Lecture 18: Deep Generative Models IV: Generative Adversarial Network (GAN)

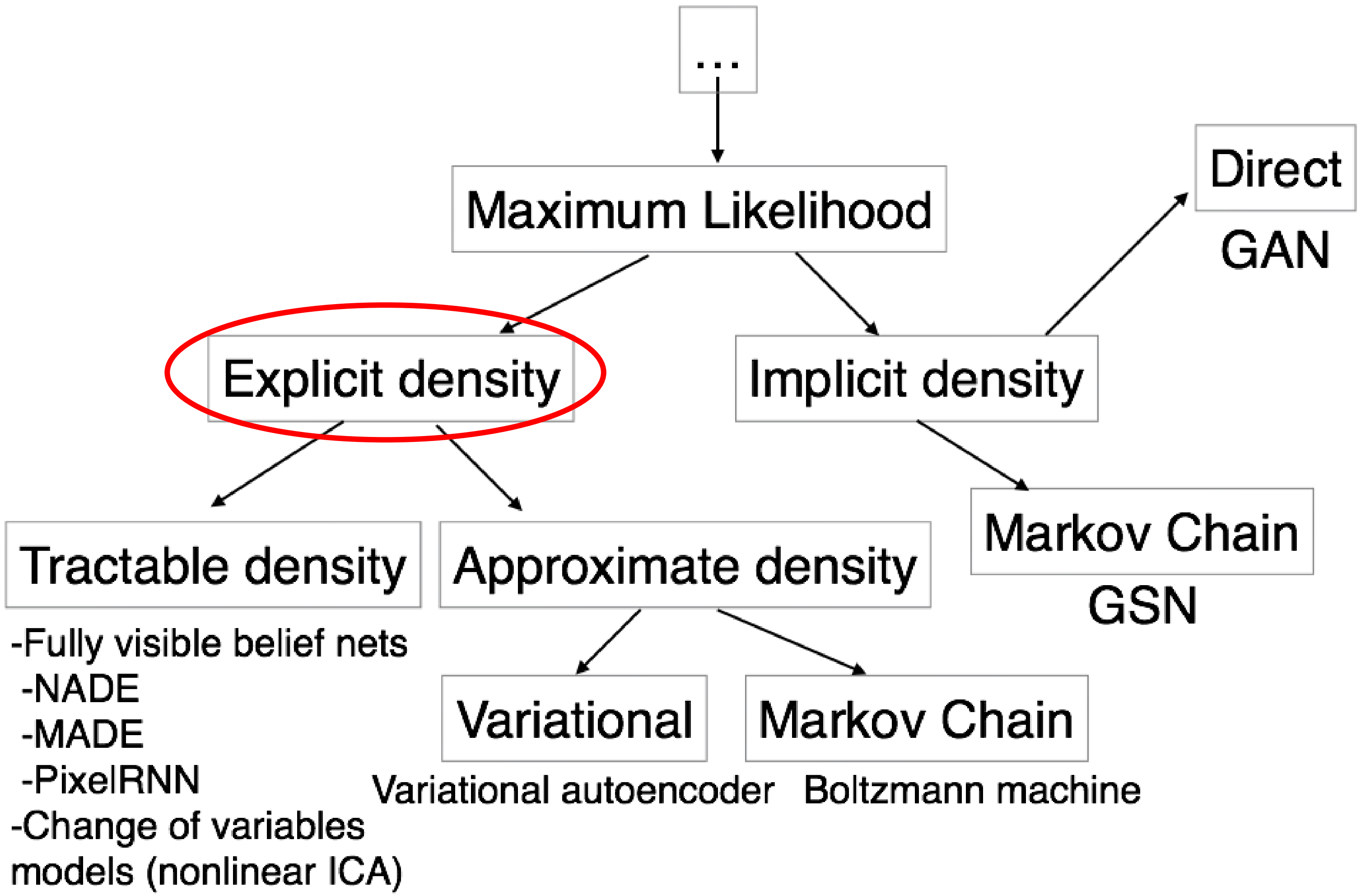
Lan Xu
SIST, ShanghaiTech
Fall, 2021

Outline

- Generative Adversarial Networks
  - Implicit generative models
  - Adversarial learning
  - Evaluation metrics

Acknowledgement: Feifei Li et al's cs231n notes

Taxonomy of Generative Models



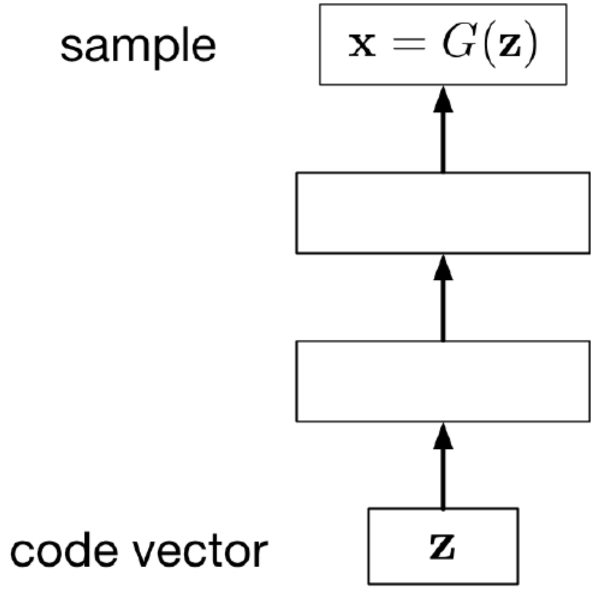
Implicit Generative Models

- Working with explicit model p(x) could be expensive
  - Variational Autoencoder (variational inference)
  - Boltzmann Machines (MCMC, not discussed)

- Representation learning may not require p(x)
  - Sometimes we are more interested in taking samples from p(x) instead of p itself

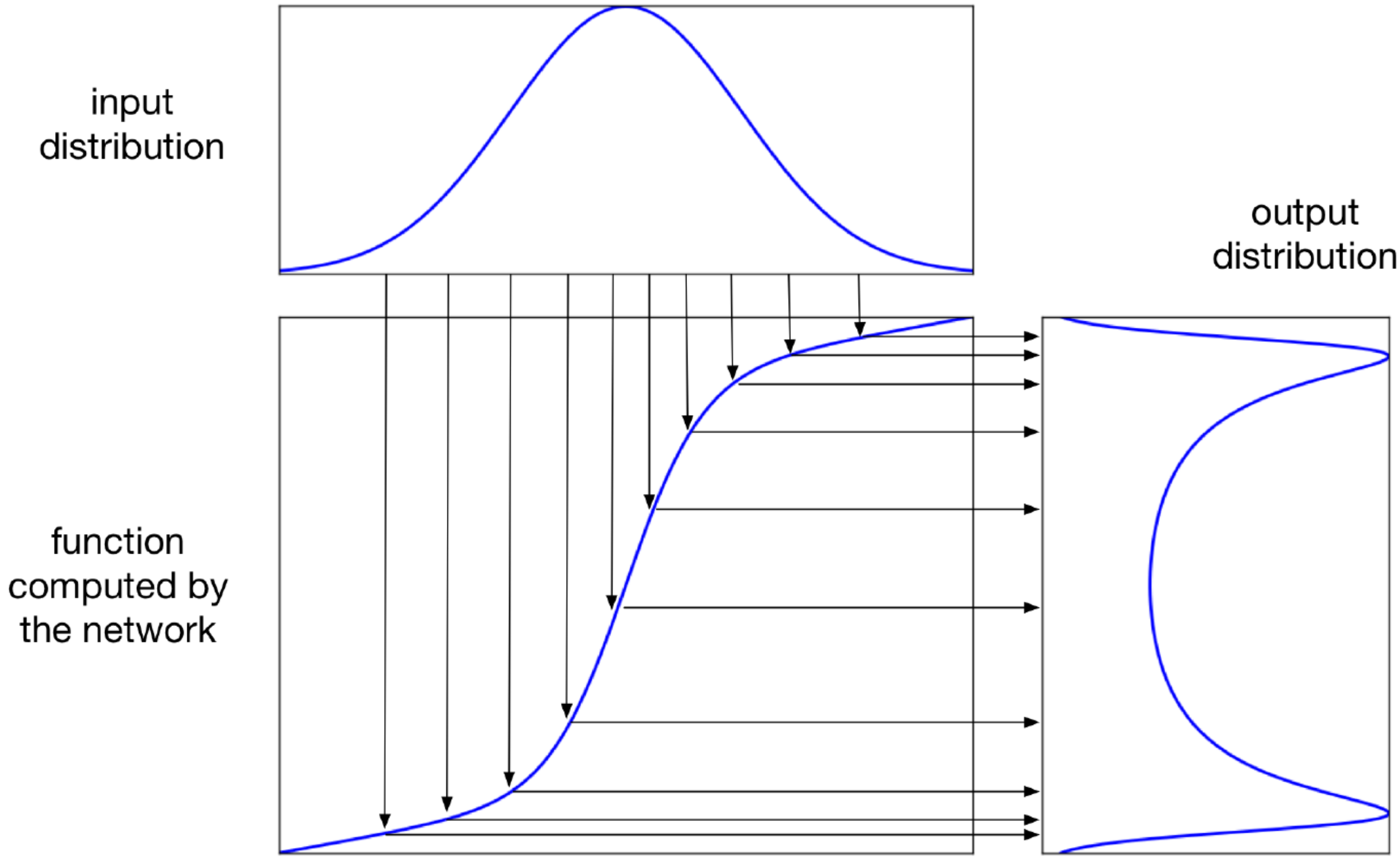
Implicit Generative Models

- Implicitly define a probability distribution
- Start by sampling the code vector z from a fixed, simple distribution
- A generator network computes a differentiable function G mapping z to an x in data space



