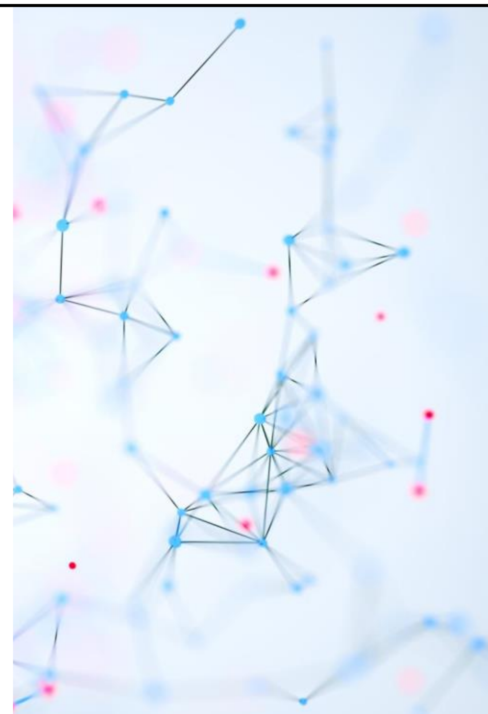
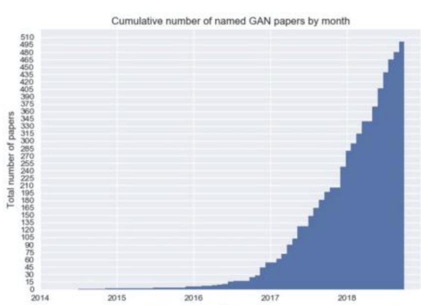
CS3750: ADVANCED MACHINE LEARNING

GENERATIVE ADVERSARIAL NETWORKS

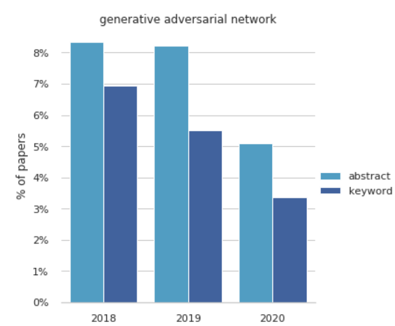
Adapted from Slides made by Khushboo Thaker

Presented by Tristan Maidment





Explosive growth—All the named GAN variants cumulatively since 2014. Credit: Bruno Gavranović



GROWTH (AND DECLINE) IN GAN PAPERS

Overview

1. Why Generative Modelling?
2. Existing Generative Models
3. Properties of GANs
4. GAN Framework
5. MiniMax game theory for GANs
6. Why GAN training is HARD
7. Tricks for GAN training
8. Common extensions to GANS
9. Conclusion

Generative Modelling

Input
Training Examples

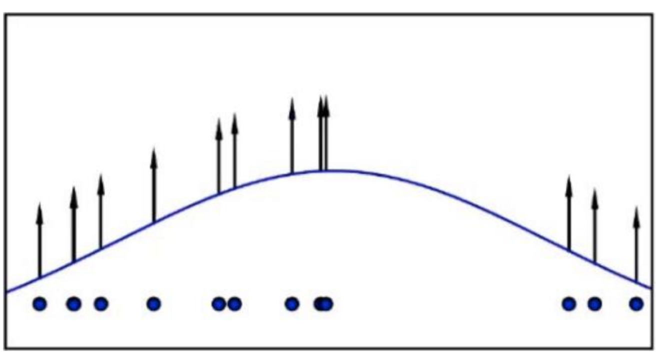
Output
Some representation of a probability distribution, which defines this example space.

