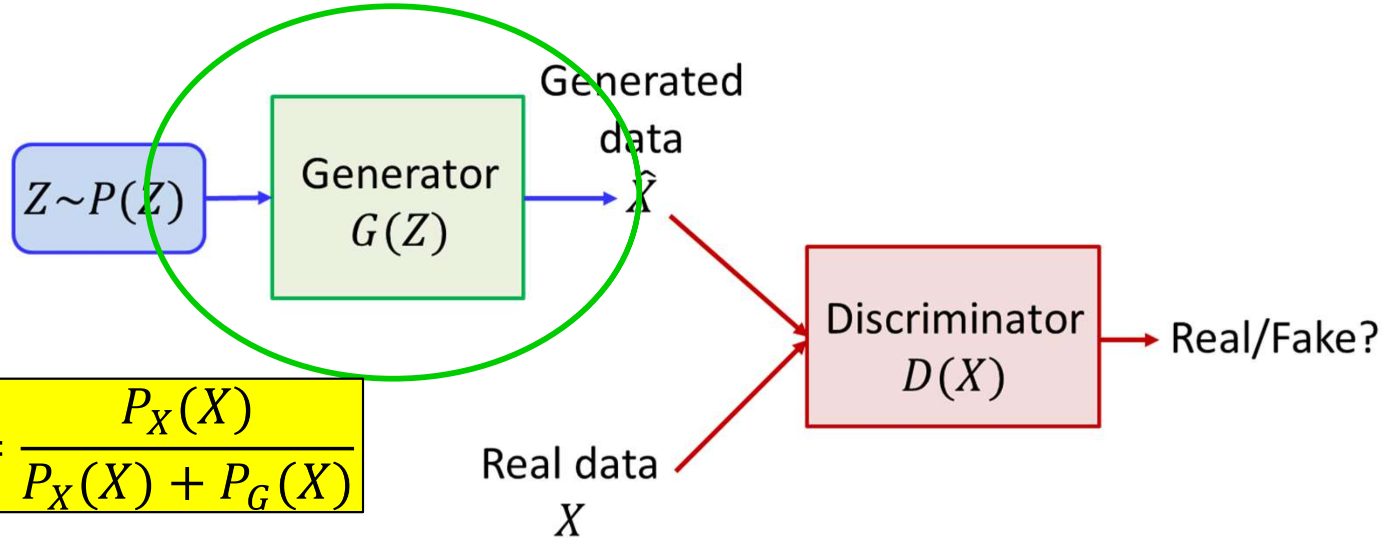
Analysis of optimal behavior: The optimal generator



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

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

Analysis of optimal behavior:
The optimal generator



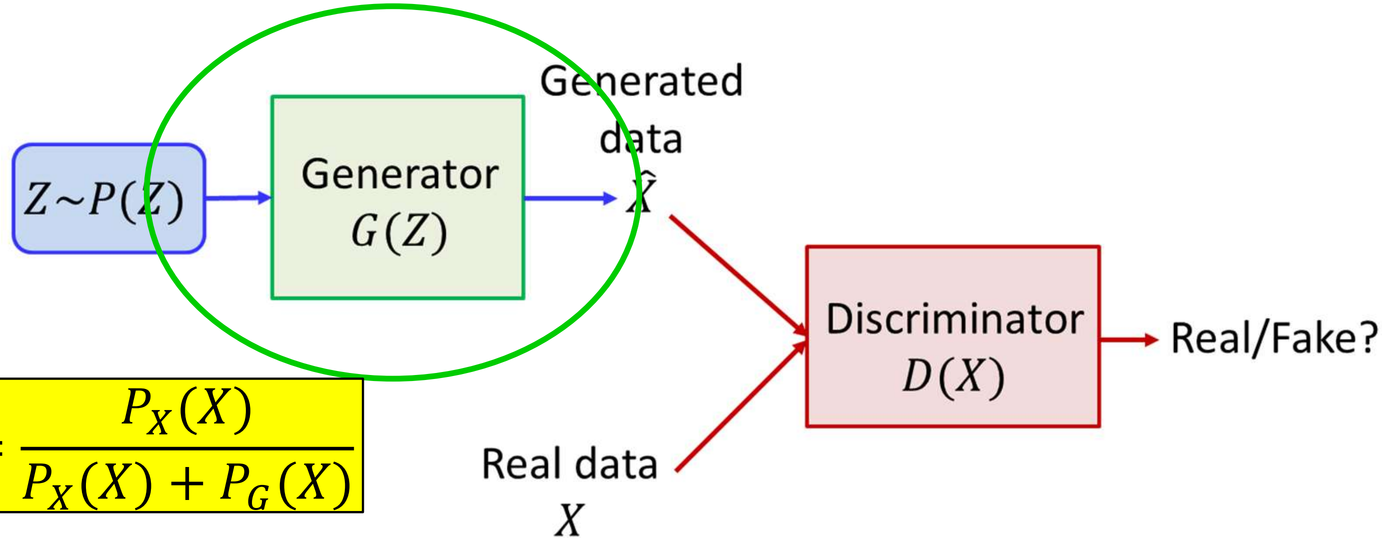
$$\boxed{D(X) = \frac{P_X(X)}{P_X(X) + P_G(X)}}$$