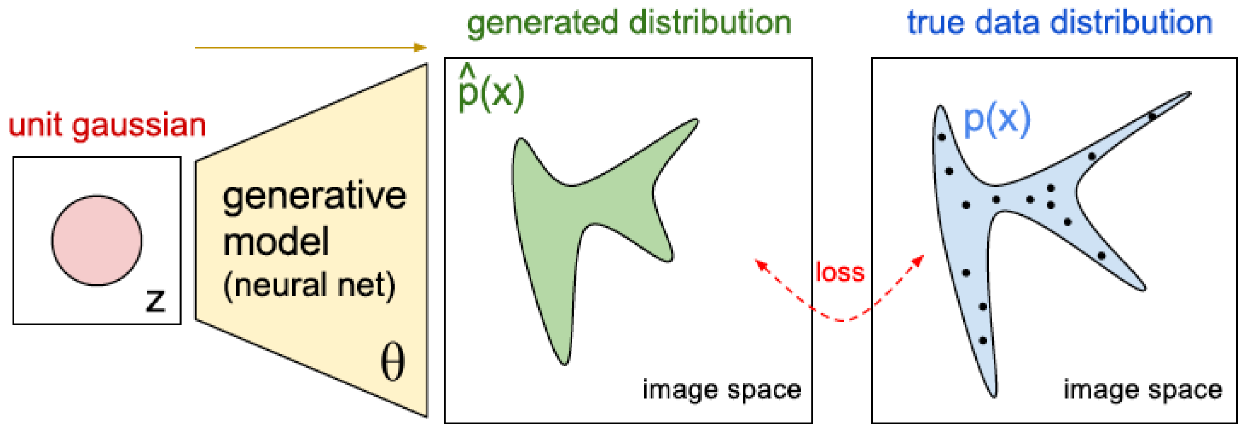
Implicit Generative Models

- Intuition: 1D example



Implicit Generative Models

- Intuition

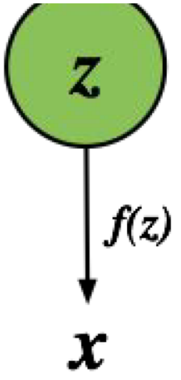


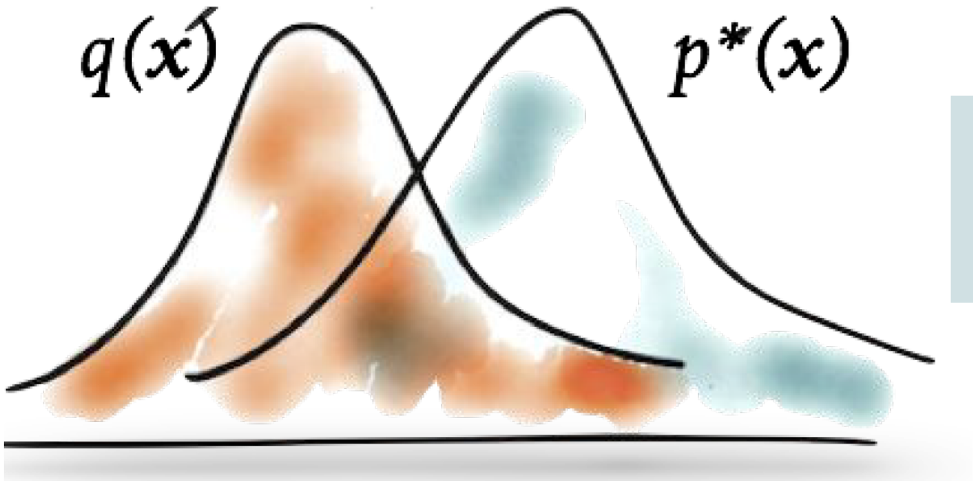
advocate/penalize samples within the blue/white region.

Learning by comparison

- Basic idea

For some models, we only have access to an unnormalised probability, partial knowledge of the distribution, or a simulator of data.





We compare the estimated distribution q(x) to the true distribution p*(x) using samples.

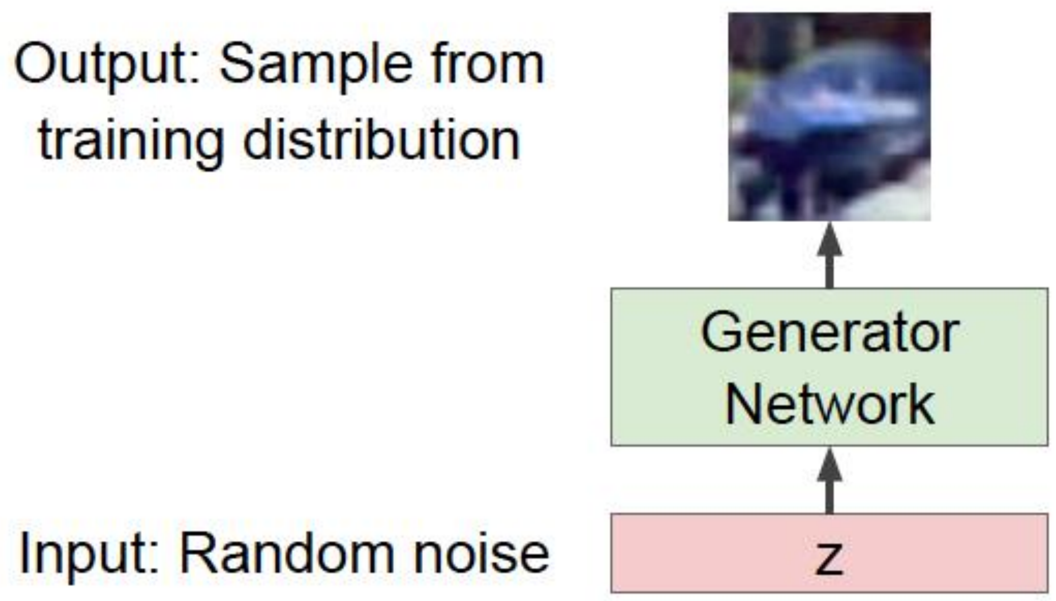
Outline

- Generative Adversarial Networks
  - Implicit generative models
  - Adversarial learning
  - Evaluation metrics