Unsupervised
Data: X
Goal: Learn hidden underlying structure of data

Supervised
Data: X, y
Goal: Learn hidden mapping from X -> y

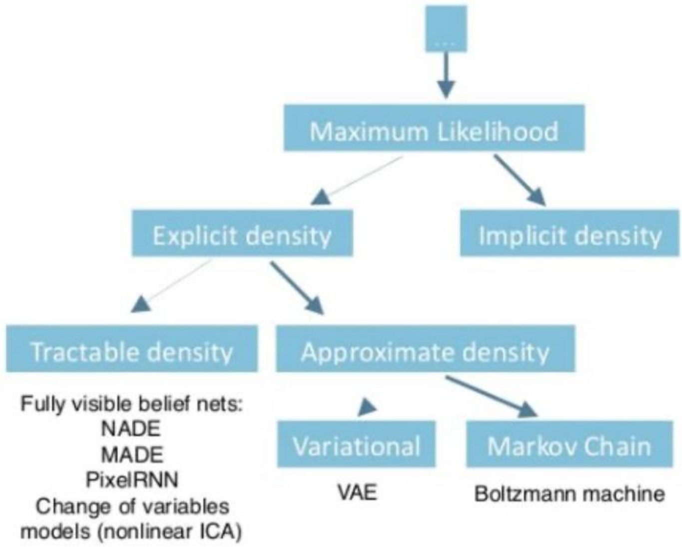
Why Generative Modelling?

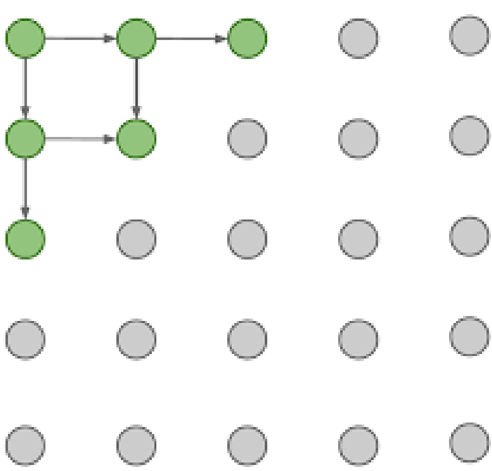
- Noisy Input
- Simulated Data
- Features Representative of Data
- Prediction of Future State
- Missing Data
- Semi-supervised Learning



MAXIMUM LIKELIHOOD BASED MODELS

$$p(x) \mid \theta^* = \text{ARG MAX}(\theta) \mid E_{x\sim Pdata} logP(\frac{x}{\theta})$$

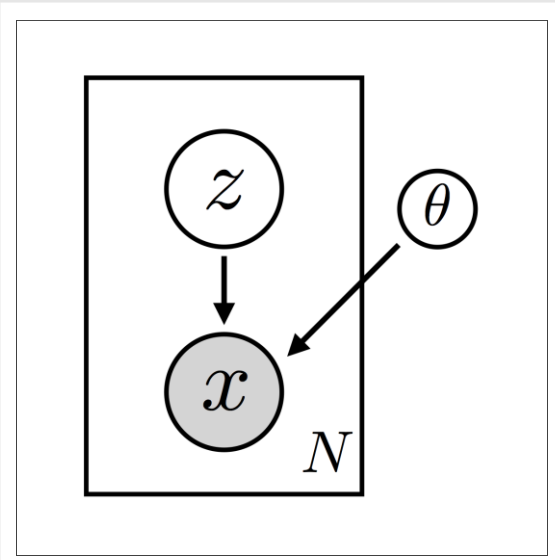




PixelRNN PixelCNN WaveNet

- Generate image pixels from the corner
- Stable and Fast training
- Slow generation (sequential)
- Cannot generate samples based on latent code
- Tractable

- $p(x) = \prod_{i=1}^{n} p(x_i | x_1, x_2, \ldots, x_{i-1})$
  - Maximum Likelihood based Training
  - Chain Rule



Variational Auto Encoder

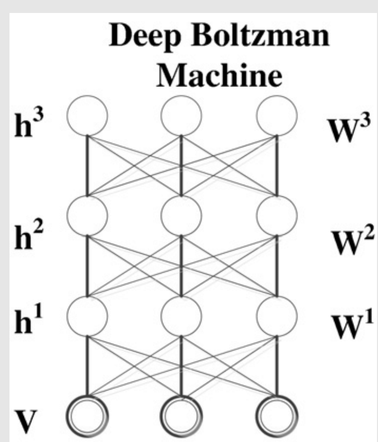
- Able to achieve high likelihood
- Not asymptotically consistent unless q is perfect
- Lower Quality (blurry) samples
- Non tractable

