Generative Adversarial Networks

11785 Deep Learning

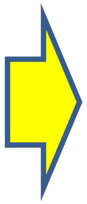
Spring 2023

Abuzar Khan
(…barely)

Topics for the week

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

The problem



Try visiting:
https://thispersondoesnotexist.com

- 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

Generative models

Discriminative vs Generative Models

Given a distribution of inputs X and labels Y.

Discriminative models

- Discriminative models learn conditional distribution $P(Y \mid X)$

Generative models

- 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 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 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

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 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.

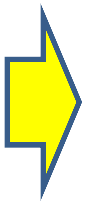
Discriminative models

- 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

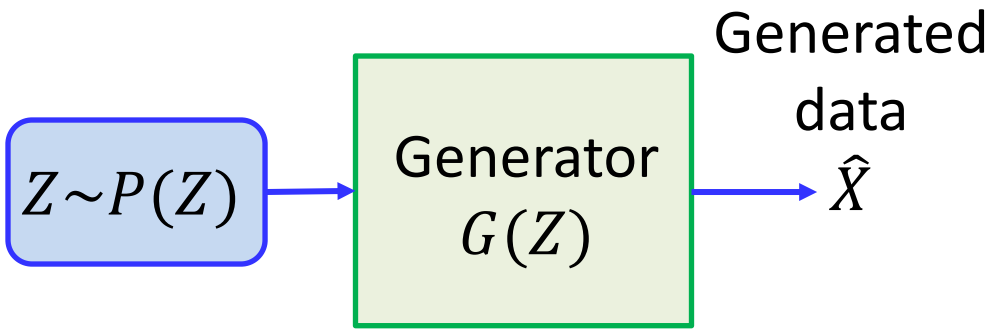
- Generative models learn the joint distribution P(Y, X)
- Learns actual probability distribution of data.
- Can do both generative and discriminative tasks.
- E.g. Naïve Bayes, Gaussian Mixture Model etc.
- Harder problem, requires a deeper understanding of the distribution than discriminative models.

The problem



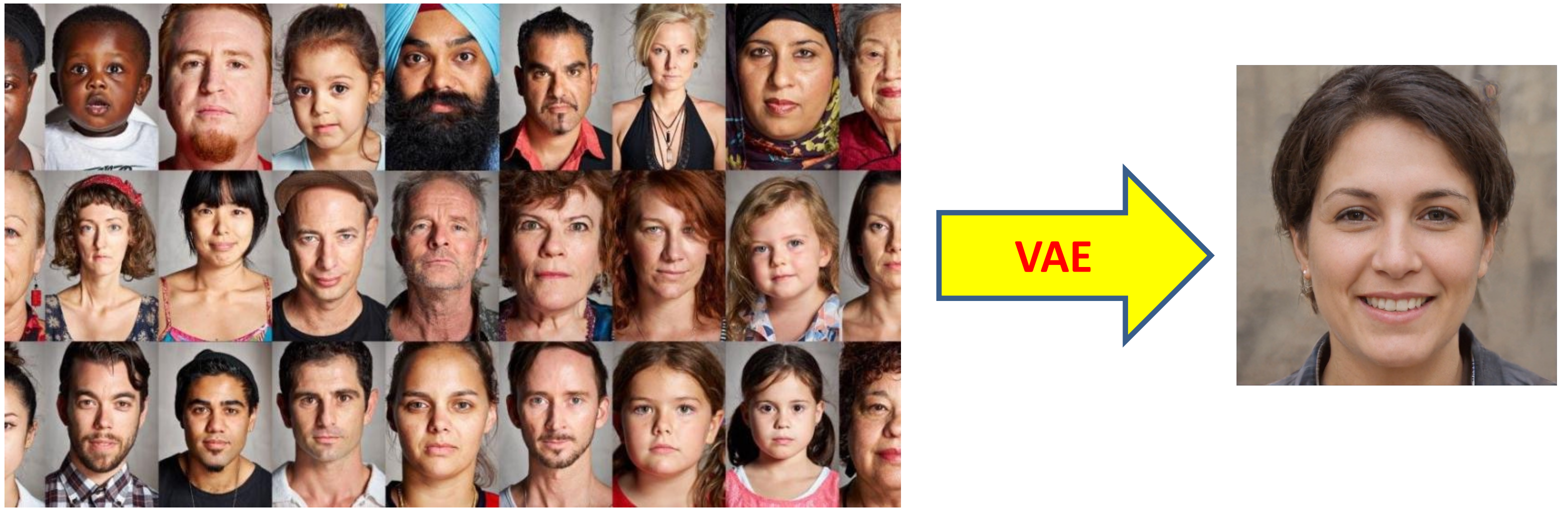
- From a large collection of images of faces, can a network learn to *generate* new portrait
  - Generate samples from the distribution of "face" images
    - How do we even characterize this distribution?

What we have seen: VAE



- The decoder of a VAE is a generator!
- $\hat{X}$ $\rightarrow$ What input image X would have been encoded into the Z that I saw?
- 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 criteria a little more direct?

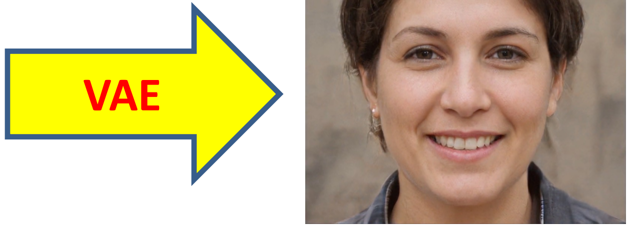
The problem



What's the easiest way to check if the produced output looks like a face or not?

- You could just eyeball 👀 the results and use your understanding of what faces look like to evaluate what is generated.
- Unfortunately, **you** are not a differentiable function 🙃

The problem



What's the easiest way to check if the produced output looks like a face or not?

- You could just eyeball 👀 the results and use your understanding of what faces look like to evaluate what is generated.
- Unfortunately, **you** are not a differentiable function 🙃
- But what if we could, find a differentiable proxy for you and your non-differentiable-ness?

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)

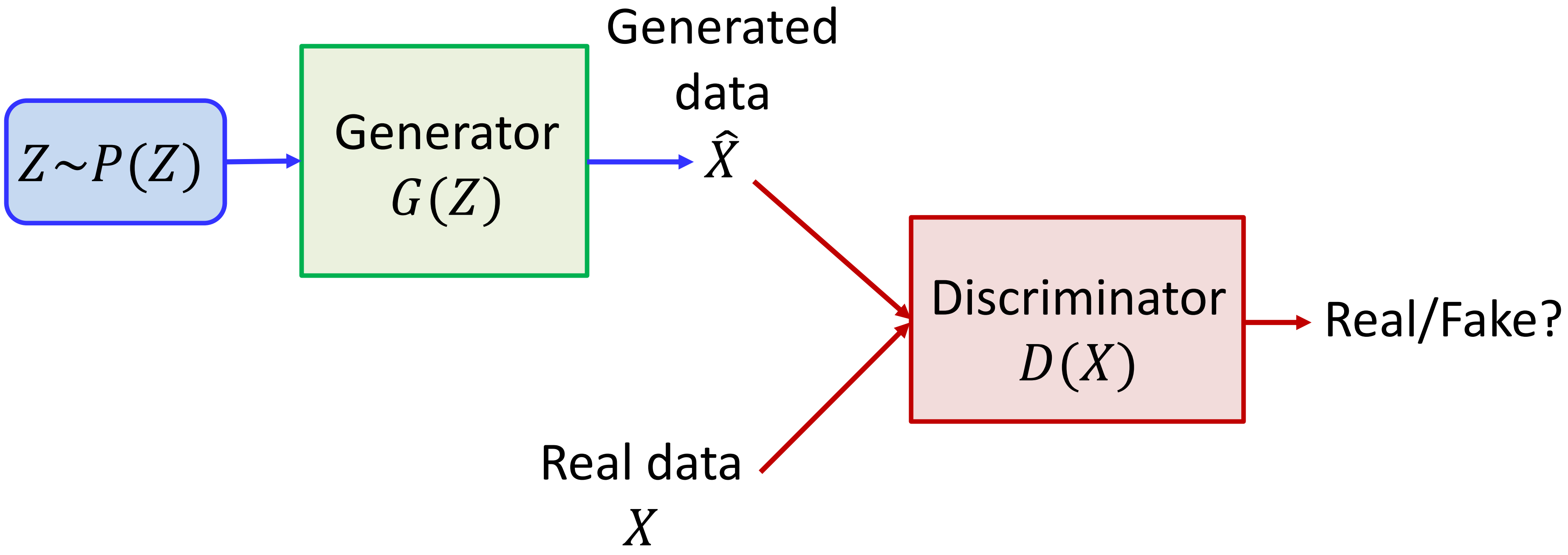
Adversarial Training
GANS are made up of two competing networks (adversaries) that are trying beat each other.
A “game” is being played between the two.