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

- **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



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

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

- **With a perfect discriminator**:

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

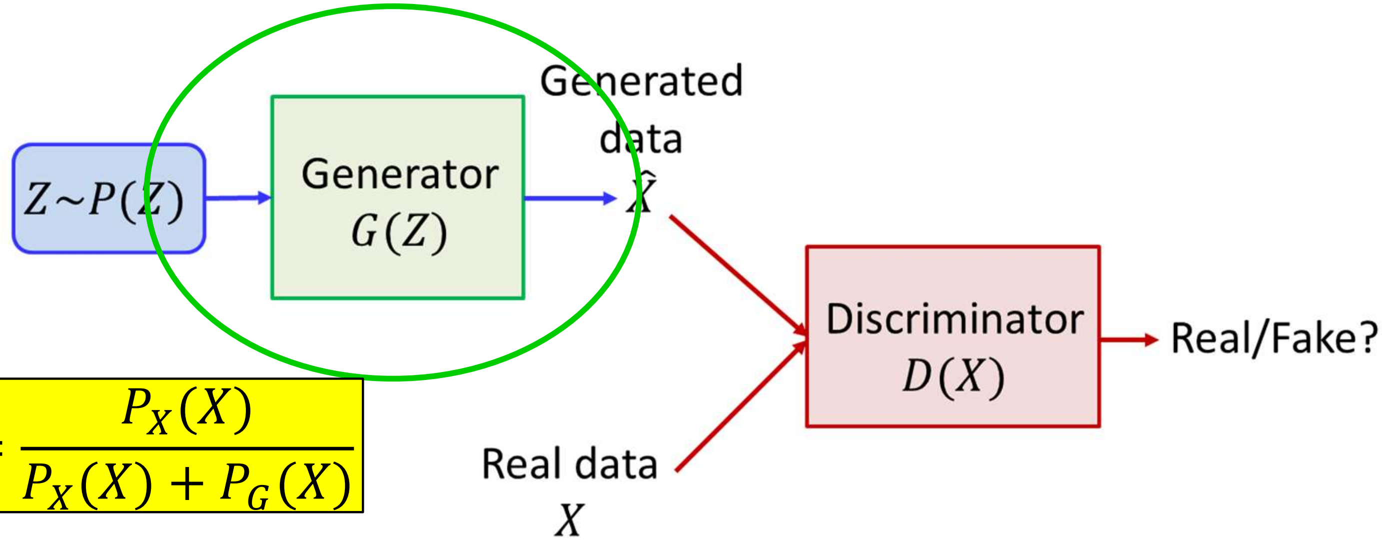
$$= E_{X \sim P_X(X)} \log\left(\frac{P_X(X)}{P_X(X) + P_D(X)}\right) + E_{X \sim P_G(X)} \log\left(\frac{P_G(X)}{P_X(X) + P_D(X)}\right)$$