Acknowledgement: Feifei Li et al's cs231n notes

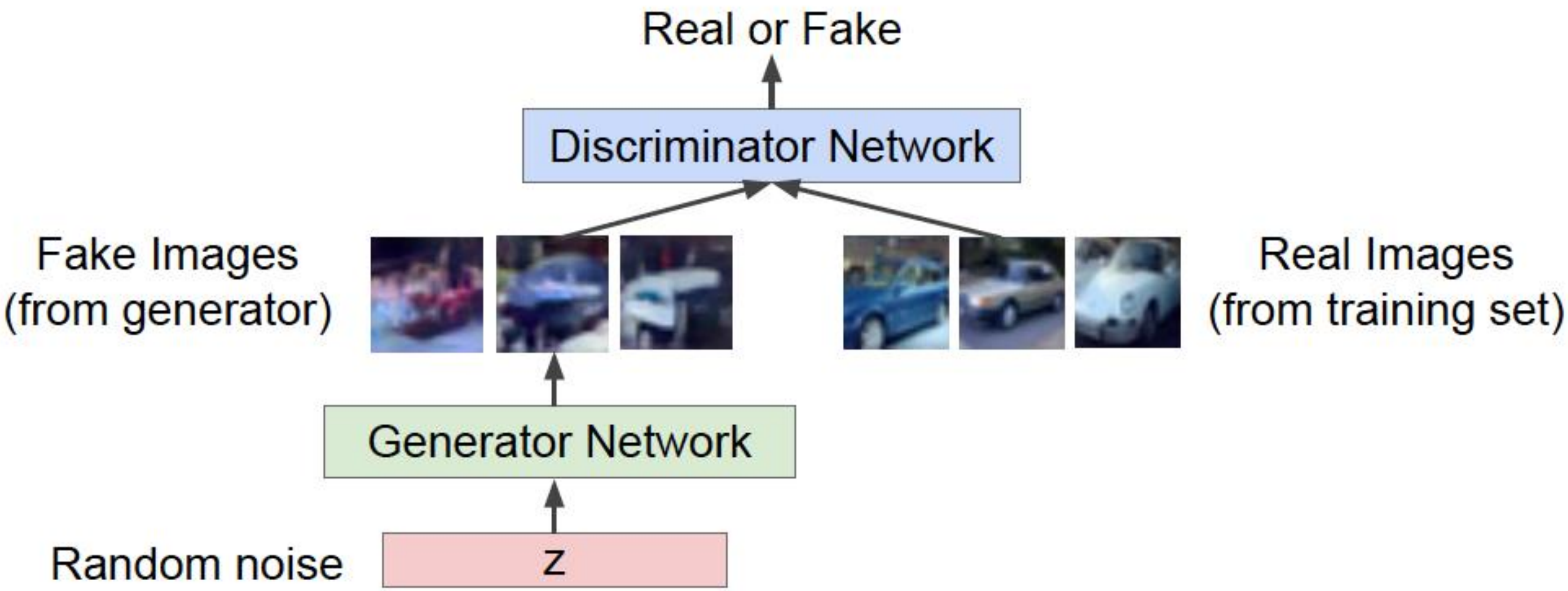
Generative Adversarial Networks

- Using a neural network to generate data



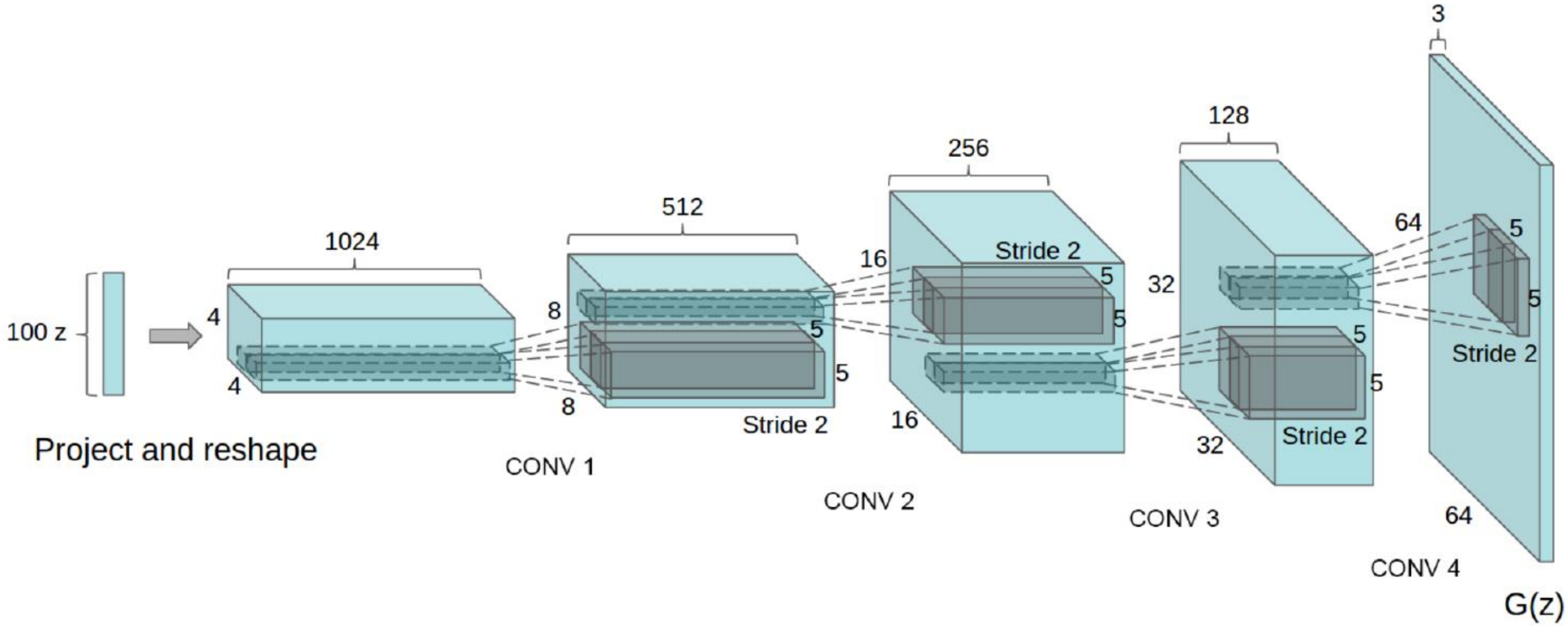
Generative Adversarial Networks

- Using another neural network to determine if the data is real or not



Typical generator architecture

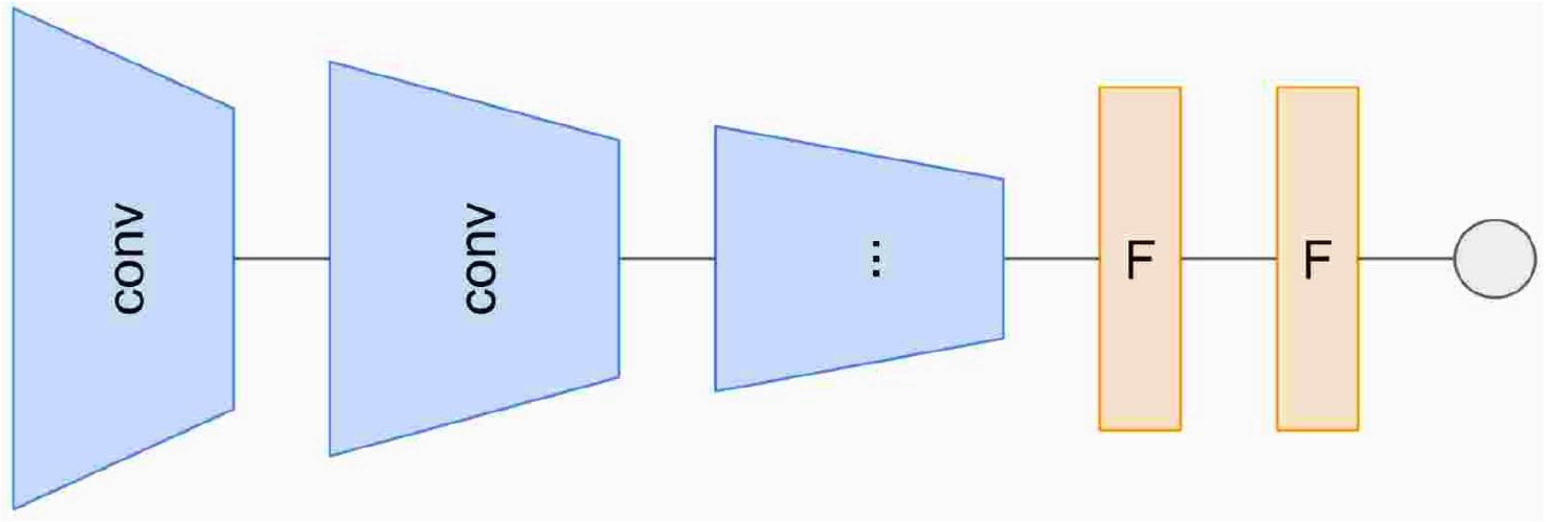
- For images



- Unit Gaussian distribution on z, typically 10-100 dim.
- Up-convolutional deep network (reverse recognition CNN)

Typical discriminator architecture

- For images



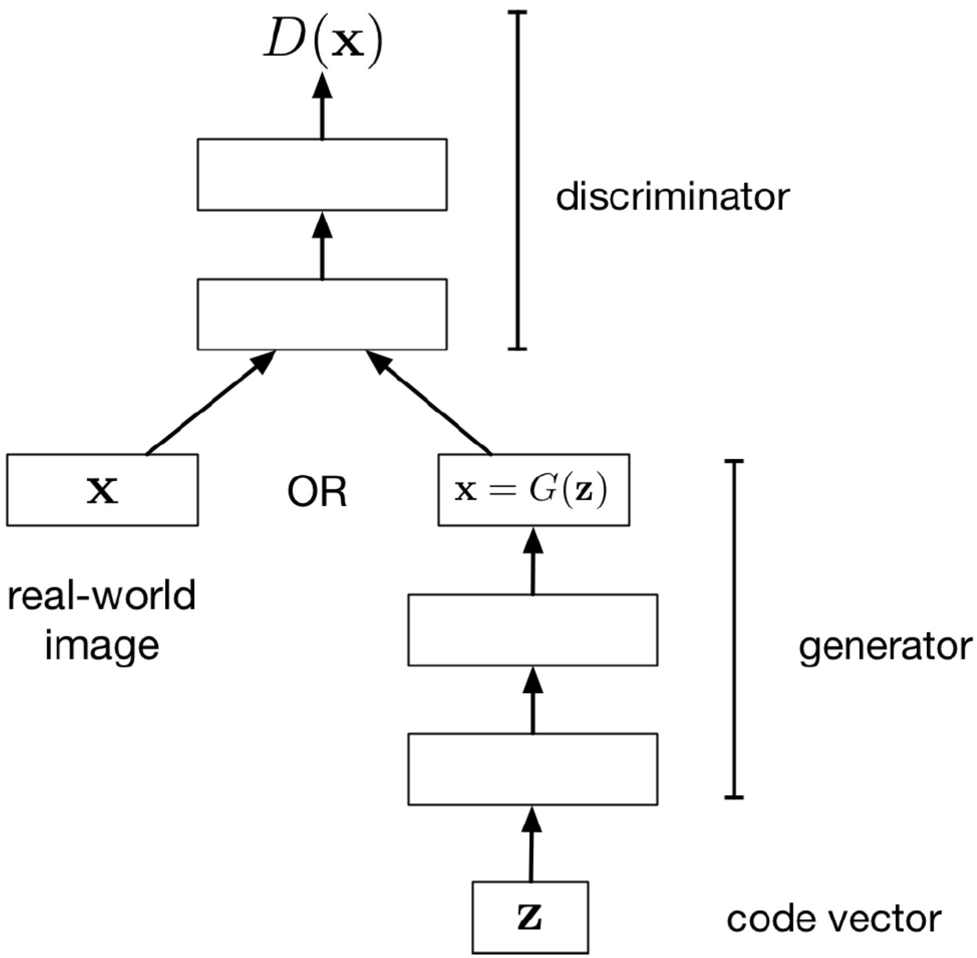
- Recognition CNN model
- Binary classification output: real / synthetic