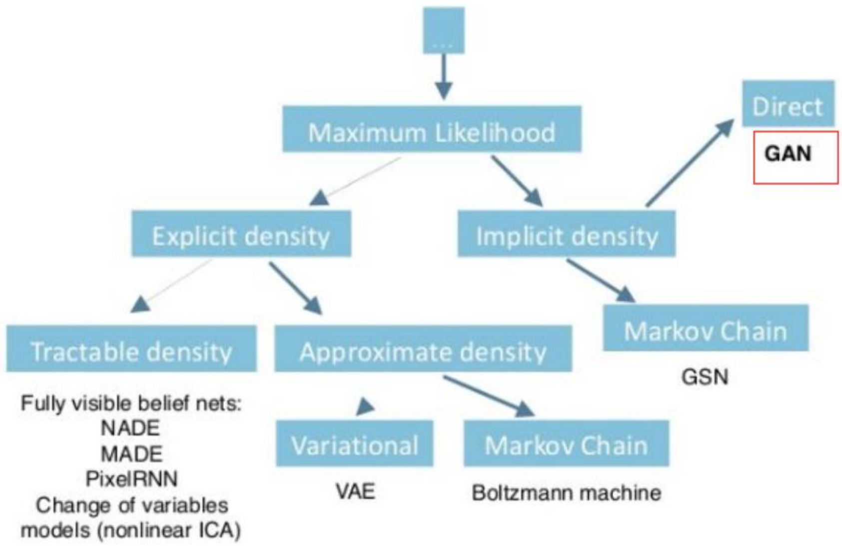
- $\log p(x) \geq \log p(x) - D_{KL}(q(z) \,||\, p(z|x))$
  $= E_{z \sim q} \log p(x, z) + H(q)$

Boltzmann Machine

- Energy Function Based Model
- Markov Chains don't work for long sequences
- Hard to scale on large dataset

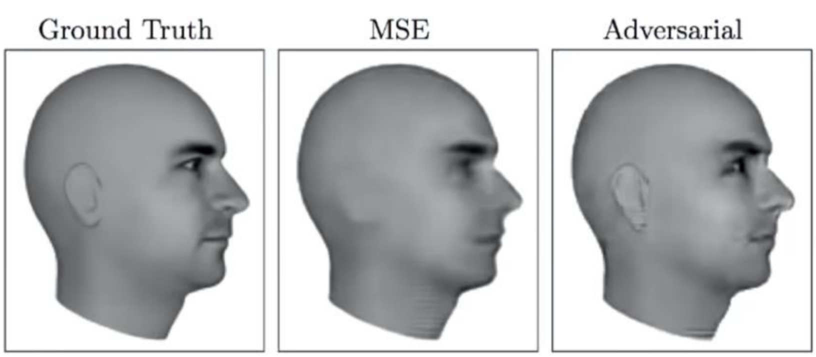
- $p(x, h) = \exp\big(-E(x, h)\big) \,|\, Z$
- $Z = \sum_{x,h} \exp(-E(x, h))$