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



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

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

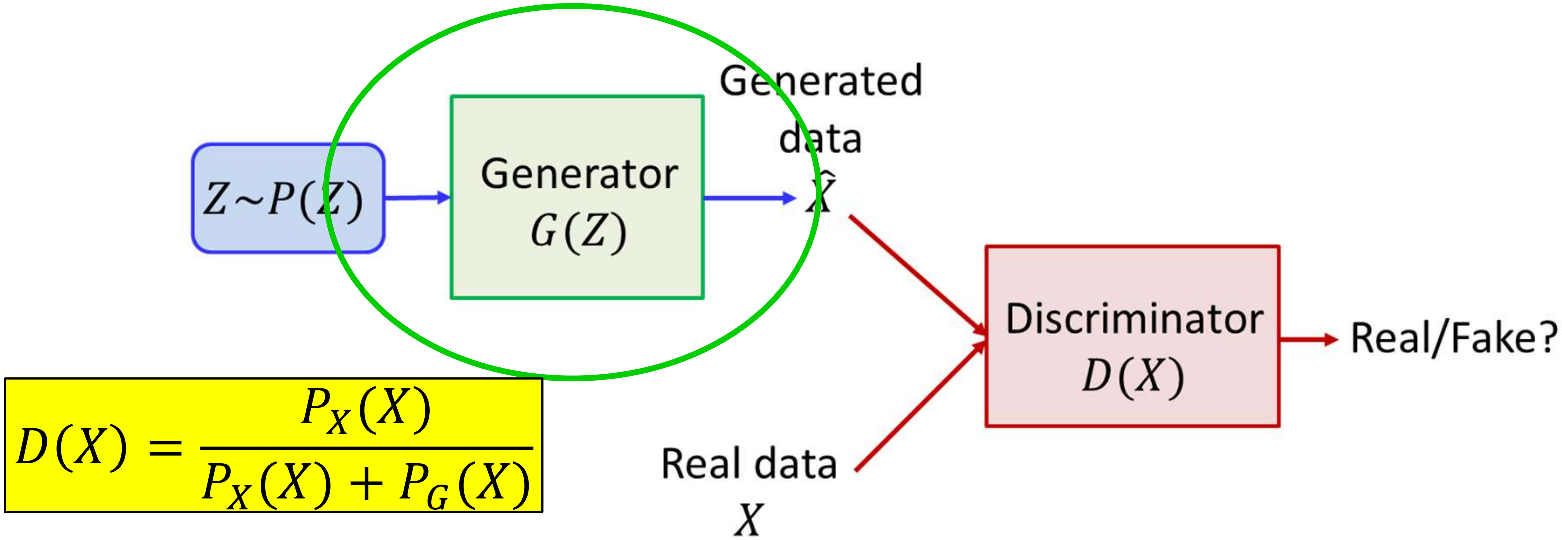
- **With a perfect discriminator**:

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

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

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

Analysis of optimal behavior: The optimal generator

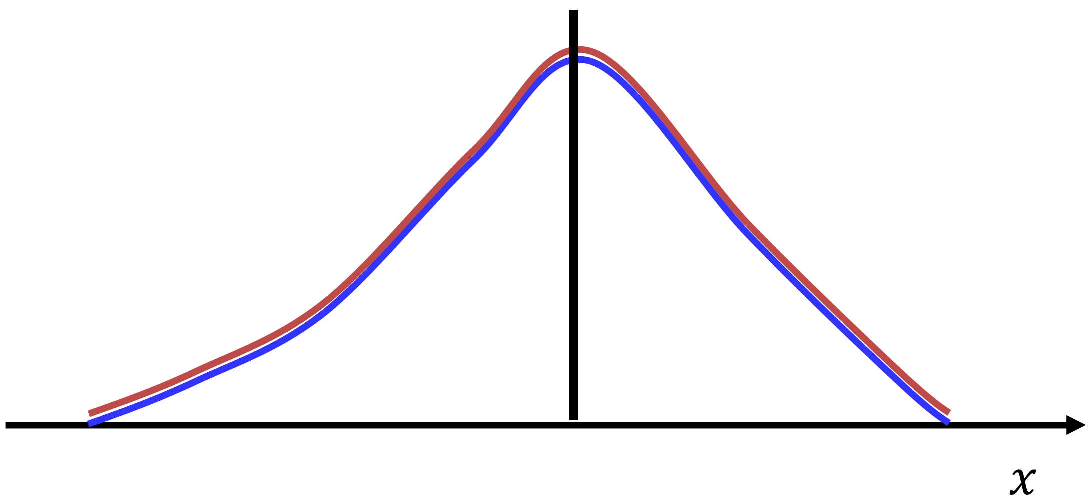


- The optimal generator:

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

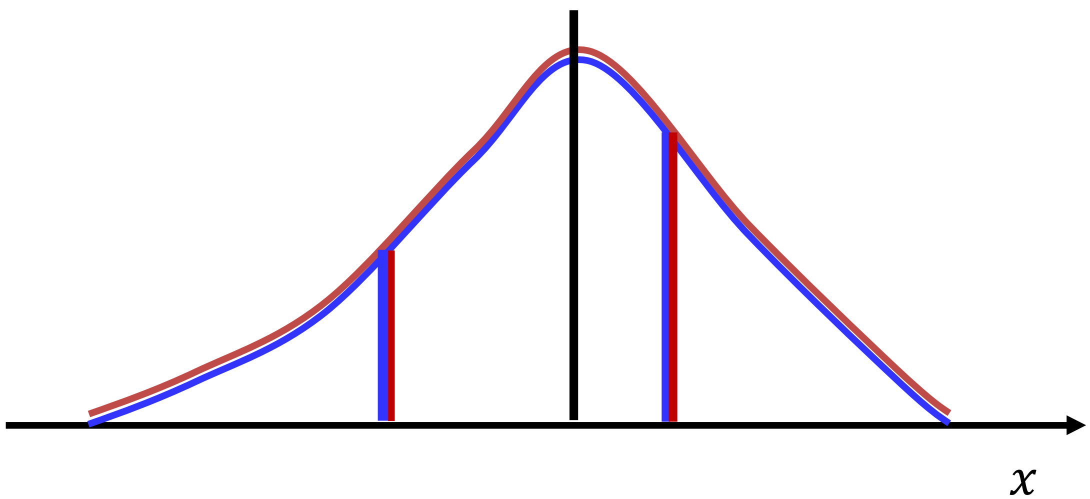
- The optimal generator minimizes the Jensen Shannon divergence between the distributions of the actual and synthetic data!
  - Tries to make the two distributions maximally similar