Adversarial learning

- GAN objective for the generator is some complicated objective function defined by a neural network.
  - This means a new way of thinking about "distance".
  - We are training networks to minimize the "distance" or "divergence" between generated images and real images.
  - Instead of some hand-crafted distance metric like L1 or L2, we can make something completely new.
  - *A neural network, with the right architecture, is arguably the definition of perceptual similarity (assuming our visual system is some sort of neural network).*

Adversarial Learning

- Adversarial loss



Adversarial Learning

- Let D denote the discriminator's predicted probability of being real data
- Discriminator's cost function: cross-entropy loss for task of classifying real vs. fake images

$$\mathcal{J}_D = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[-\log(1 - D(G(\mathbf{z})))]$$

- One possible cost function for the generator: the opposite of the discriminator's

$$\begin{aligned}\mathcal{J}_G &= -\mathcal{J}_D \\ &= \text{const} + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]\end{aligned}$$

Two-player game

- Minimax formulation
  - The generator and discriminator are playing a zero-sum game against each other

$$\min_G \max_D \mathcal{J}_D$$

  - Using parametric models

Discriminator outputs likelihood in (0,1) of real image

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

Learning procedure

- Minimax objective function

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

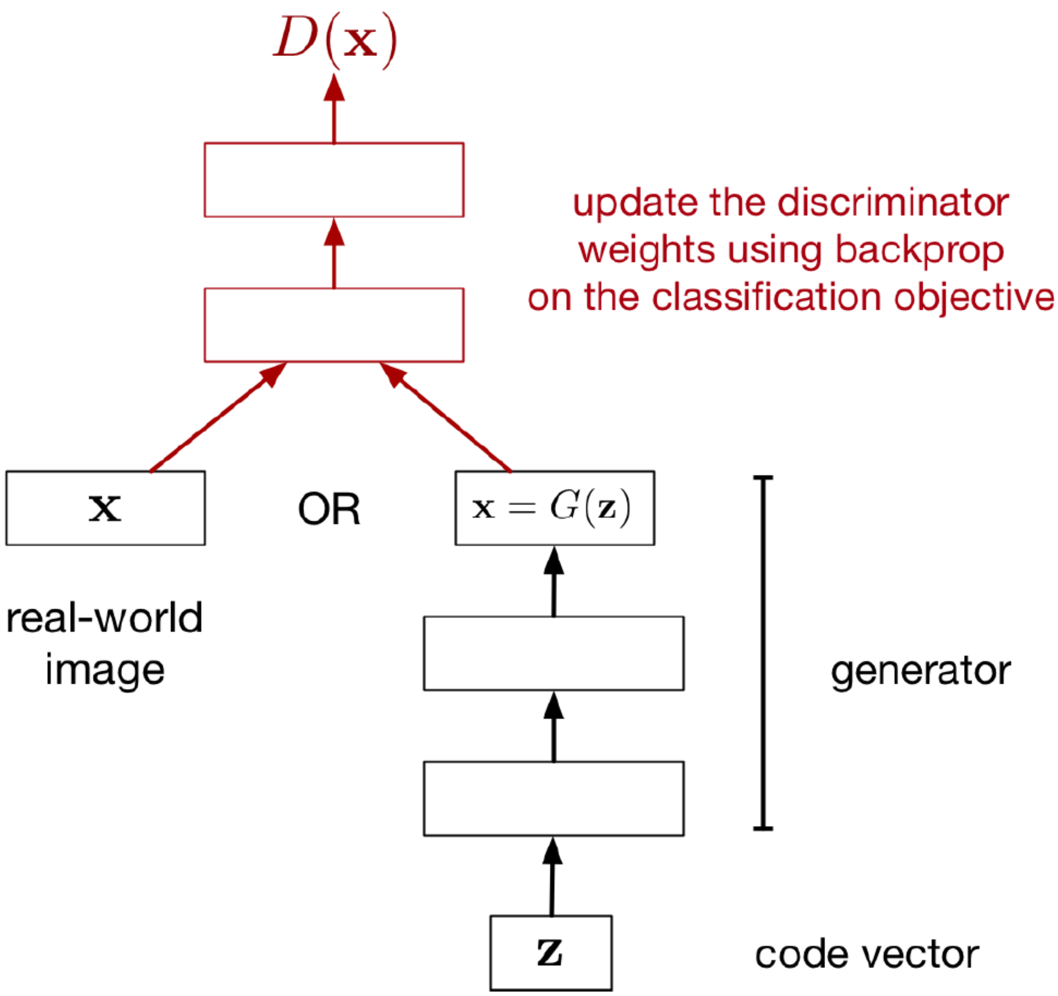
$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

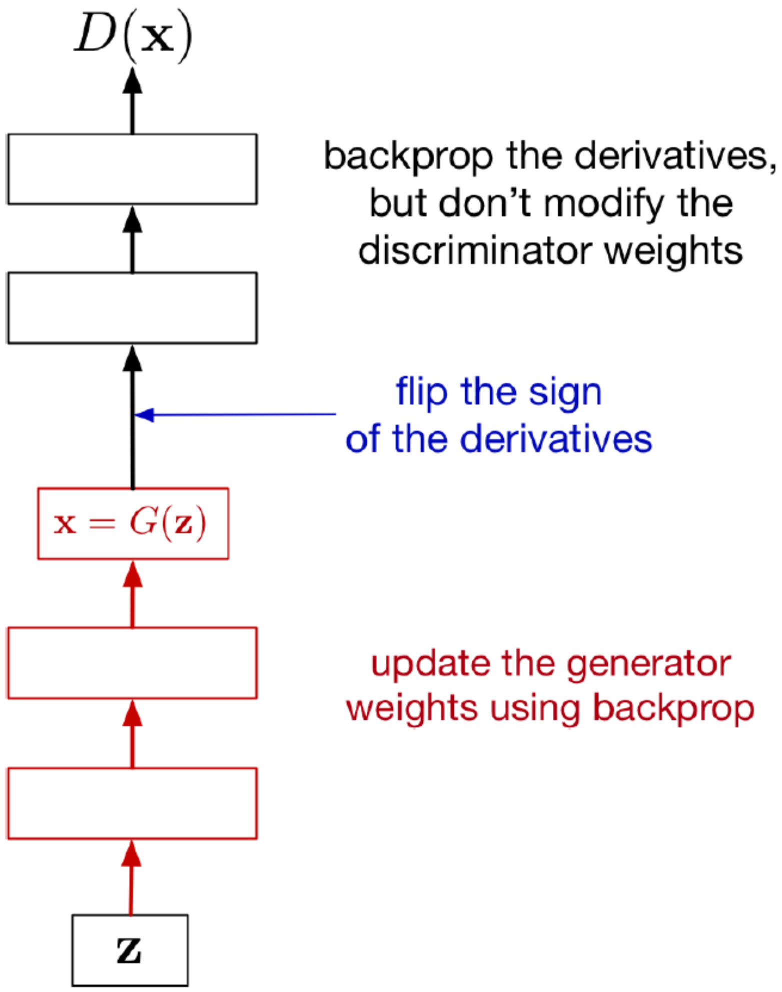
Learning procedure

- Updating the discriminator



Learning procedure

- Updating the generator



A better cost function

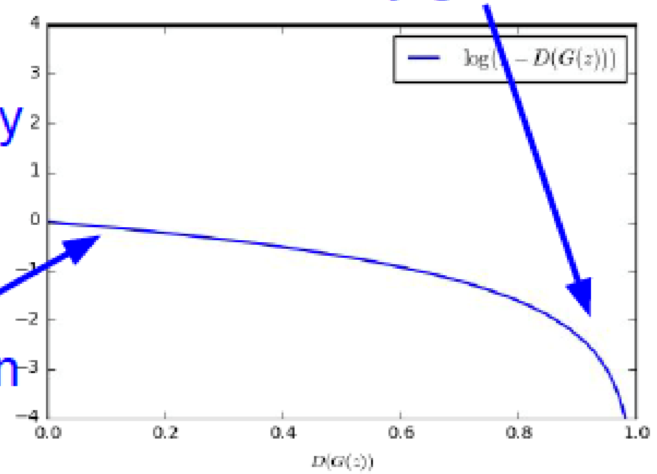
- The minimax cost function for the generator

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

- One problem is saturation

Gradient signal dominated by region where sample is already good

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!