Neural Networks

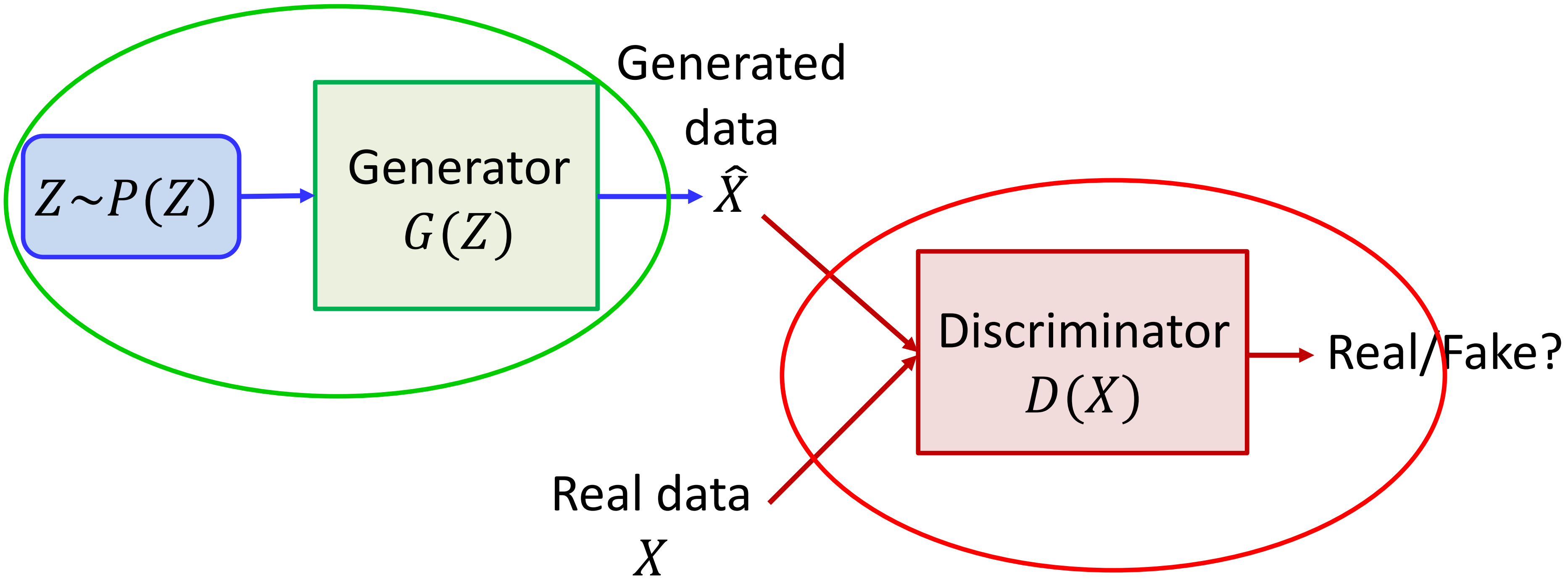
Generative Adversarial Networks

- 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

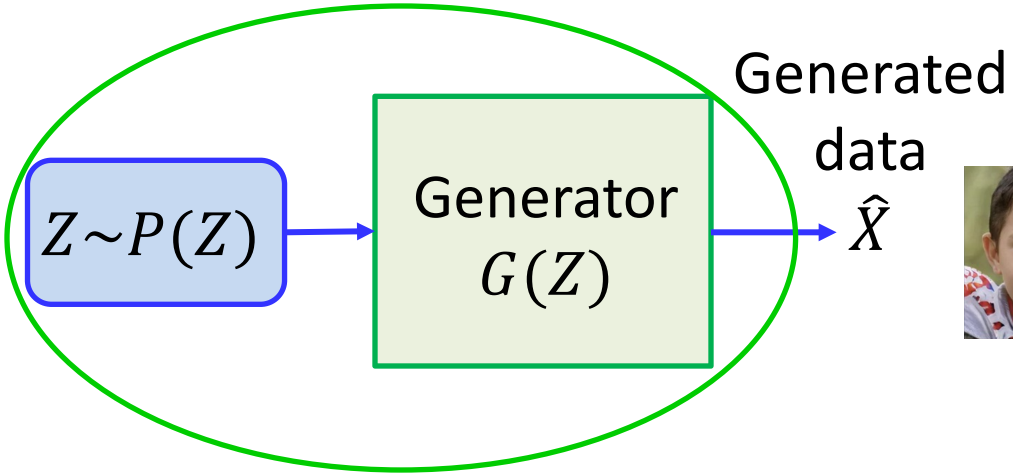
What are GANs?



What are GANs?



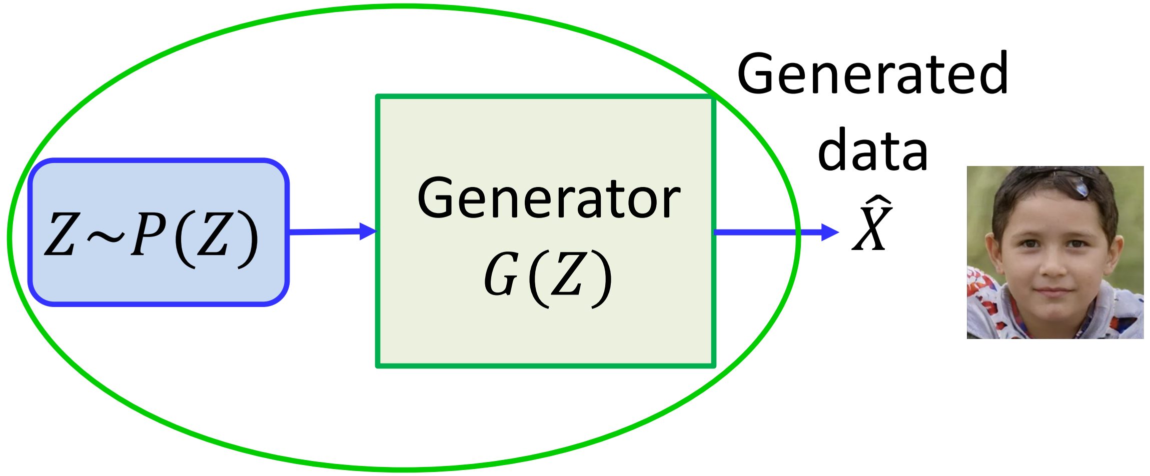
The Generator



- **The generator** produces realistic looking $\hat{\boldsymbol{X}} = G(z)$ 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 $P_{\hat{\boldsymbol{X}}}(X)$, the probability that a real data point X could be generated by our model as $\hat{X}$

The Generator

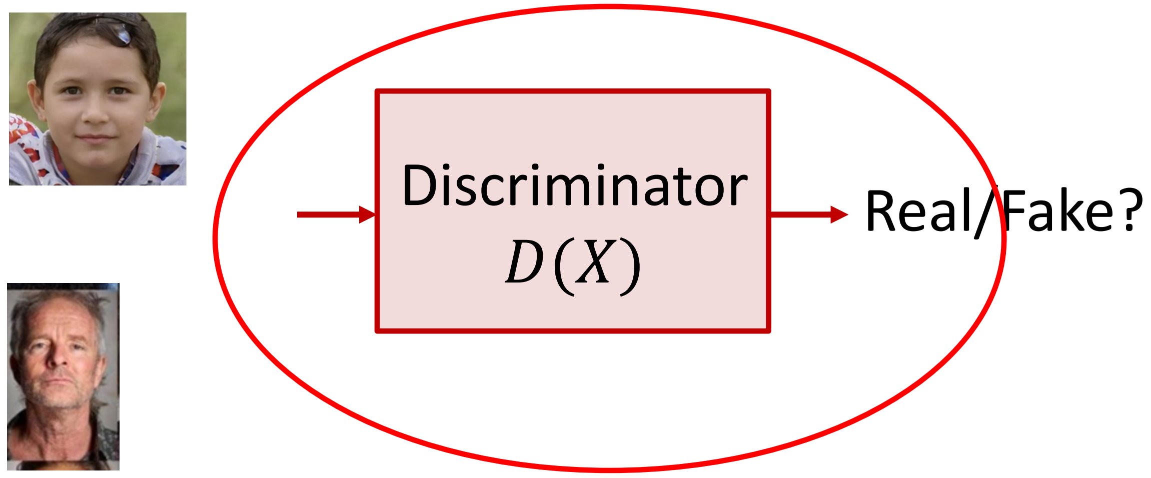


- **The generator** produces realistic looking $\hat{\boldsymbol{X}} = G(z)$ from a latent vector Z
- Generator input Z can be sampled from a known prior, e.g. standard Gaussian

- **Goal**: generated distribution, $P_G(X)$ matches the true data distribution $P_X(X)$
  - $P_G(X)$ is the more "memorable" notati for $P_{\hat{X}}(X)$, the probability that a real data point X could be generated by our model a $\hat{X}$

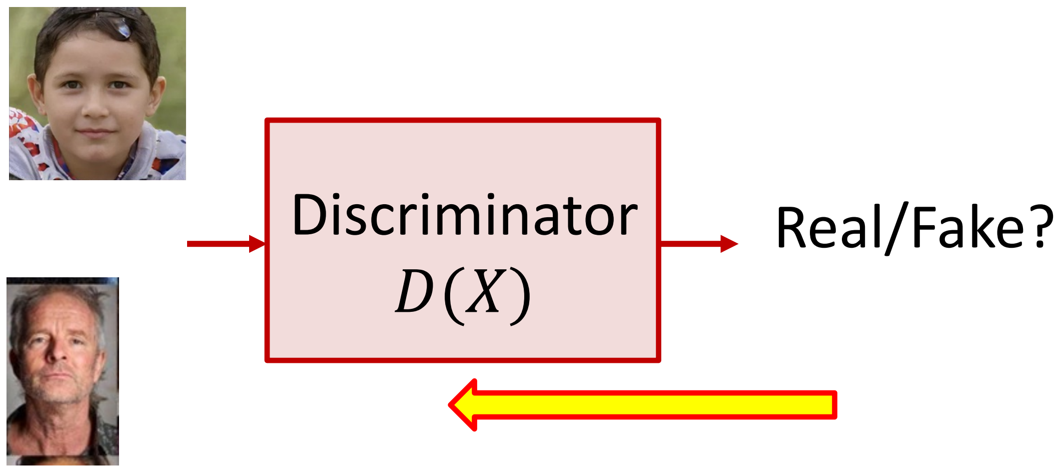
Implicit Model!