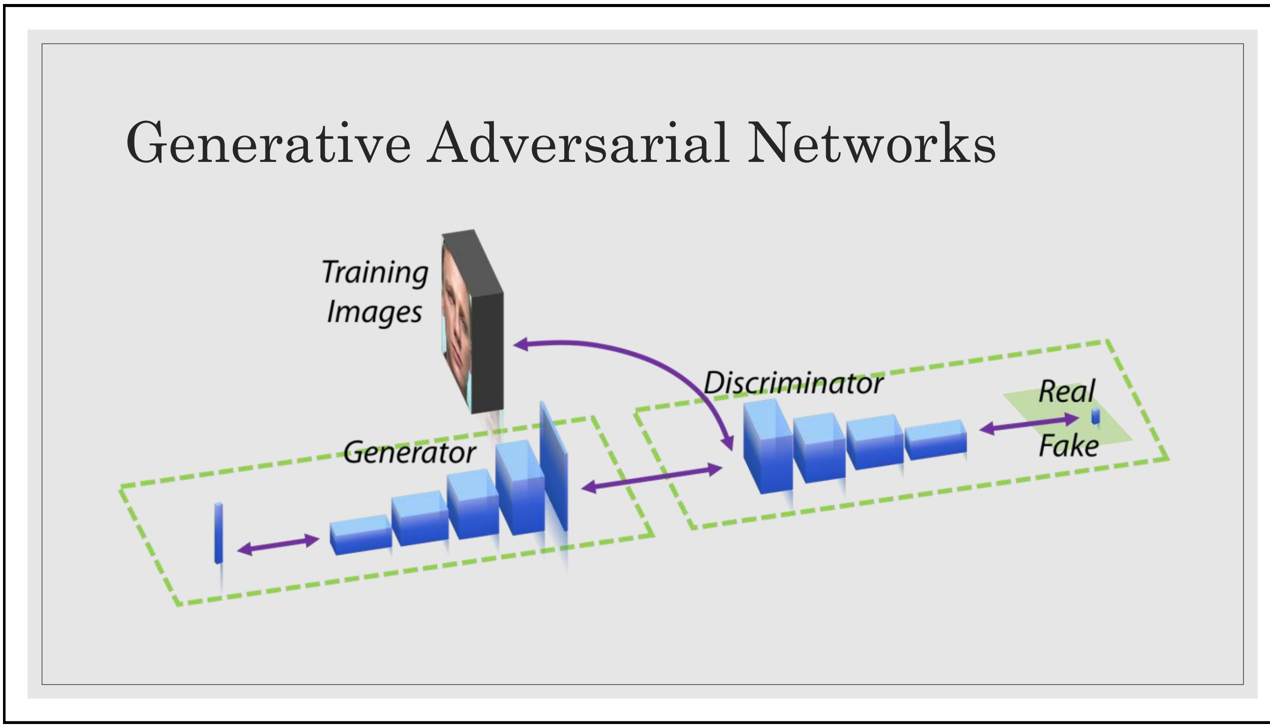


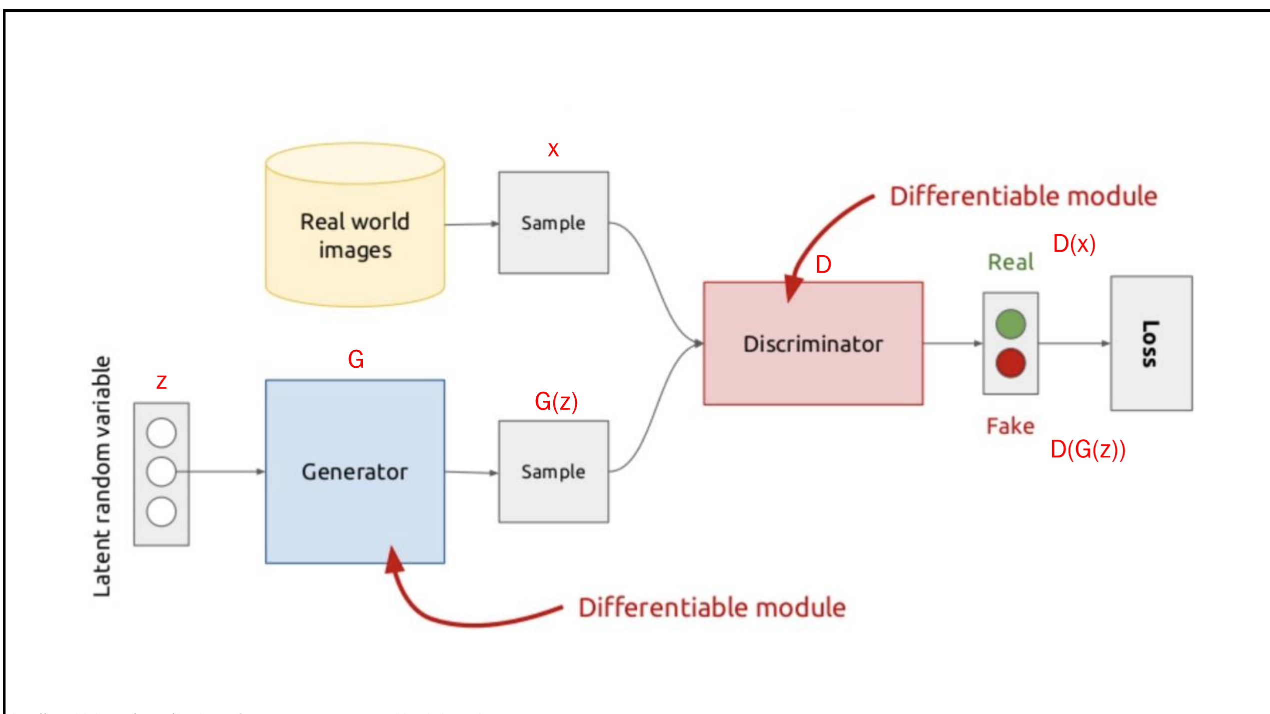
Where are some properties of GANs?