A better cost function

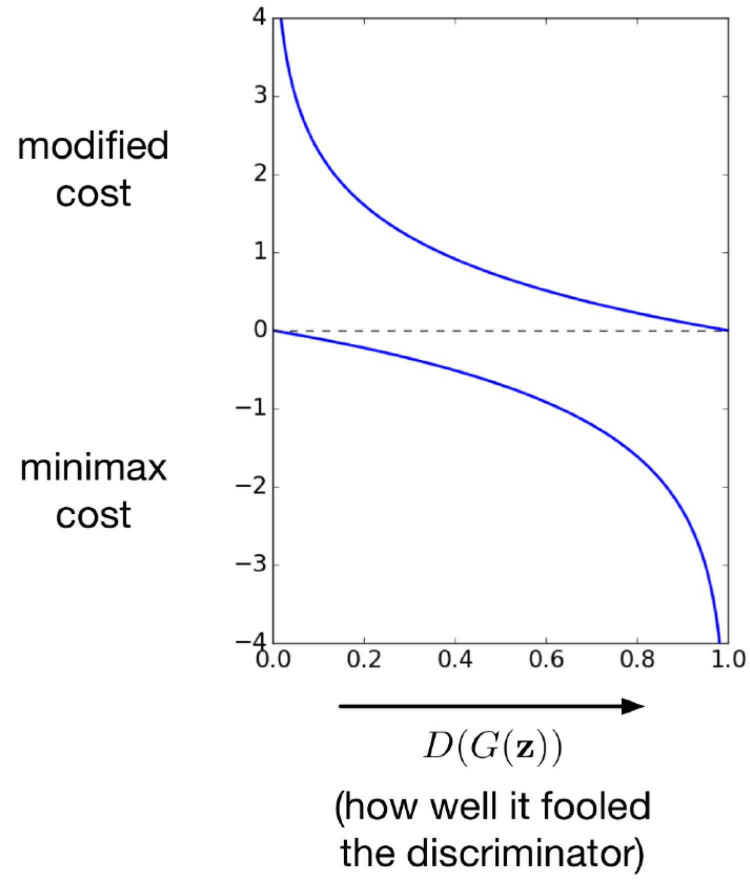
- Changing the generator cost
  - Original minimax cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]$$

  - Modified generator cost:

$$\mathcal{J}_G = \mathbb{E}_{\mathbf{z}}[-\log D(G(\mathbf{z}))]$$

  - This fixes the saturation problem.



Theoretical property

- Adversarial loss

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim data}\log D(x) - \frac{1}{2}\mathbb{E}_z \log(1 - D(G(z))) \qquad (1)$$

$$J^{(G)} = -J^{(D)} \qquad (2)$$

- The optimal discriminator $D^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_{model}(x)}$.
- In this case, $J^{(G)} = 2D_{JS}(p_{data}||p_{model}) + const.$
- Jenson-Shannon divergence:
  $D_{JS}(p||q) = \frac{1}{2}D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2}D_{KL}(q||\frac{p+q}{2})$.

Theoretical property

- Stationary point

There is a theoretical point in this game at which the game will be stable and both players will stop changing.

- If the generated data exactly matches the distribution of the real data, the generator should output 0.5 for all points (argmax of loss function)
- If the discriminator is outputting a constant value for all inputs, then there is no gradient that should cause the generator to update

We rarely reach a completely stable point in practice due to practical issues

Theoretical property

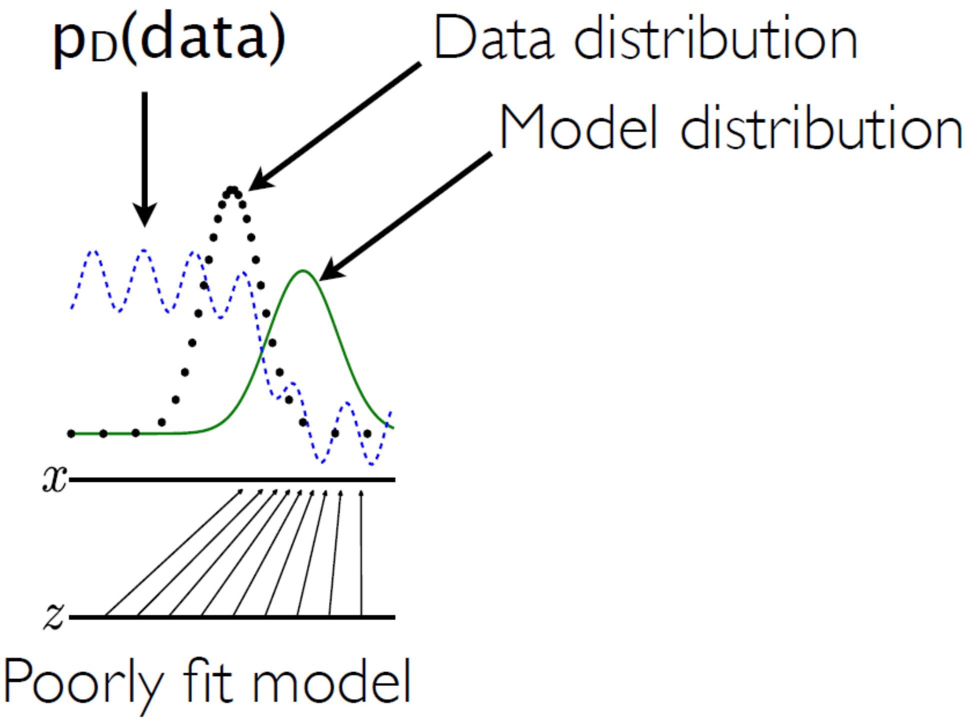
- Convergence

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

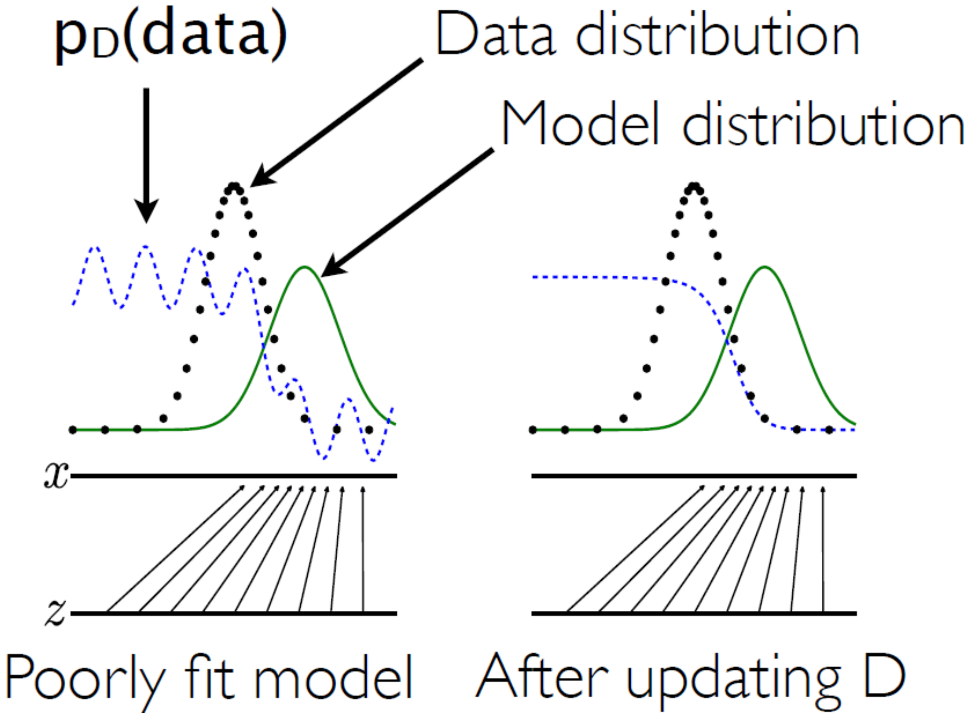
- Theoretical properties (assuming infinite data, infinite model capacity, direct updating of generator's distribution):
  - Unique global optimum.
  - Optimum corresponds to data distribution.
  - Convergence to optimum guaranteed.