The Discriminator



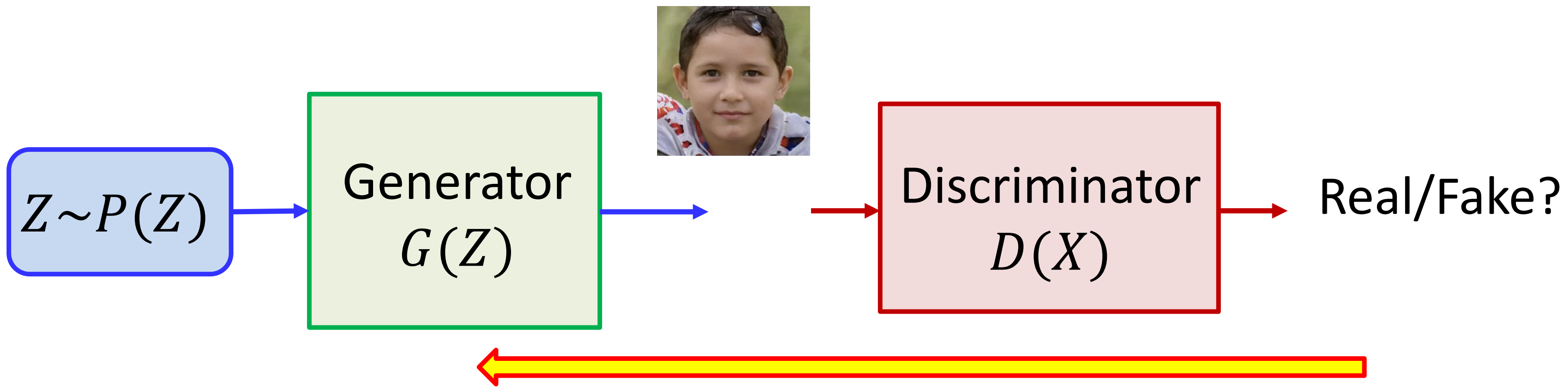
- Discriminator $D(X)$ is trained to tell the difference between real and generated (fake) data
  - Specifically, data produced by the generator
  - If a perfect discriminator is fooled, the generated data cannot be distinguished from real data

Training 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

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

Poll - 1

@1300 (such a nice number 😌)

Which of the following is the best way to train a GAN?

A. First train the generator, then train a discriminator that can correct the mistakes of the generator and continue this cycle.
B. First train the discriminator to be perfect, and then train a generator that can beat this discriminator. This way we will get a good generator in one go.
C. First train the discriminator to be not terrible, then train the generator to produce data that fools the discriminator. Continue this cycle till convergence.
D. Throw AdamW at it and hope it works.
E. Poke it with a stick.

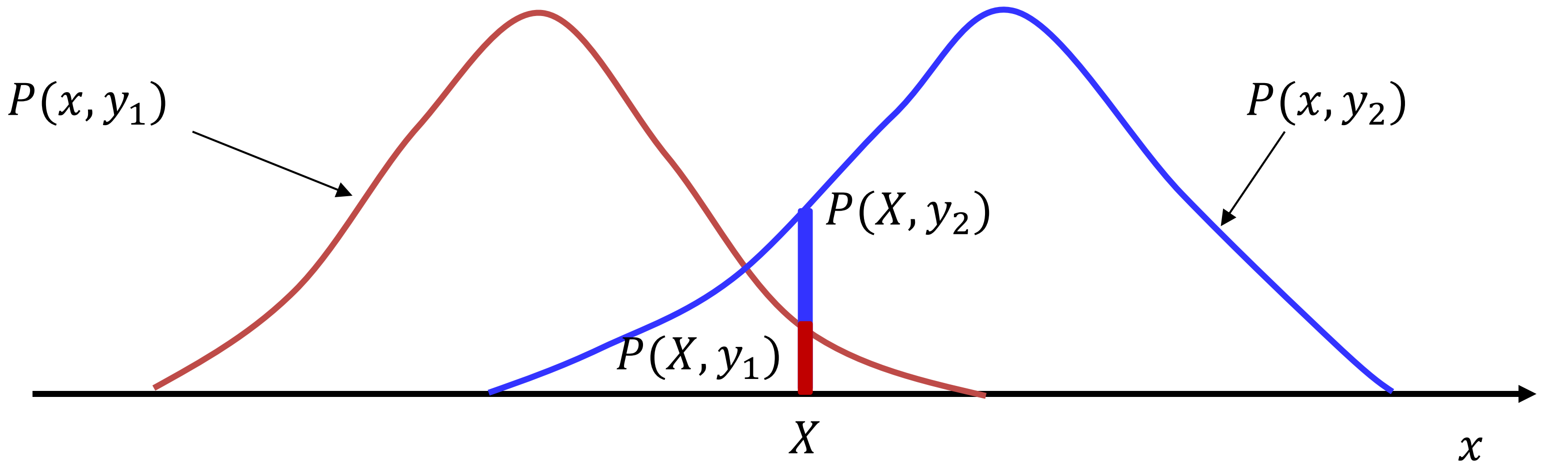
Poll - 1

@1300 (such a nice number 😌)

Which of the following is the best way to train a GAN?

A. First train the generator, then train a discriminator that can correct the mistakes of the generator and continue this cycle.
B. First train the discriminator to be perfect, and then train a generator that can beat this discriminator. This way we will get a good generator in one go.
C. First train the discriminator to be not terrible, then train the generator to produce data that fools the discriminator. Continue this cycle till convergence.
D. Throw AdamW at it and hope it works.
E. Poke it with a stick.