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

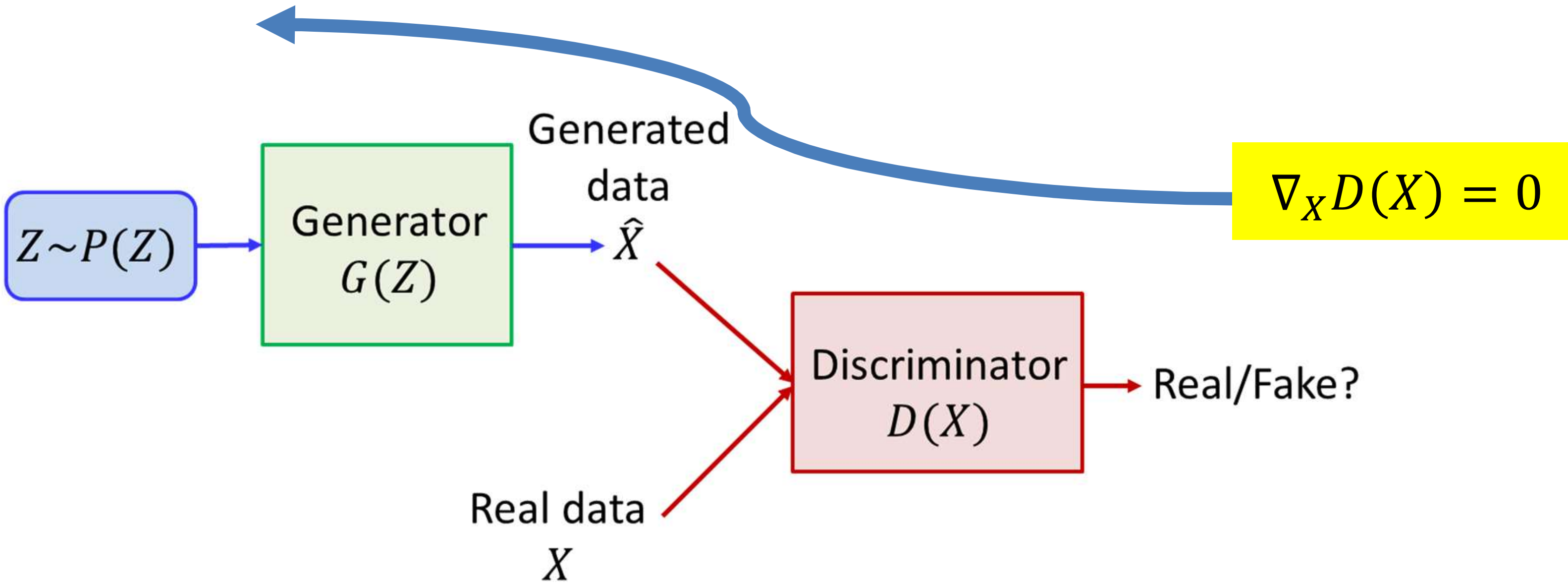
The optimal generator with the optimal discriminator



- 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

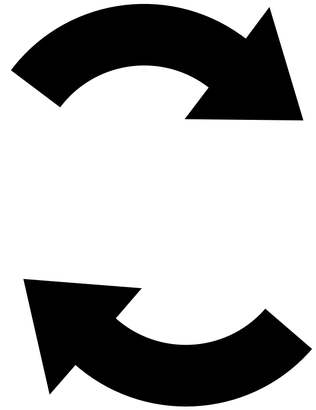
- Stationary points need not be stable (depends on the exact GANs formulation and other factors)
  - Generator may overshoot some values or oscillate around the optimum
  - A discriminator with unlimited capacity can still assign an arbitrarily large distance to 2 similar distributions

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?