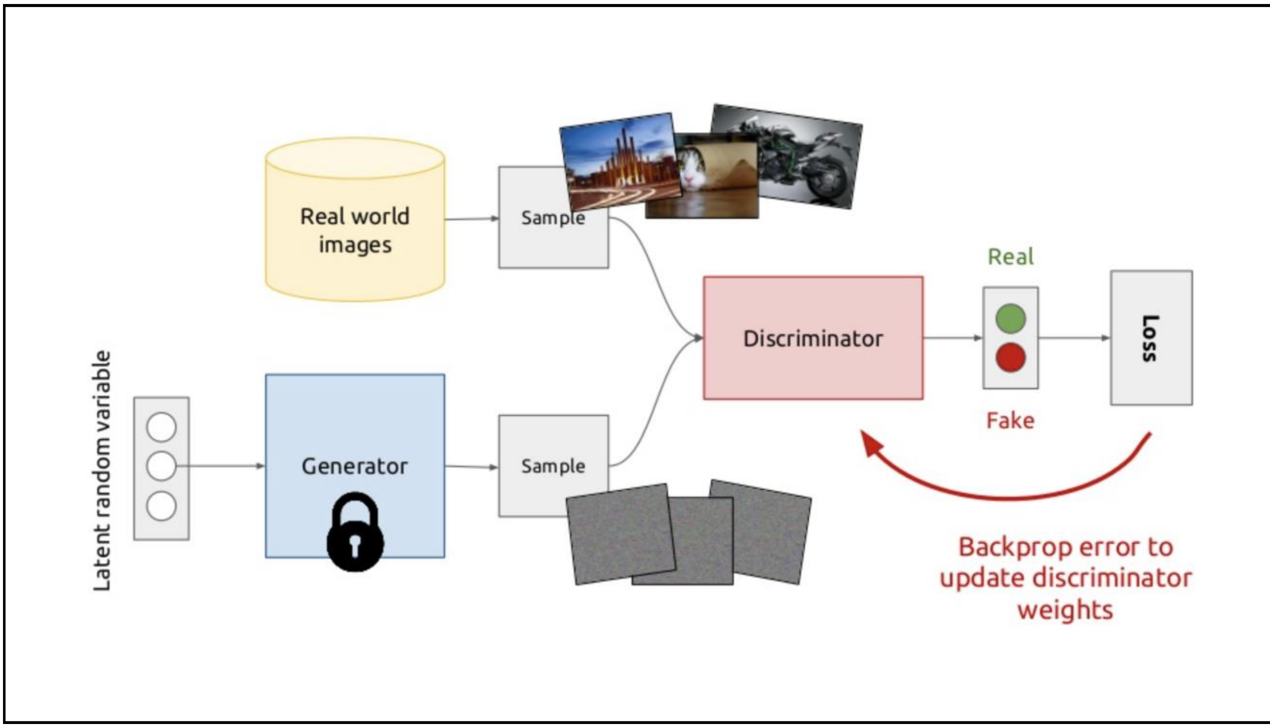
- Can use latent information
- Asymptotically consistent
- No Markov Chain assumption
- Samples produced are high quality