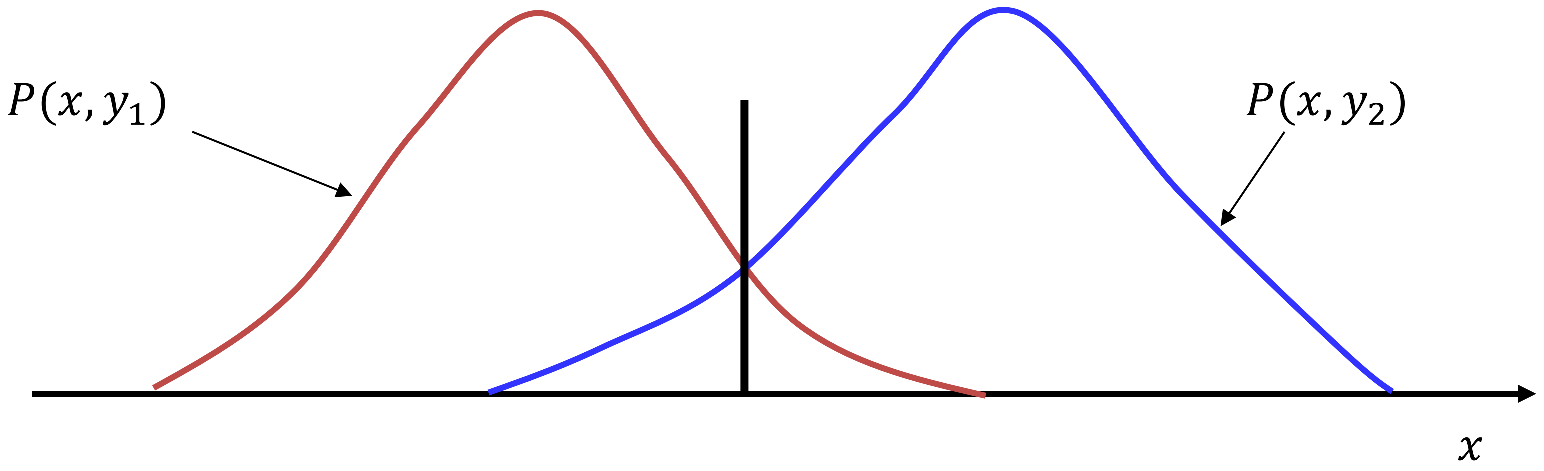
The perfect discriminator



- 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

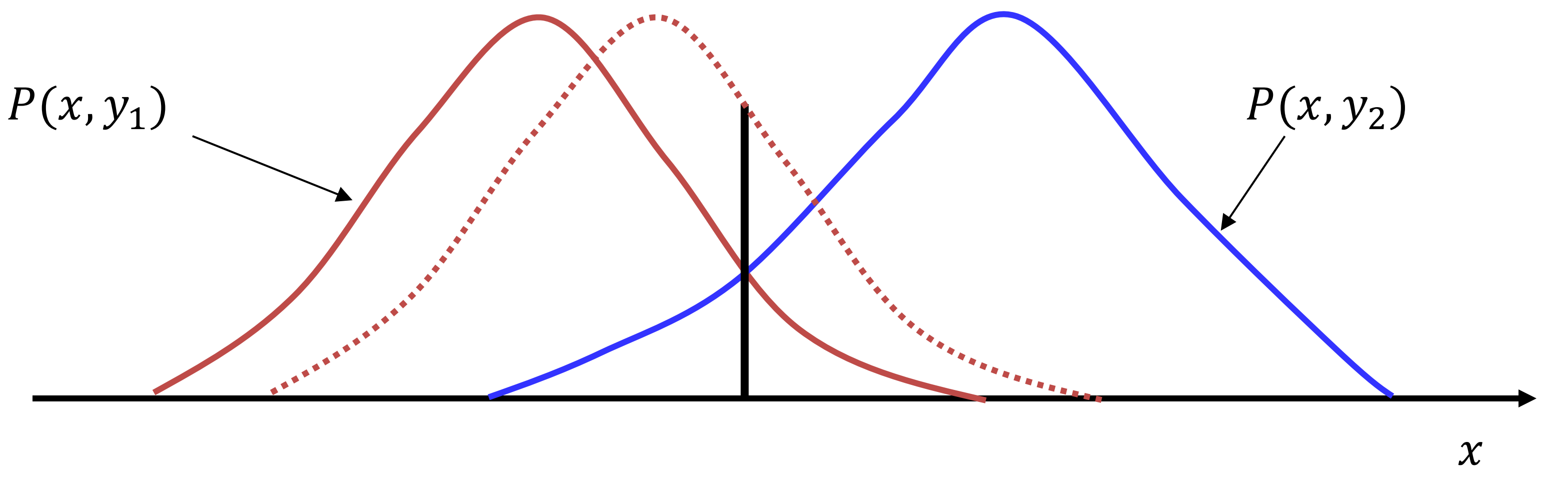


- The a posteriori probability of the classes for any instance $x = X$ is

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

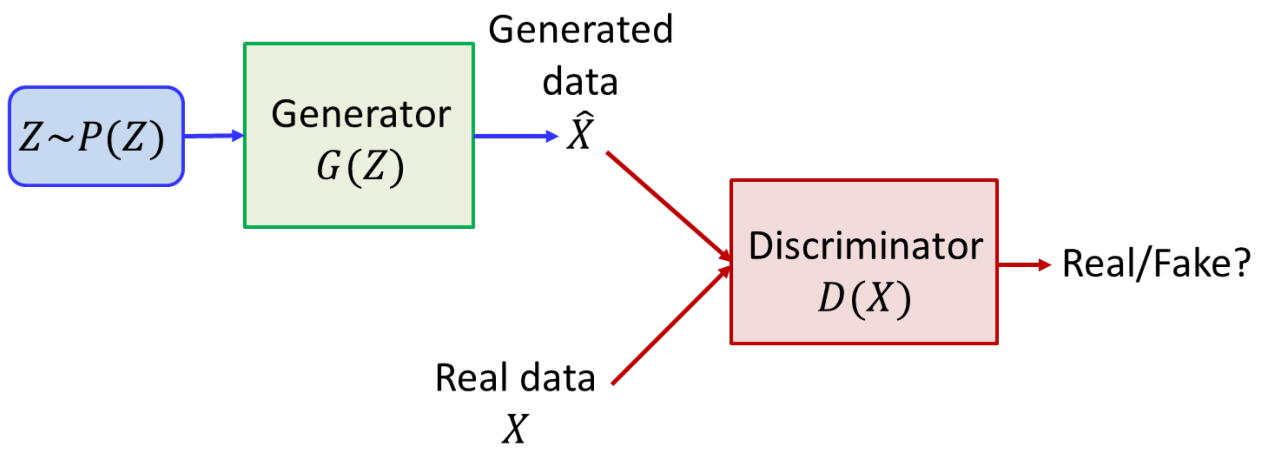
- The perfect decision boundary is where $P(y_1|X) = P(y_2|X)$
  - The perfect discriminator will compute $P(y_i|X)$ for each class
  - It will assign any X to the class with the higher $P(y_i|X)$

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 GAN formulation



- For real data X, the desired output of the discriminator is $D(X) = 1$
  - The log probability that the instance is real, as computed by the discriminator is $\log D(X)$

- For synthetic data $\hat{X}$, the desired output of the discriminator is $D(\hat{X}) = 0$
  - The log probability that the instance is synthetic, as computed by the discriminator, is $\log(1 - D(\hat{X}))$
    - $= \log(1 - D(G(Z)))$

Detour: Information

$$\boldsymbol{I}(\boldsymbol{x}) = -\log(\boldsymbol{P}(\boldsymbol{x}))$$

Information is more where probability is low

“Today I didn’t see a tornado”

“Today I saw a tornado”

Detour: Information

$$\boldsymbol{I}(\boldsymbol{x}) = -\log(\boldsymbol{P}(\boldsymbol{x}))$$

Information is more where probability is low

“Today I didn’t see a tornado” – low information

“Today I saw a tornado” – high information

Detour: Entropy

$$\boldsymbol{H}(\boldsymbol{x}) = -\sum \boldsymbol{P}(\boldsymbol{x})\log\left(\boldsymbol{P}(\boldsymbol{x})\right)$$
$$\boldsymbol{H}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{x} \sim \boldsymbol{X}}[-\log(\boldsymbol{P}(\boldsymbol{x}))]$$

Measure of uncertainty – weighted average of information

Detour: Cross Entropy

Information of event under some distribution Q measured over another distribution P

$$\boldsymbol{XEnt}(\boldsymbol{x}; \boldsymbol{P}, \boldsymbol{Q}) \; = \; -\sum \boldsymbol{P}(\boldsymbol{x}) \log\left(\boldsymbol{Q}(\boldsymbol{x})\right)$$

$$\boldsymbol{XEnt}(\boldsymbol{x}; \boldsymbol{P}, \boldsymbol{Q}) = \; \mathbb{E}_{\boldsymbol{x} \sim \boldsymbol{P}(\boldsymbol{X})}[-\log(\boldsymbol{Q}(\boldsymbol{x}))]$$

Detour: Binary Cross Entropy

Information of event under some distribution Q measured over another distribution P

When $P(x)$ can be **0** or **1**

$$\boldsymbol{BCE(x; P, Q)}$$
$$= -\sum \boldsymbol{P(x)}\log\left(\boldsymbol{Q(x)}\right) + \left(\boldsymbol{1} - \boldsymbol{P(x)}\right)\log(\boldsymbol{1} - \boldsymbol{Q(x)})$$

$$\boldsymbol{BCE(x; P, Q)} =$$
$$-\left\{\mathbb{E}_{\boldsymbol{x} \sim \boldsymbol{P}(\boldsymbol{X})}\left[\log\left(\boldsymbol{Q(x)}\right)\right] + \mathbb{E}_{\boldsymbol{x} \sim \bar{\boldsymbol{P}}(\boldsymbol{X})}\left[\log\left(\boldsymbol{1} - \boldsymbol{Q(x)}\right)\right]\right\}$$

Coming back: Discriminator

$$\mathrm{BCE}_{\boldsymbol{D}}(\boldsymbol{x};\boldsymbol{D}) = -\{\underbrace{\boxed{\mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{P}_X}\left[\log\left(\boldsymbol{D}(\boldsymbol{x})\right)\right]}}_{\text{True}} + \underbrace{\boxed{\mathbb{E}_{\boldsymbol{x}\sim \overline{\boldsymbol{P}}_X}\left[\log\left(\boldsymbol{1} - \boldsymbol{D}(\boldsymbol{x})\right)\right]}}_{\text{Fake}}\}$$

Discriminator

$$\text{BCE}_{\boldsymbol{D}}(\boldsymbol{x}\,;\boldsymbol{D},\boldsymbol{G}) = \\ -\boxed{\{\mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{P}_{\boldsymbol{X}}}\left[\log\left(\boldsymbol{D}(\boldsymbol{x})\right)\right]} + \boxed{\mathbb{E}_{\boldsymbol{z}\sim \boldsymbol{P}_{\boldsymbol{z}}}[\log(\mathbf{1} - \boldsymbol{D}(\,\boldsymbol{G}(\boldsymbol{z})))]\}}$$

True

Generated

$$\text{BCE}_{\boldsymbol{D}}(\boldsymbol{x}\,;\boldsymbol{D},\boldsymbol{G}) = \\ -\{\mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{P}_{\boldsymbol{X}}}\left[\log\left(\boldsymbol{D}(\boldsymbol{x})\right)\right] + \mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{P}_{\boldsymbol{G}}}[\log(\mathbf{1} - \boldsymbol{D}(\boldsymbol{x})))]\}$$

Implicit Model

Discriminator

$$\mathrm{BCE}_{\boldsymbol{D}}(\boldsymbol{x}\,;\boldsymbol{D},\boldsymbol{G}) = \\ -\{\mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{P}_X}\left[\log\left(\boldsymbol{D}(\boldsymbol{x})\right)\right] + \mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{P}_G}[\log(\mathbf{1}\;-\boldsymbol{D}(\boldsymbol{x}))]\}$$