If discriminator is finite and modest-sized, this message is incorrect. (regardless of training time, # samples, training objective etc..) See Sanjeev Arora, CVPR 2018 Tutorial

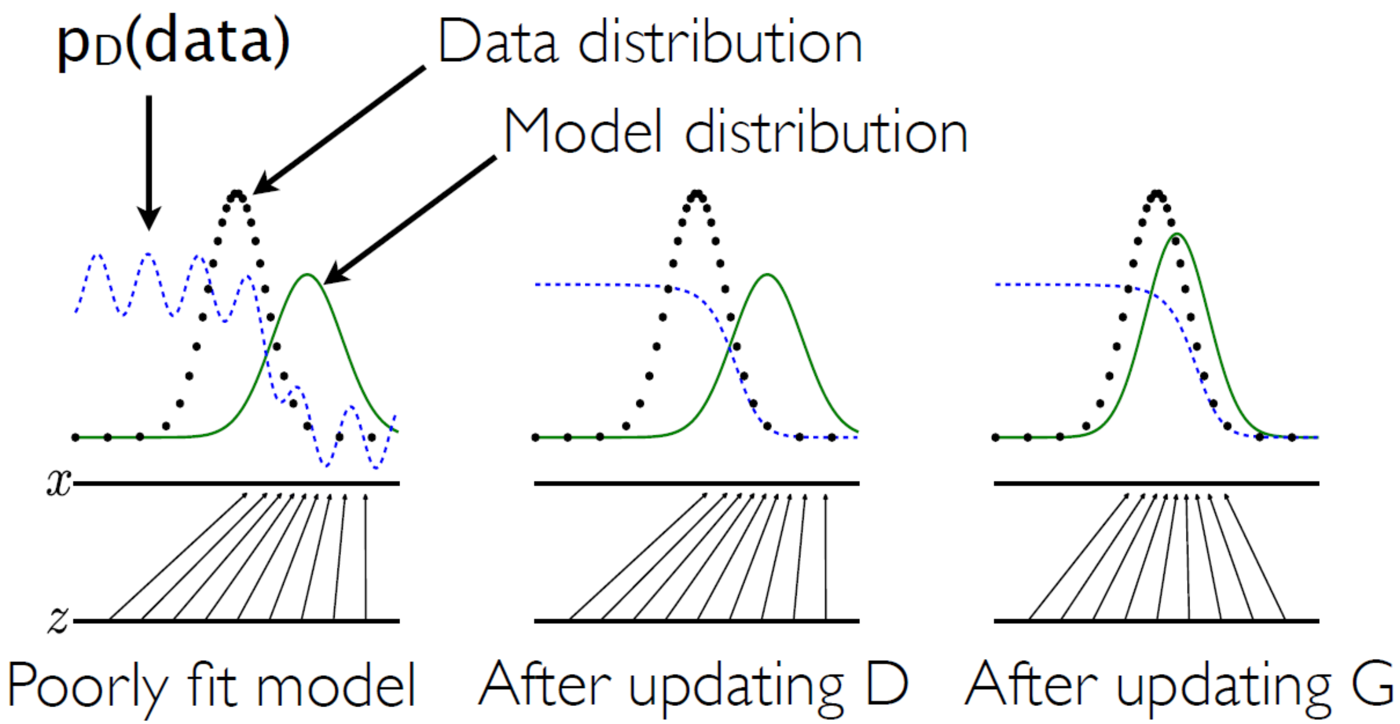
Training GANs



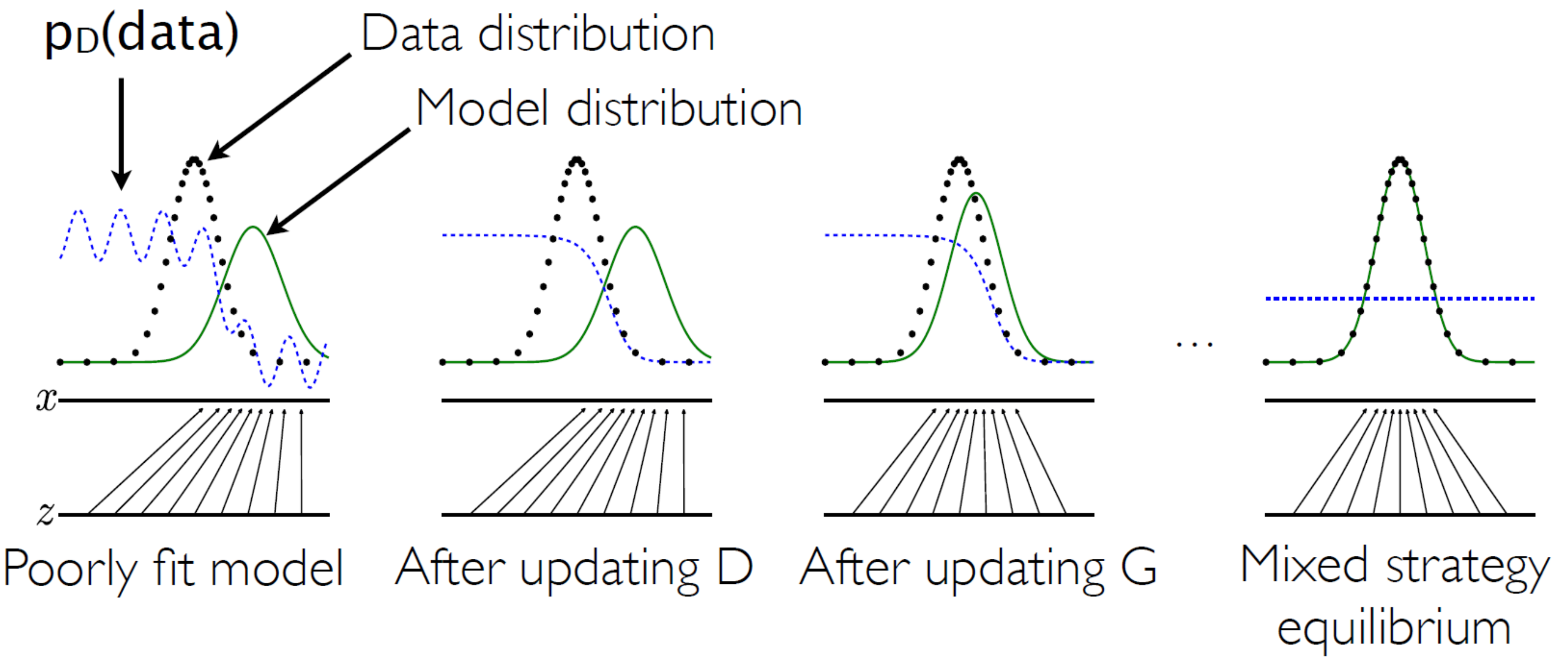
Training GANs



Training GANs



Training GANs



Training GANs

- Since GANs were introduced in 2014, there have been hundreds of papers introducing various architectures and training methods
- GAN Zoo: https://github.com/hindupuravinash/the-gan-zoo
- In general, training a GAN is tricky and unstable
- Many tricks:
  - S. Chintala, How to train a GAN, ICCV 2017 tutorial
  - https://github.com/soumith/talks/blob/master/2017-ICCV_Venice/How_To_Train_a_GAN.pdf

Generated Samples

Celebrities:

Karras et al., 2017. Progressive growing of GANs for improved quality, stability, and variation

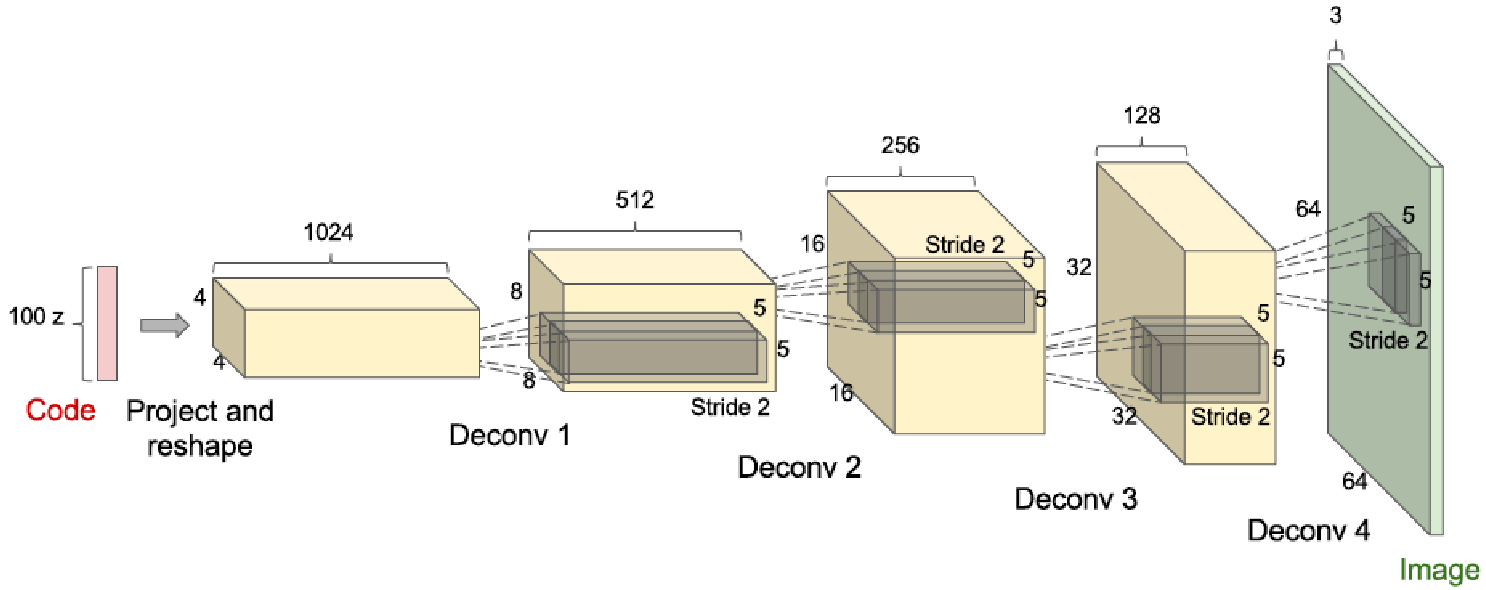
Generated Samples

Objects:

DCGAN

- GAN with convolutional architetures
  - Generator is an upsampling convolutional network
  - Discriminator is a convolutional network

Deep Convolutional GAN [Radford et al., 2015]



Generated Samples

Generated Bad Samples

- Problems with Global Structure and Counting

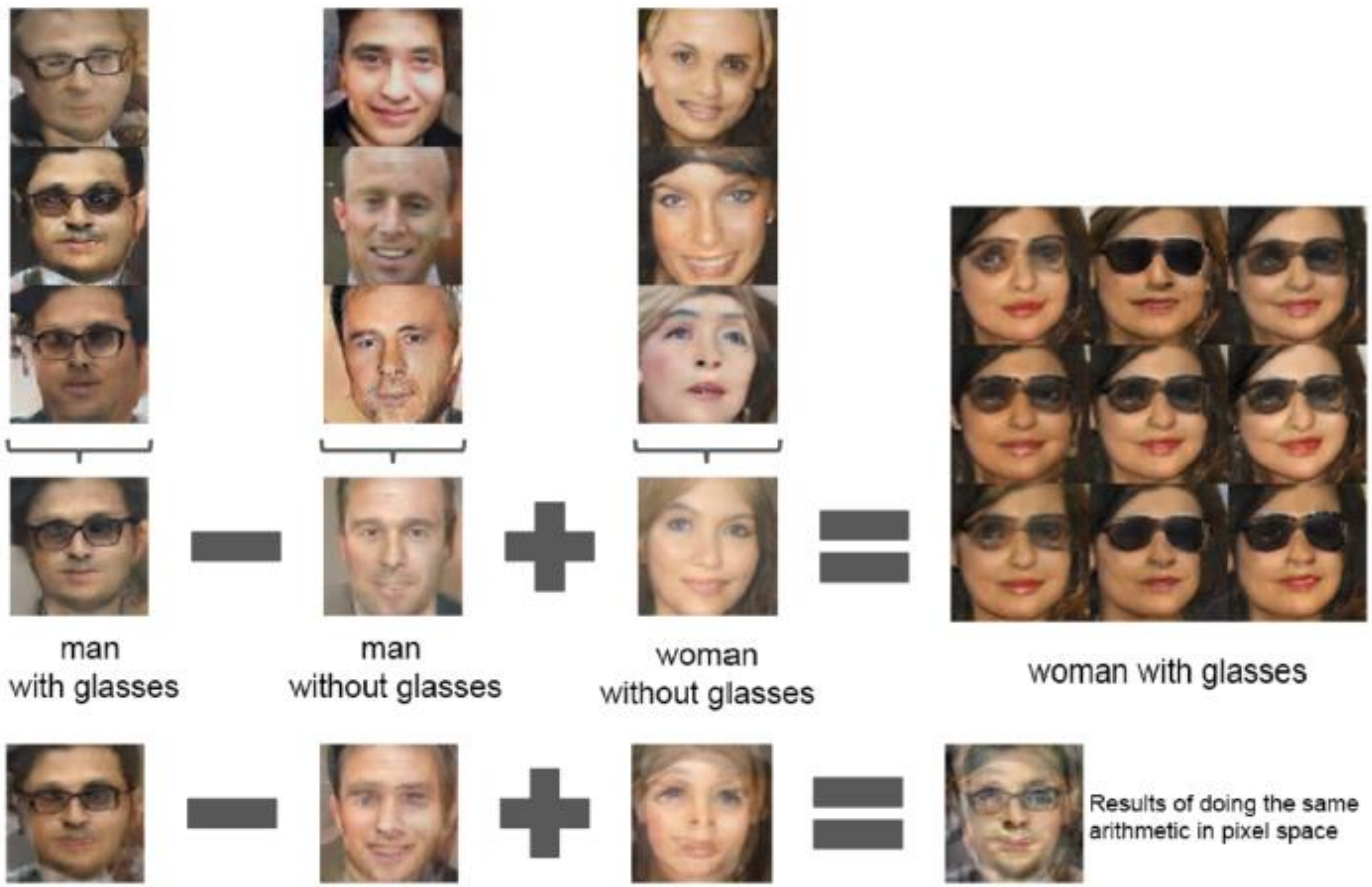
Walk Around Data Manifold

Interpolating between random points in laten space

Radford et al, ICLR 2016

Walk Around Data Manifold

- Vector Arithmetic



Outline

- Generative Adversarial Networks
  - Implicit generative models
  - Adversarial learning
  - Evaluation metrics

Acknowledgement: Feifei Li et al's cs231n notes

Evaluation metrics

- What makes a good generative model?
  - Each generated sample is indistinguishable from a real sample

  - Generated samples should have variety

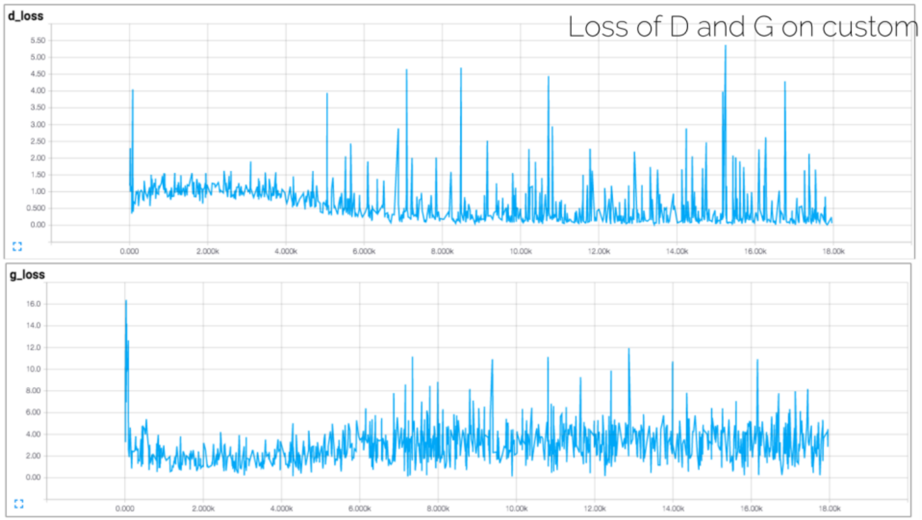
Images from Karras et al., 2017

Evaluation metrics

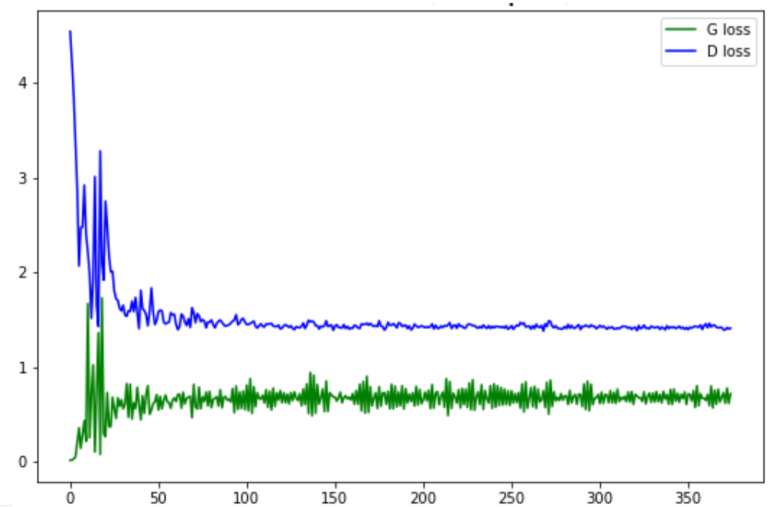
- How to evaluate the generated samples?
  - Cannot rely on the models' loss :-(
  - Human evaluation :-/
  - Use a pre-trained model :-)