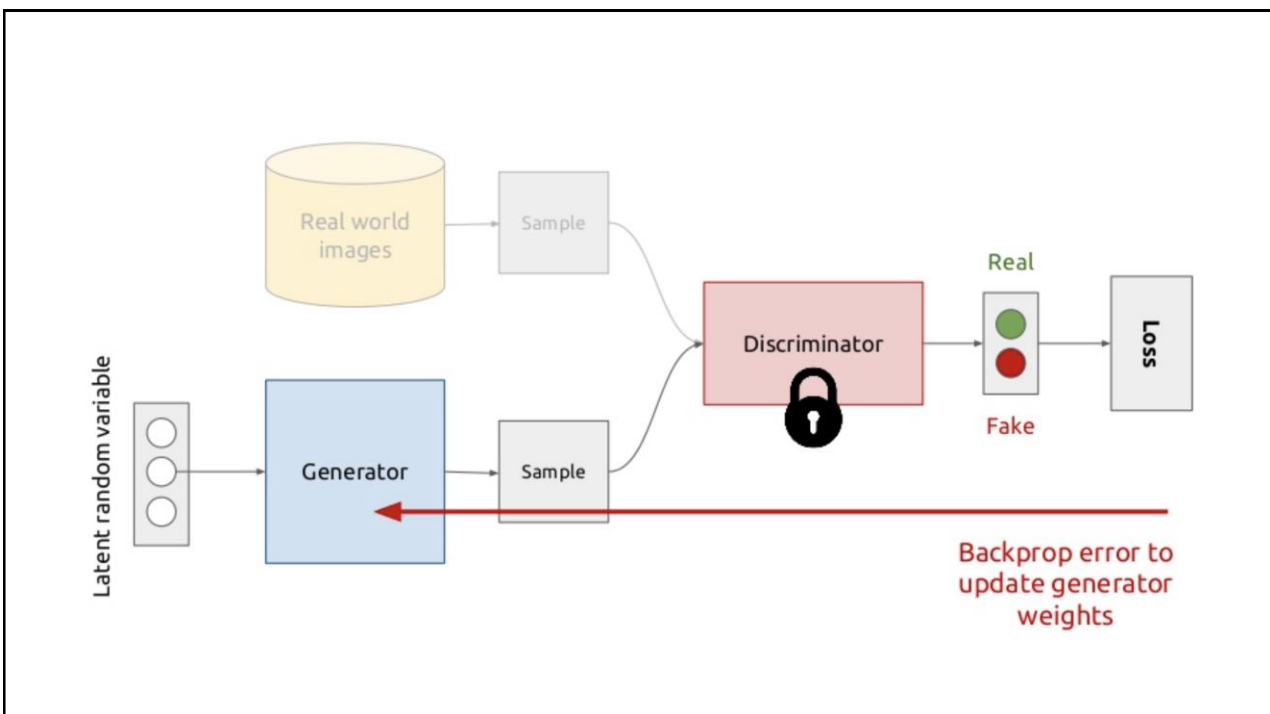


NEXT FRAME VIDEO GENERATION


Generative Adversarial Networks
Training Images
Discriminator
Real
Fake
Generator


x
Real world images
Sample
Differentiable module
D(x)
Real
D
Discriminator
Loss
Latent random variable
z
G
Generator
G(z)
Sample
Fake
D(G(z))
Differentiable module
https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016


Real world
images
Sample
Discriminator
Real
Fake
Loss
Latent random variable
Generator
Sample
Backprop error to
update discriminator
weights


Real world
images
Sample
Discriminator
Real
Fake
Loss
Latent random variable
Generator
Sample
Backprop error to
update generator
weights

Generative Adversarial Networks

"The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles."

– Goodfellow, et. Al. "Generative Adversarial Nets" (2014)

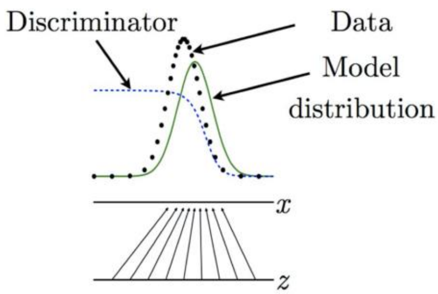
$$\min_G \max_D -J^D$$

$$J^D = -\frac{1}{2} E_{x\sim P_{data}} \underbrace{\log D(x)}_{\text{Discriminator output for real data x}} - \frac{1}{2} E_z \log \Big(1 - \underbrace{D\big(G(z)\big)}_{\text{Discriminator output for fake data G(z)}}\Big)$$

$$J^G = -J^D$$

Minimax Game Approach

- Generator minimizes the log-probability of the discriminator being correct
- Resembles Jensen-Shannon divergence
- Saddle point of Discriminator's loss