Implicit Model

Minimizing the above is the same as maximizing

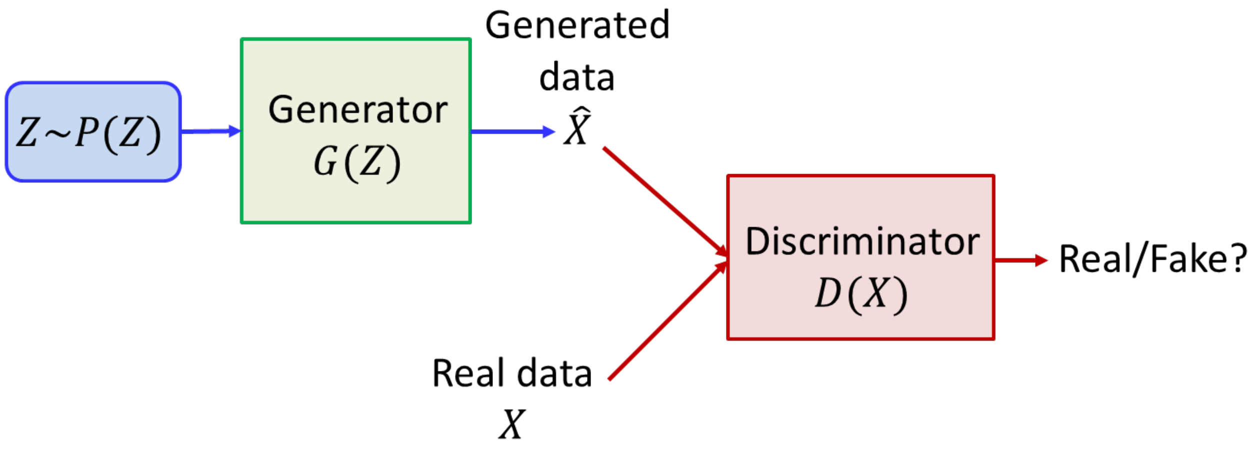
$$\mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{P}_X}\left[\log\left(\boldsymbol{D}(\boldsymbol{x})\right)\right] + \mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{P}_G}[\log(\mathbf{1}\;-\boldsymbol{D}(\boldsymbol{x}))]$$

Generator

But the generator wants to fool the Discriminator, so it wants to minimize

$$\mathbb{E}_{\boldsymbol{x} \sim \boldsymbol{P}_X}\left[\log\left(\boldsymbol{D}(\boldsymbol{x})\right)\right] + \mathbb{E}_{\boldsymbol{x} \sim \boldsymbol{P}_G}[\log(\mathbf{1} - \boldsymbol{D}(\boldsymbol{x}))]$$

The GAN formulation

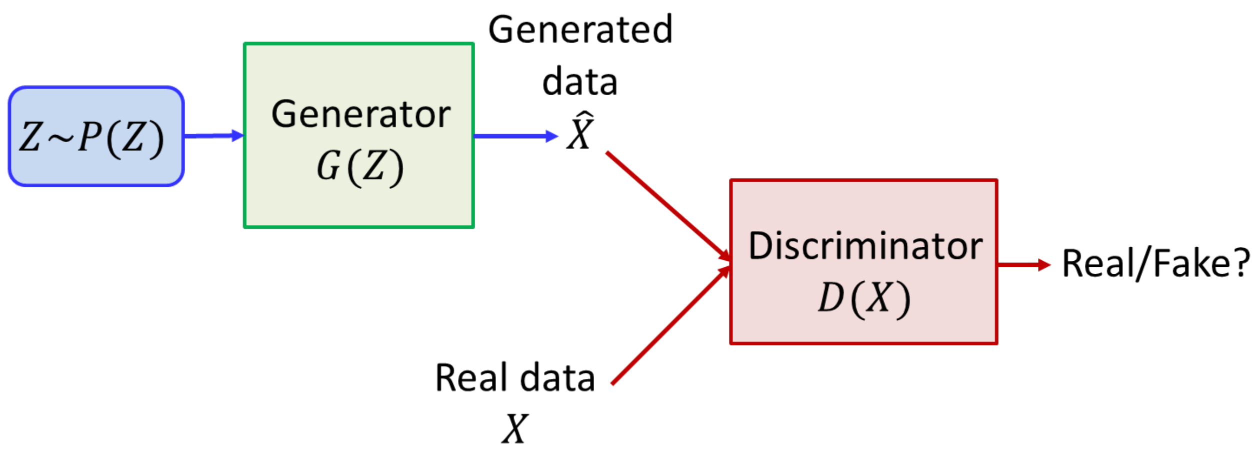


- Can be written as

$$\min_{G} \max_{D} \quad E_{\boldsymbol{x} \sim \boldsymbol{P_X}} \log D(x) + E_{\boldsymbol{x} \sim \boldsymbol{P_G}} \log(1 - D(x))$$

In theory, we are optimized when we reach a Nash Equilibrium in this “Game”

The GAN formulation

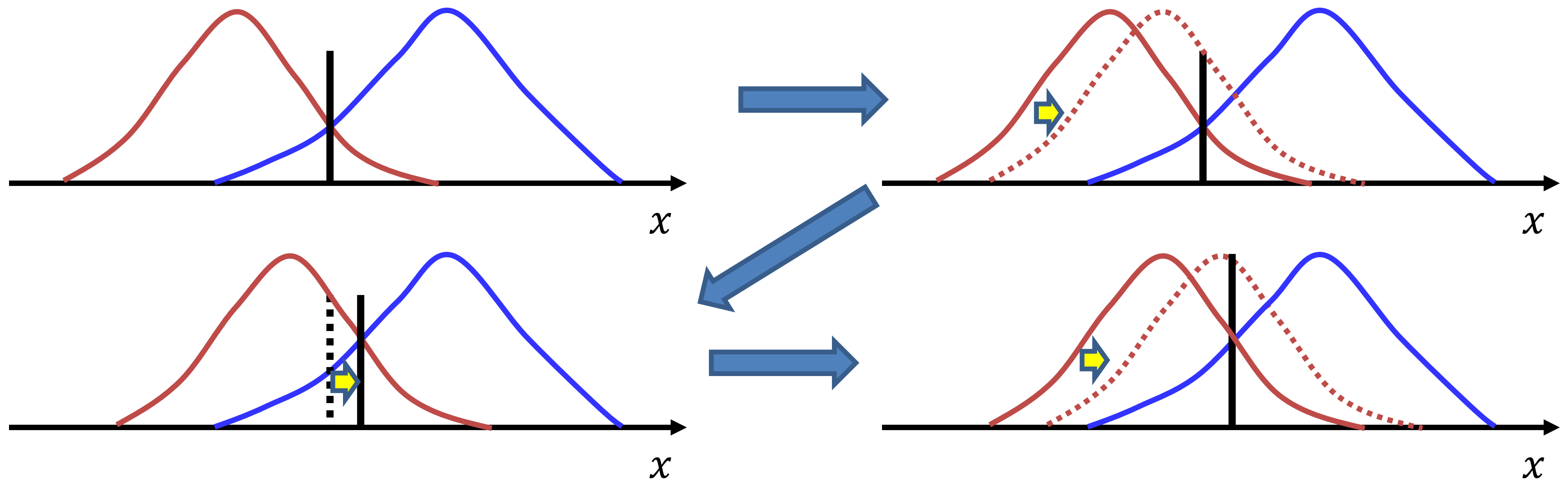


- The original GAN formulation is the following min-max optimization

$$\min_{G} \max_{D} \quad E_{\boldsymbol{x} \sim \boldsymbol{P_X}} \log D(x) + E_{\boldsymbol{x} \sim \boldsymbol{P_G}} \log(1 - D(x))$$

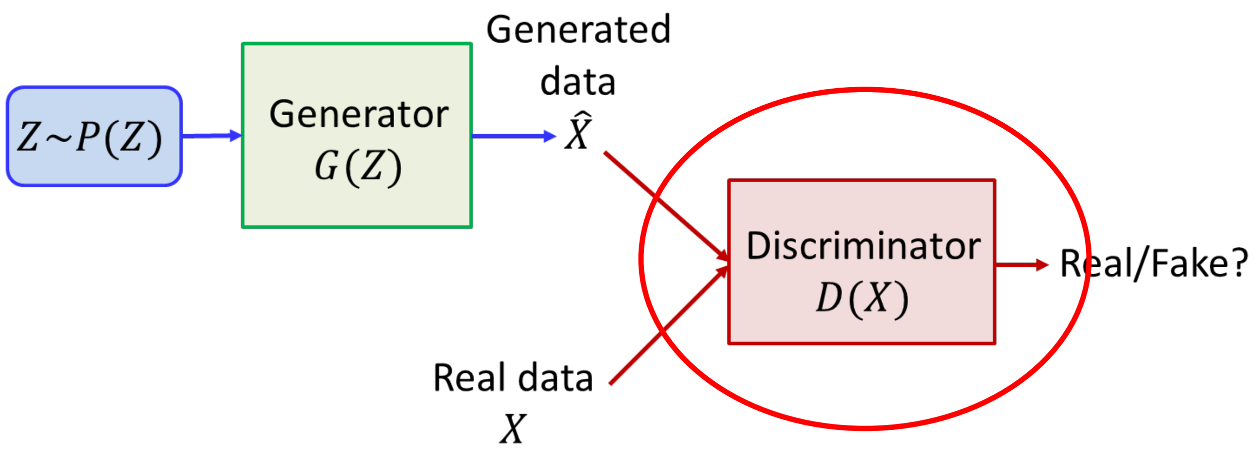
- Objective of D: $D(x) = 1$ and $D(G(z)) = 0$
- Objective of G: $D(G(z)) = 1$