Discriminator
$D(X)$



Generator
$G(Z)$

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

- 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**

Variants and updates

- A number of variations have been proposed to improve the stability and outputs of GANs
  - LAPGAN
  - Wasserstein GAN
  - C-GAN
  - DCGAN
  - CycleGAN
  - StarGAN
  - …

Evaluate with Discriminative Network

- Inception Score
  - Use the Inception V3 image classifier to classify generated images
  - Inception should produce a variety of labels
    - **As measured by the entropy of the average label distribution**
  - Each label should have high confidence (low entropy)
    - **As measured by the average entropy of the Inception outputs for individual instances**
  - The two scores are combined into a single “inception” score

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)

Manipulating Celebrity Faces [CCK+17]

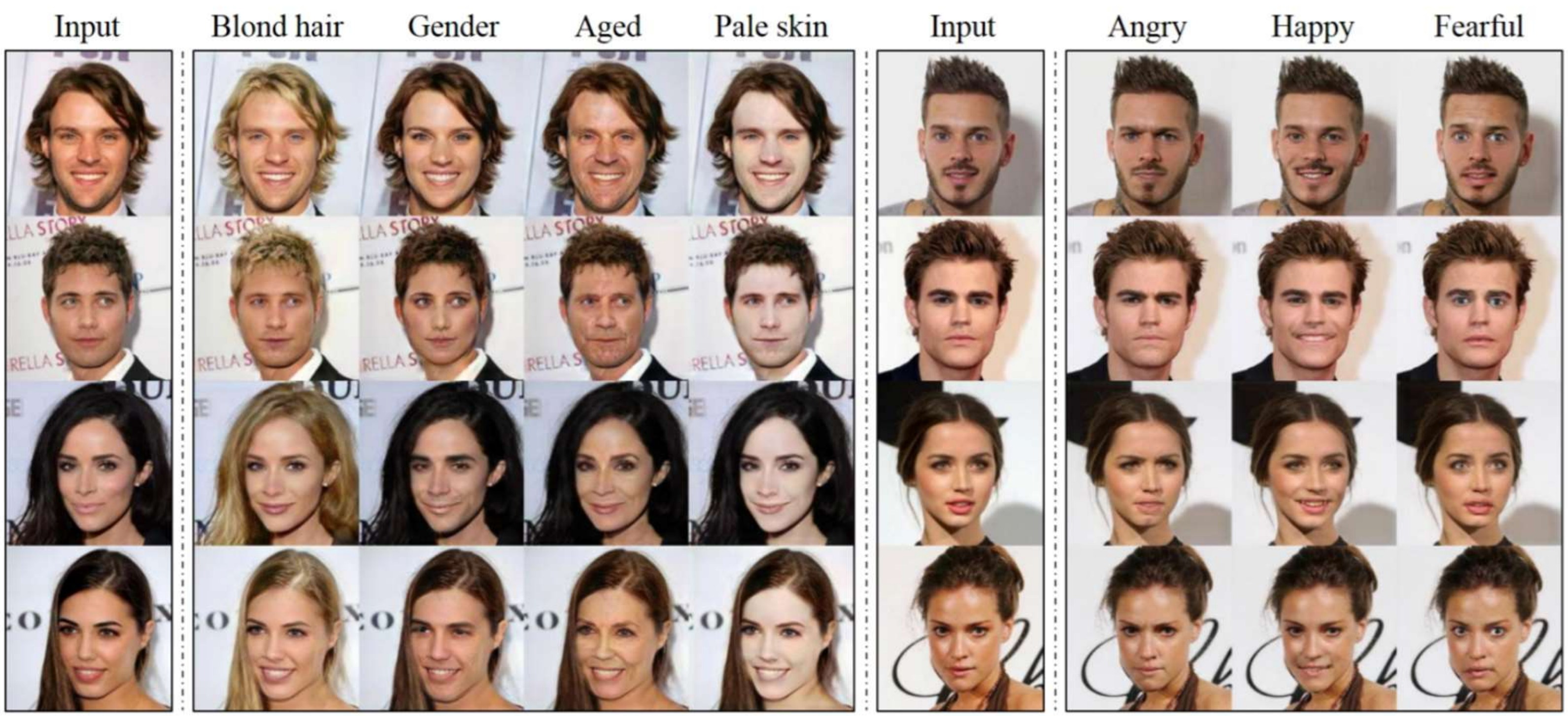


Figure 1. Multi-domain image-to-image translation results on the CelebA dataset via transferring knowledge learned from the RaFD dataset. The first and sixth columns show input images while the remaining columns are images generated by StarGAN. Note that the images are generated by a single generator 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]

Next class

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