Nash Equilibrium / Saddle Point

$$\frac{\partial J^D}{\partial D(X)} = 0 \quad \rightarrow D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_{model}(x)}$$

$$p_{data(x)} = P_{model}(x) \quad \forall x$$

$$D^*(x) = ½ \quad \forall x$$

Minimax Game Approach

- Generator minimizes the log-probability of the discriminator being correct
- Resembles Jensen-Shannon divergence
- Saddle point of Discriminator's loss

$$J^D = -\frac{1}{2} E_{x \sim P_{data}} \log D(x) \boxed{- \frac{1}{2} E_z \log \left(1 - D(G(z))\right)}$$

Vanishing Gradient Problem

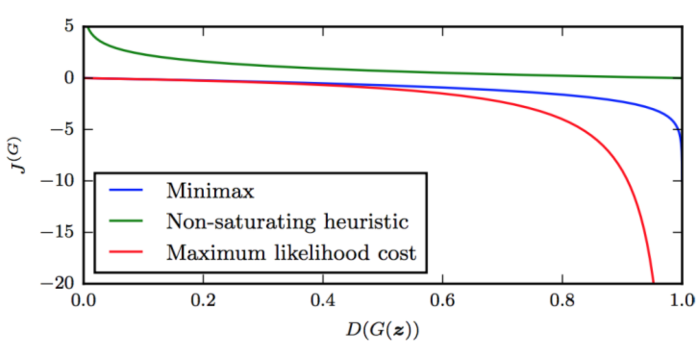
- Gradient disappears if D is confident, i.e. D(G(z)) ⟶ 0
- As can be seen that whenever the discriminator becomes very confident the loss value will be zero
- Nothing to improve for Generator

$$J^D = -\frac{1}{2} E_{x \sim P_{data}} \log D(x) - \frac{1}{2} E_z \log\left(1 - D\big(G(z)\big)\right)$$

$$J^G = -\frac{1}{2} E_z \log D\big(G(z)\big)$$

Heuristic Non-Saturating Games

- Generator maximizes the log probability of the discriminator's mistake
- Does not change when discriminator is successful



$$J^G = -J^D$$

$$J^G = -\frac{1}{2} E_z \exp\left(\sigma^{-1} D\big(G(z)\big)\right)$$

$$J^G = -\frac{1}{2} E_z \log D\big(G(z)\big)$$

COMPARISON OF GENERATOR LOSSES

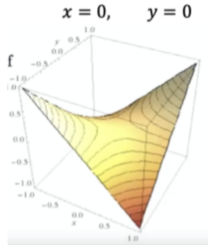
Non-Convergence

D & G nullifies each others learning in every iteration

Train for a long time – without generating good quality samples

$V(x,y) = xy$

$x = 0, \quad y = 0$



$V(x(t), y(t)) = x(t)y(t)$

$$\frac{\partial x}{\partial t} = -y(t)$$

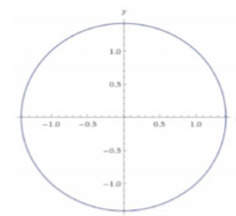
$$\frac{\partial y}{\partial t} = x(t)$$

$$\frac{\partial^2 y}{\partial t^2} = \frac{\partial x}{\partial t} . = -y(t)$$

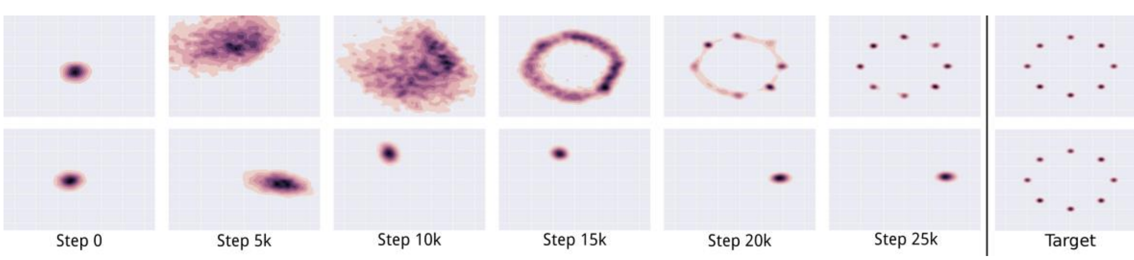
$$x(t) = x(0)cost(t) - y(0)\sin(t)$$

$$y(t) = x(0)cost(t) - y(0)\sin(t)$$

- Differential Equation's solution has sinusoidal terms
- Even with a small learning rate, it will not converge
- Discrete time gradient descent can spiral outward for large step size