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

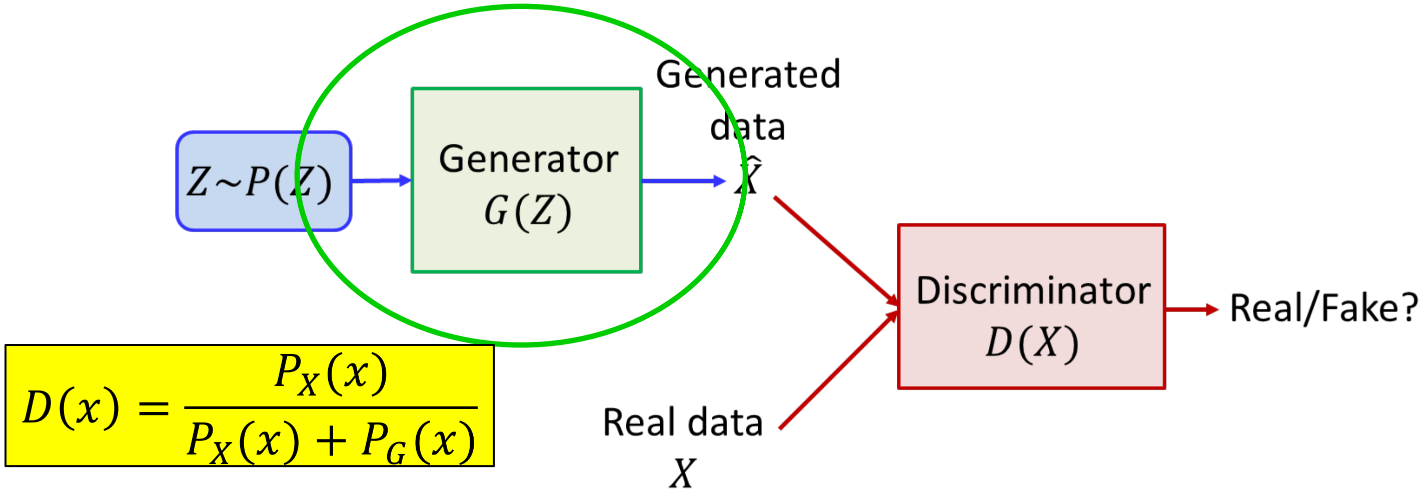
$$D(x) = \frac{P_X(x)}{P_X(x) + P_G(x)}$$

  – Assuming uniform prior

Black Board!

- Math at the end of the slide-deck

Analysis of optimal behavior: The optimal generator



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

- **With a perfect discriminator**:

$$L = \mathbb{E}_{x \sim P_X(x)} \log D(x) + \mathbb{E}_{x \sim P_G(x)} \log(1 - D(x))$$
$$= \mathbb{E}_{x \sim P_X(x)} \left[\log \left(\frac{P_X(x)}{P_X(x) + P_G(x)} \right) \right] + \mathbb{E}_{x \sim P_G(x)} \left[\log \left(\frac{P_G(x)}{P_X(x) + P_G(x)} \right) \right]$$

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

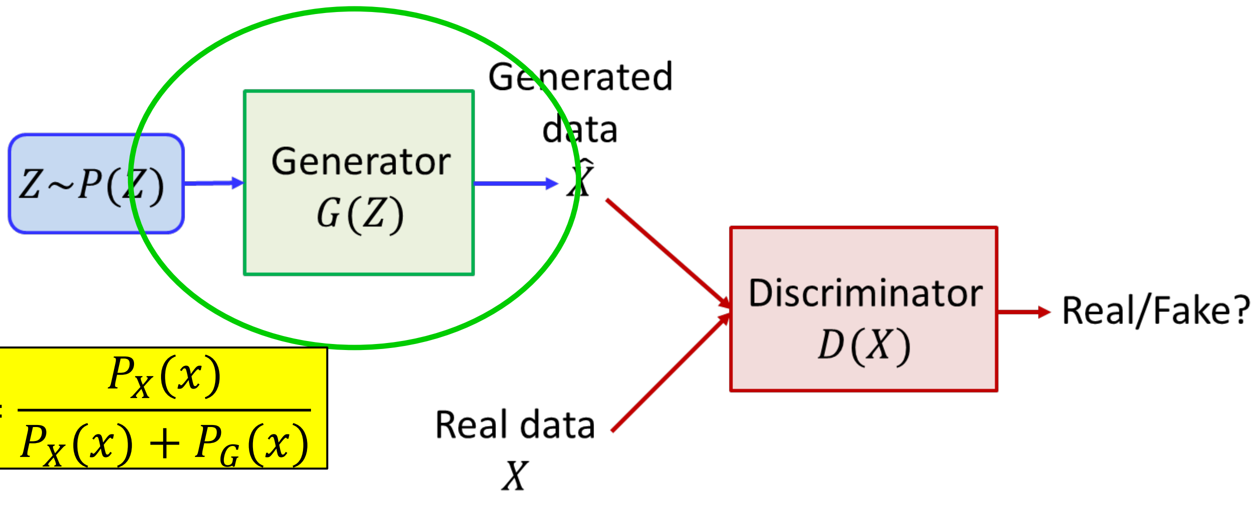
$$L = 2D_{JSD}\big(P_X(x), P_G(x)\big) - \log 4$$

The Jensen Shannon Divergence

$$D_{JSD}\,(P, Q) \;=\; \frac{D_{KL}\left(P, \frac{P+Q}{2}\right) + D_{KL}\left(Q, \frac{P+Q}{2}\right)}{2}$$

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

Analysis of optimal behavior: The optimal generator



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

- The optimal generator:

$$\min_G 2D_{JSD}\big(P_X(x), P_G(x)\big) - \log 4$$

- 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

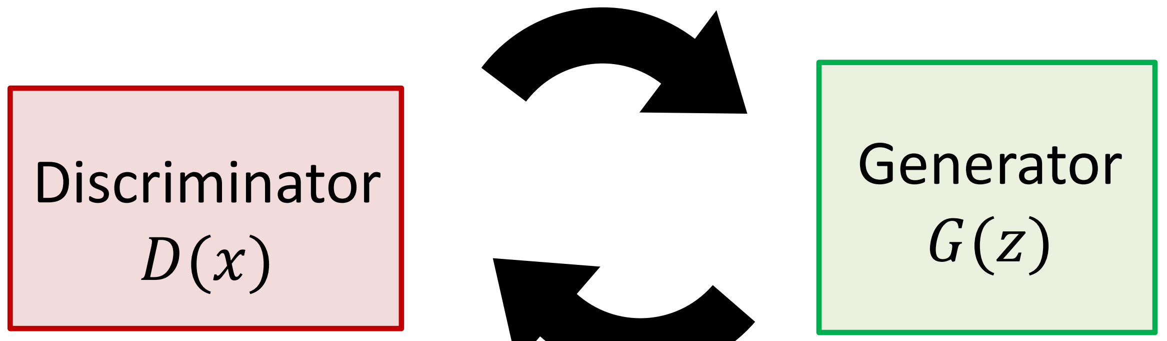
Min-Max Stationary Point

- 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: $\min_G \max_D \mathbb{E}_x \log D(x) + \mathbb{E}_z \log(1 - D(G(z)))$

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

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

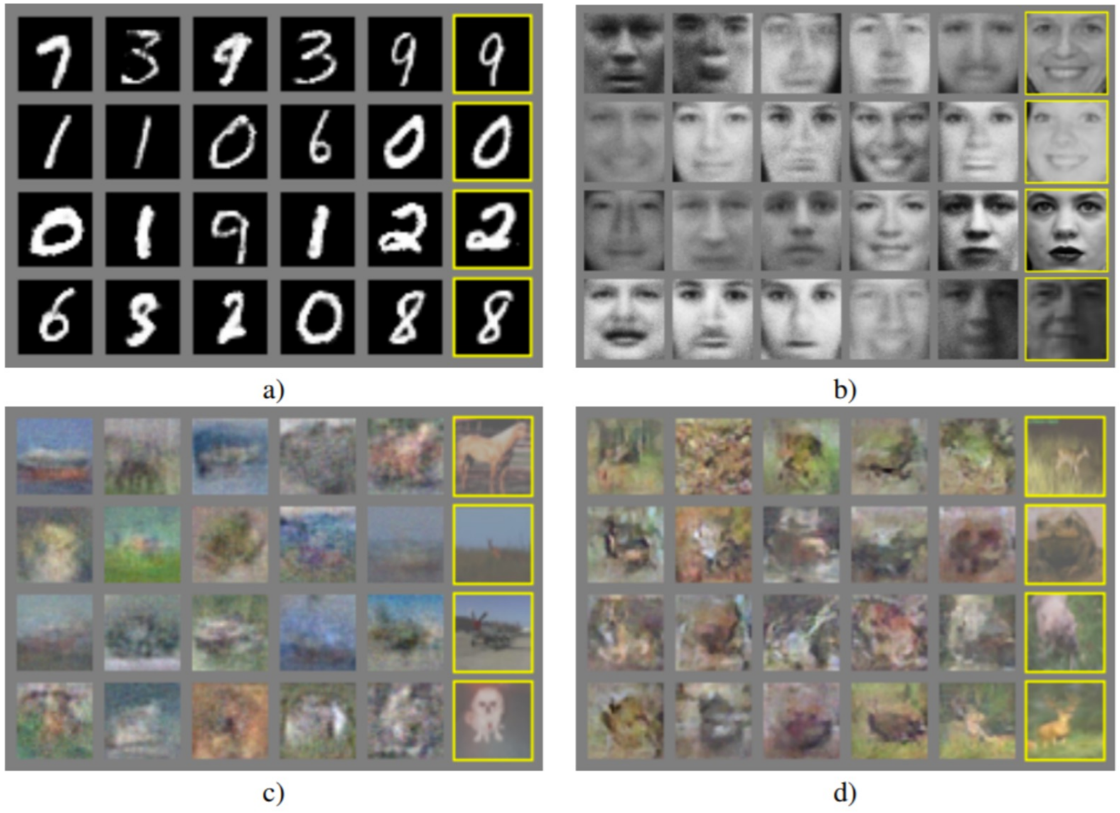
- Inception Score
  - Use the Inception V3 image classifier to classify generated images
  - Inception should produce a variety of labels
    - **As measured by the entropy of the average label distribution**
  - Each label should have high confidence (low entropy)
    - **As measured by the average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