Evaluation metrics

- “Good” Training Curves



Loss of D and G on custom dataset

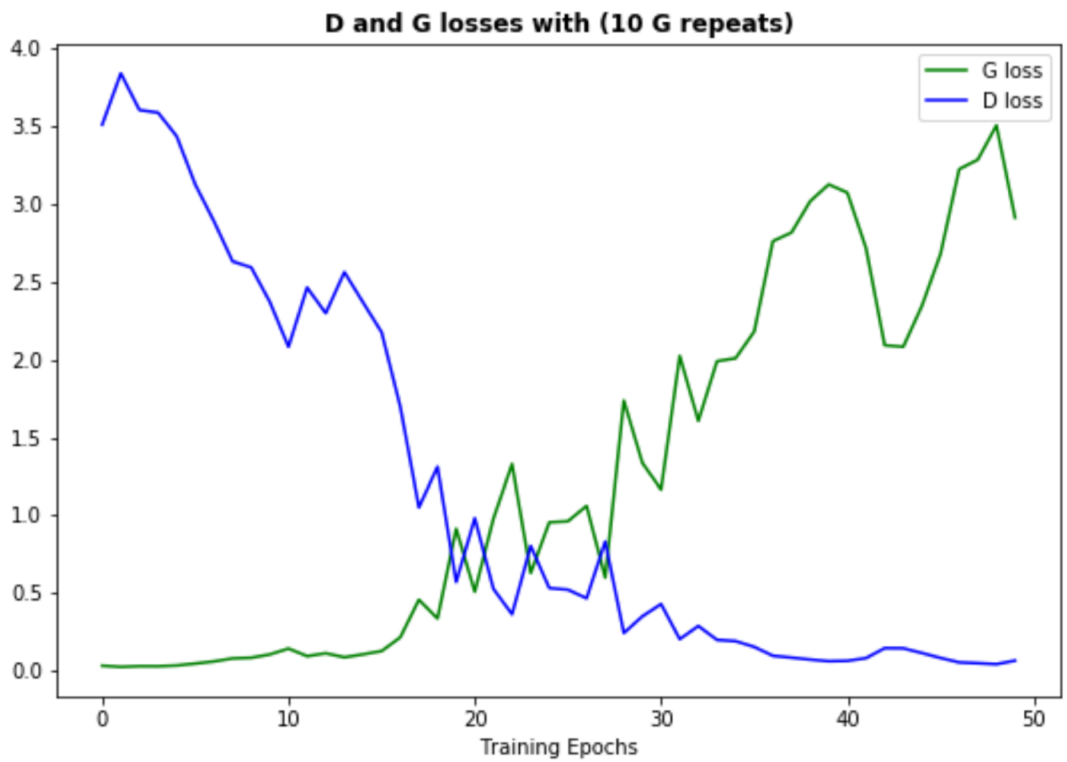


Generator's Error through Time

Discriminator's Error through Time

Evaluation metrics

- "Bad" Training Curves



Evaluation metrics

- Inception Score (IS) [Salimans et al., 2016]
  - Inception model p trained on ImageNet
  - Given generated image x, assigned the label y by model p

$$p(y|\boldsymbol{x}) \quad \Rightarrow \quad \text{low entropy (one class)}$$

  - The distribution over all generated images should be spread

$$\int p(y|\boldsymbol{x} = G(z))dz \quad \Rightarrow \quad \text{high entropy (many classes)}$$

  - Combining the above, we get the final metric:

$$\exp(\mathbb{E}_{\boldsymbol{x}} \mathrm{KL}(p(y|\boldsymbol{x})||p(y)))$$

Evaluation metrics

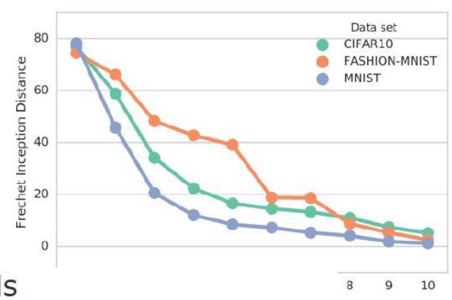
- Frechet Inception Distance (FID) [Heusel et al. 2017]
  - Calculates the distance between real and fake data (lower the better)
  - Uses the embeddings of the real and fake data from the last pooling layer of Inception v3.
  - Converts the embeddings into continuous distributions and uses the *mean* and *covariance* of each to calculate their distance.

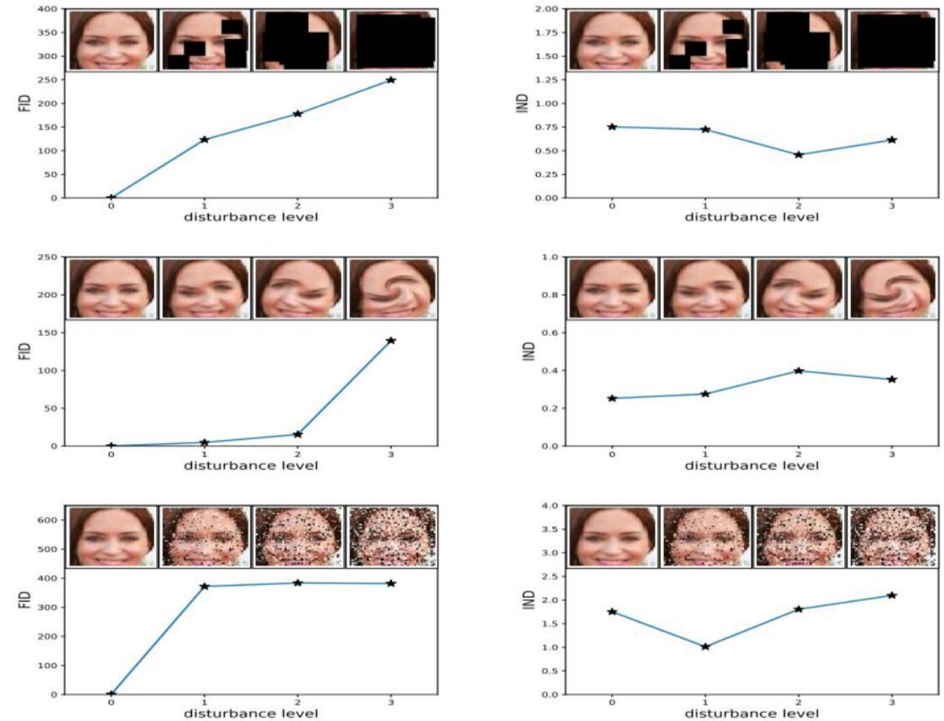
$$\mathrm{FID}(r,g) = ||\mu_r - \mu_g||_2^2 + \mathrm{Tr}(cov(r) + cov(g) - 2(cov(r)cov(g))^{\frac{1}{2}})$$

Evaluation metrics

- Comparisons

  - IS vs FID
  - ✓ FID considers the real dataset
  - ✓ FID requires less sampling (faster) (~10k instead of 50k in IS)
  - ✓ FID more robust to noise and human judgement
  - ✓ FID also sensitive to mode collapse





FID (lower is better) IS (higher is better)

Images from Lucic et al., 2017 and Heusel et al., 2017

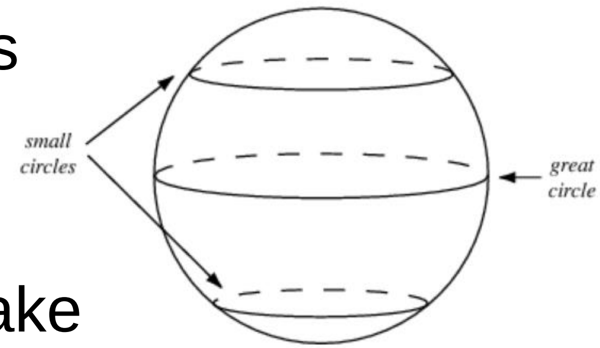
The GAN Zoo

- https://github.com/hindupuravinash/the-gan-zoo

  - 3D-ED-GAN - Shape Inpainting using 3D Generative Adversarial Network and Recurrent Convolutional Networks
  - 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling (github)
  - 3D-IWGAN - Improved Adversarial Systems for 3D Object Generation and Reconstruction (github)
  - 3D-PhysNet - 3D-PhysNet: Learning the Intuitive Physics of Non-Rigid Object Deformations
  - 3D-RecGAN - 3D Object Reconstruction from a Single Depth View with Adversarial Learning (github)
  - ABC-GAN - ABC-GAN: Adaptive Blur and Control for improved training stability of Generative Adversarial Networks (github)
  - ABC-GAN - GANs for LIFE: Generative Adversarial Networks for Likelihood Free Inference
  - AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
  - acGAN - Face Aging With Conditional Generative Adversarial Networks
  - ACGAN - Coverless Information Hiding Based on Generative adversarial networks
  - acGAN - On-line Adaptative Curriculum Learning for GANs
  - ACtuAL - ACtuAL: Actor-Critic Under Adversarial Learning
  - AdaGAN - AdaGAN: Boosting Generative Models
  - Adaptive GAN - Customizing an Adversarial Example Generator with Class-Conditional GANs
  - AdvEntuRe - AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples
  - AdvGAN - Generating adversarial examples with adversarial networks
  - AE-GAN - AE-GAN: adversarial eliminating with GAN
  - AE-OT - Latent Space Optimal Transport for Generative Models
  - AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
  - AF-DCGAN - AF-DCGAN: Amplitude Feature Deep Convolutional GAN for Fingerprint Construction in Indoor Localization System
  - AffGAN - Amortised MAP Inference for Image Super-resolution
  - AIM - Generating Informative and Diverse Conversational Responses via Adversarial Information Maximization

GAN Hacks

- https://github.com/soumith/ganhacks
- Normalize the inputs: [-1, 1], Tanh
- Use a spherical z; Use Batch Norm
- Different mini-batches for real and fake
- SGD for discriminator; ADAM for generator
- One-sided Label Smoothing



$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \underset{\uparrow}{\lambda} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

Some value smaller than 1; e.g.,0.9

- Avoid Sparse Gradients: no ReLU and MaxPooling

LeakyReLU → good in both G and D

Downsample → use average pool, conv+stride

Upsample → deconv+stride, PixelShuffle

Summary of GANs

Don’t work with an explicit density function
Take game-theoretic approach: learn to generate from training distribution through 2-player game

Pros:
- Beautiful, state-of-the-art samples!

Cons:
- Trickier / more unstable to train
- Can’t solve inference queries such as p(x), p(z|x)

Active areas of research:
- Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)
- Conditional GANs, GANs for all kinds of applications

- The GAN Zoo
  - https://github.com/hindupuravinash/the-gan-zoo