VAEs vs GANs

VAEs	GANs
• Minimizing the KL divergence between distributions of synthetic and true data	• Minimizing the Jenson-Shannon divergence between distributions of synthetic and true data
• Uses an encoder to predict latent distributions to optimize generator	• Use a discriminator to optimize generator
• More complex formulation	• Simpler formulation
• Simpler optimization. Trains faster and more reliably	• Noisy and difficult optimization
• Results are blurry	• Sharper results

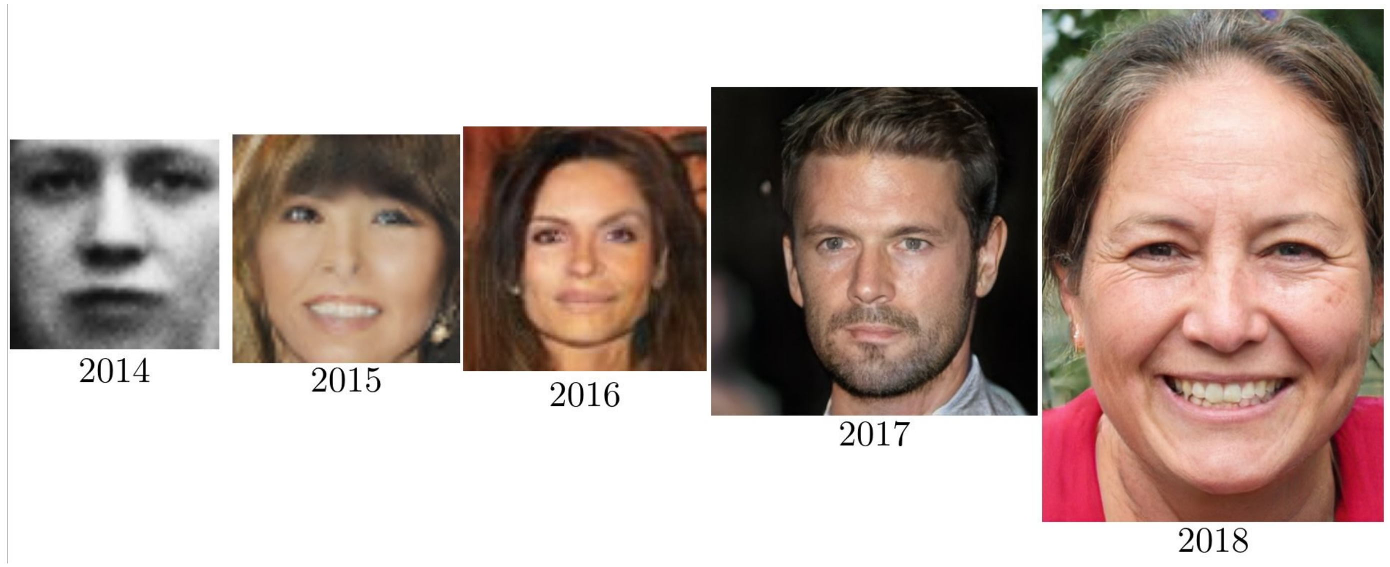
Original paper (GAN, 2014)

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]

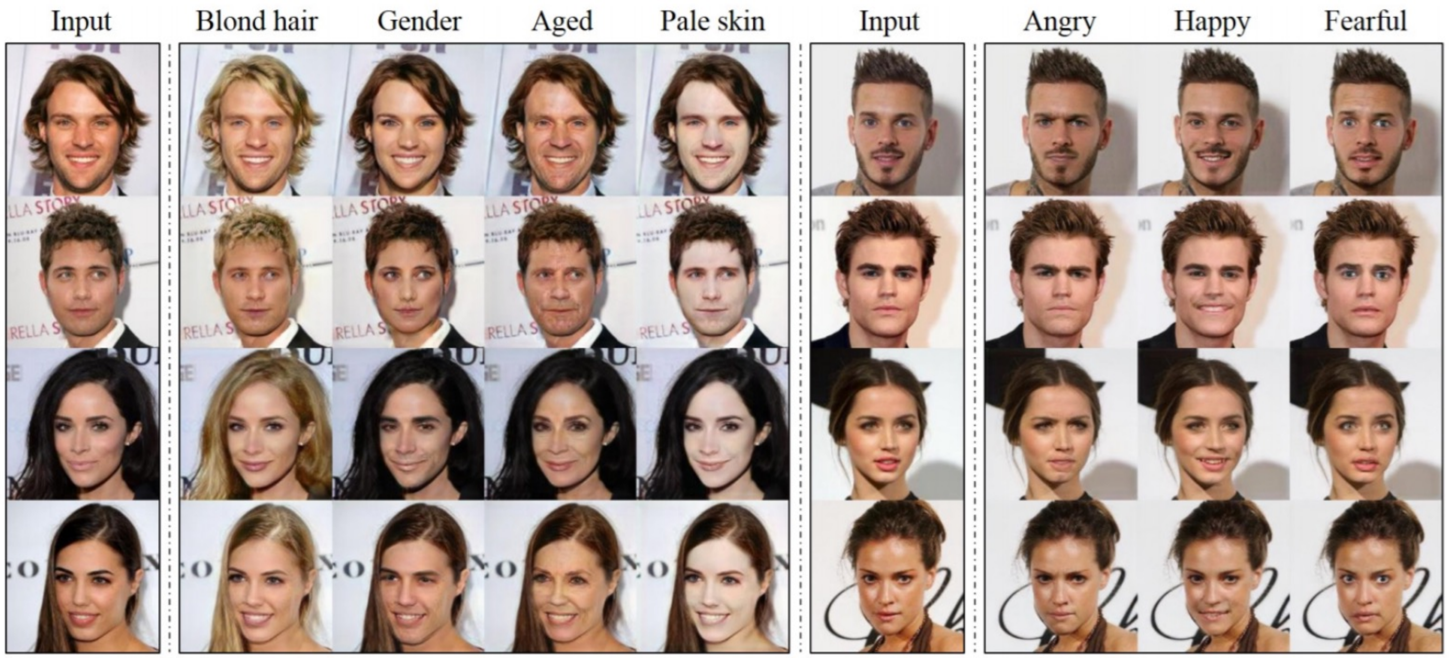


Figure 1. Multi-domain image-to-image translation results on the CelebA dataset via transferring knowledge learned from the RaFD dataset. The first and sixth columns show input images while the remaining columns are images generated by StarGAN. Note that the images are generated by a single generator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

Progressive growing of GANs (2018)

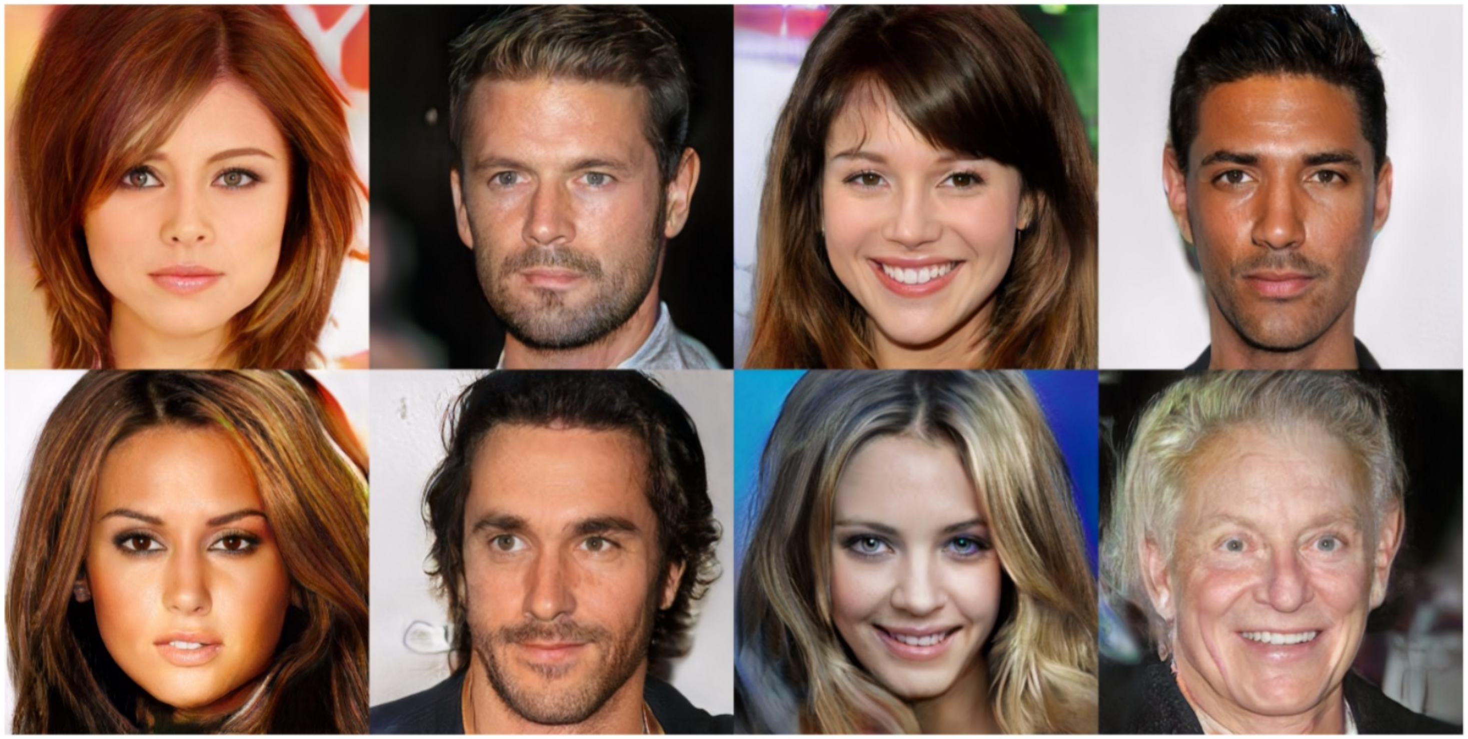


Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

High fidelity natural images (2019)

Generating High-Quality Images [BDS18]

Appendix

Math: Optimum Discriminator

$$\min_G \max_D \underbrace{\mathbb{E}_{x\sim P_x}[\log D(x)] + \mathbb{E}_{z\sim P_z}[\log(1-D(G(z)))]}$$

$$\Downarrow$$

$$= \int_x P_x(x)\log D(x)\,dx + \int_z P_z(z)\log(1-D(G(z)))\,dz$$

Use implicit model $P_G(x)$

$$= \int_x P_x(x)\log D(x)\,dx + \int_x P_G(x)\log(1-D(x))\,dx$$

$$= \int_x \left(P_x(x)\log D(x) + P_G(x)\log(1-D(x))\right)dx$$

$$\frac{d}{dD(x)}(\ldots) = 0$$ # find optimum

Math: Optimum Discriminator

$$\frac{d}{d\,D(x)}\left[P_x(x)\log(D(x)) + P_G(x)\log(1-D(x))\right] = 0$$

$$\Rightarrow \quad \frac{P_x(x)}{D(x)} - \frac{P_G(x)}{1-D(x)} = 0$$

$$\Rightarrow \quad \frac{P_x(x)}{D(x)} = \frac{P_G(x)}{1-D(x)} \Rightarrow P_x(x)\left(1-D(x)\right) = P_G(x)\,D(x)$$

$$\underset{\text{optimum}}{D(x)} = \frac{P_x(x)}{P_x(x)+P_G(x)} \Rightarrow \text{Bayes Optimum Classifier!}$$

Math: Optimum Generator

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

$\Rightarrow$ Plug this in for the generator for optimum generator!

$$\min_G \underbrace{\mathbb{E}_{x \sim P_X}[\log D(x)] + \mathbb{E}_{x \sim P_G}[\log(1 - D(x))]}$$

$$\Downarrow$$

$$\Rightarrow \mathbb{E}_{x \sim P_X}\left[\log \frac{P_X(x)}{P_X(x) + P_G(x)}\right] + \mathbb{E}_{x \sim P_G}\left[\log \frac{P_G(x)}{P_X(x) + P_G(x)}\right]$$

$$\boxed{+ \log 4 \quad - \log 4}$$

$$\Rightarrow \mathbb{E}_{x \sim P_X}\left[\log \frac{2 P_X(x)}{P_X(x) + P_G(x)}\right] + \mathbb{E}_{x \sim P_G}\left[\log \frac{2 P_G(x)}{P_X(x) + P_G(x)}\right] - \log 4$$

Math: Optimum Generator

$$\Rightarrow \mathbb{E}_{x\sim P_X}\left[\log \frac{2P_X(x)}{P_X(x)+P_G(x)}\right] + \mathbb{E}_{x\sim P_G}\left[\log \frac{2P_G(x)}{P_X(x)+P_G(x)}\right] - \log 4$$

$$= \mathbb{E}_{x\sim P_X}\left[\log \frac{P_X(x)}{\frac{P_X(x)+P_G(x)}{2}}\right] + \mathbb{E}_{x\sim P_G}\left[\log \frac{P_G(x)}{\frac{P_X(x)+P_G(x)}{2}}\right] - \log 4$$

$$= D_{KL}\left(P_X(x), \frac{P_X(x)+P_G(x)}{2}\right) + D_{KL}\left(P_G(x), \frac{P_X(x)+P_G(x)}{2}\right) - \log 4$$

$$D_{JSD}(P,Q) = \frac{D_{KL}\left(P, \frac{P+Q}{2}\right) + D_{KL}\left(Q, \frac{P+Q}{2}\right)}{2}$$ JS Divergence

$$D_{KL}\left(P, \frac{P+Q}{2}\right) = \mathbb{E}_P\left[\log \frac{P}{(P+Q)/2}\right]$$ KL Divergence

Math: Optimum Generator

$$= \frac{2}{2}\left(D_{KL}\left(P_x(x), \frac{P_x(x)+P_G(x)}{2}\right) + D_{KL}\left(P_G(x), \frac{P_x(x)+P_G(x)}{2}\right)\right) - \log 4$$

$$= 2\, D_{JSD}\left(P_x(x), P_G(x)\right) - \log 4$$

The optimum generator tries to minimize D_{JSD} !

$$\min_G \; 2\, D_{JSD}\left(P_x(x), P_G(x)\right) - \log 4$$

Detour: Information

$$\boldsymbol{I}(\boldsymbol{x}) = -\log(\boldsymbol{P}(\boldsymbol{x}))$$

Information is more where probability is low

“Today I didn’t see a tornado”

“Today I saw a tornado”

Detour: Information

$$\boldsymbol{I}(\boldsymbol{x}) = -\log(\boldsymbol{P}(\boldsymbol{x}))$$

Information is more where probability is low

“Today I didn’t see a tornado” – low information

“Today I saw a tornado” – high information

Detour: Entropy

$$\boldsymbol{H}(\boldsymbol{x}) = -\sum \boldsymbol{P}(\boldsymbol{x}) \log\left(\boldsymbol{P}(\boldsymbol{x})\right)$$
$$\boldsymbol{H}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{x} \sim \boldsymbol{X}}[-\log(\boldsymbol{P}(\boldsymbol{x}))]$$

Measure of uncertainty – weighted average of information

Detour on a Detour: Softmax

$$\boldsymbol{softmax(x)} = \frac{\boldsymbol{exp(x)}}{\boldsymbol{\sum exp(x)}}$$

Gives a probability distribution.

Detour on a Detour: Softmax

$$\boldsymbol{softmax(x)} = \frac{\boldsymbol{exp(x)}}{\boldsymbol{\sum exp(x)}}$$

$$\boldsymbol{Boltzmann(x,\tau)} = \frac{\boldsymbol{exp\left(\frac{x}{\tau}\right)}}{\boldsymbol{\sum exp\left(\frac{x}{\tau}\right)}}$$

$\boldsymbol{\tau}$ is called a ‘temperature’ parameter.

Detour on a Detour: Softmax

$$\boldsymbol{Boltzmann(x,\tau)} = \frac{\boldsymbol{exp}\left(\frac{\boldsymbol{x}}{\boldsymbol{\tau}}\right)}{\sum \boldsymbol{exp}\left(\frac{\boldsymbol{x}}{\boldsymbol{\tau}}\right)}$$

High value of $\boldsymbol{\tau}$ causes distribution to be more uniform – high entropy – “hot”

Low value “cools” the distribution – low entropy

Detour on a Detour: Softmax

$$\boldsymbol{Boltzmann(x,\tau) = \frac{exp\left(\frac{x}{\tau}\right)}{\sum exp\left(\frac{x}{\tau}\right)}}$$

Try plotting using numpy and matplotlib for various values of $\boldsymbol{\tau}$ `[1e-3, 1e-2,1e-1,1,10,100]`!

You might want to use stable-softmax.
(look it up!)

Detour: Entropy

$$\boldsymbol{H}(\boldsymbol{x}) = -\sum \boldsymbol{P}(\boldsymbol{x}) \log\left(\boldsymbol{P}(\boldsymbol{x})\right)$$
$$\boldsymbol{H}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{x} \sim \boldsymbol{X}}[-\log(\boldsymbol{P}(\boldsymbol{x}))]$$

Measure of uncertainty – weighted average of information