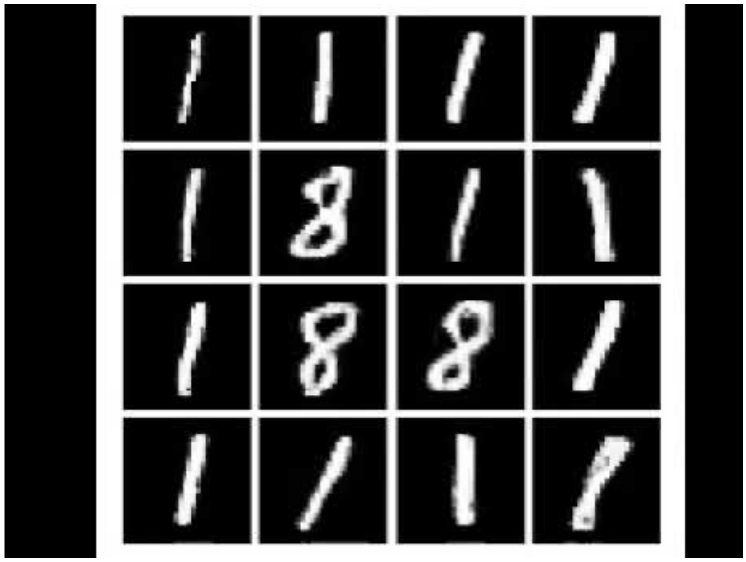


Khushboo Thaker 27



MODE COLLAPSE

$$min_G max_D V(G,D) \neq max_D min_G V(G,D)$$



Why are GANs hard to train?

GENERATOR KEEPS GENERATING SIMILAR IMAGES – SO NOTHING TO LEARN

MAINTAIN TRADE-OFF OF GENERATING **MORE ACCURATE** VS **HIGH COVERAGE** SAMPLES

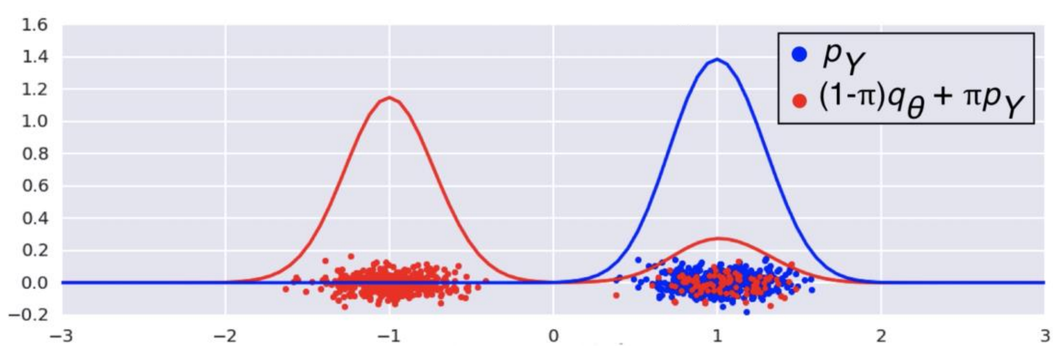
THE TWO LEARNING TASKS NEED TO HAVE BALANCE TO ACHIEVE STABILITY

IF DISCRIMINATOR IS NOT SUFFICIENTLY TRAINED – LEADS TO POOR GENERATOR PERFORMANCE

IF DISCRIMINATOR IS OVER-TRAINED – VANISHING GRADIENT PROBLEM

Tricks to Train GANs

- One-Sided Label Smoothing
- Historically generated batches
- Feature Matching
- Batch Normalization
- Regularizing discriminator gradient in region around real data (DRAGAN)



$$J^D = -\frac{1}{2} E_{x \sim P_{data}} 0.9 \log D(x) - \frac{1}{2} E_z \log\left(1 - D(G(z))\right)$$

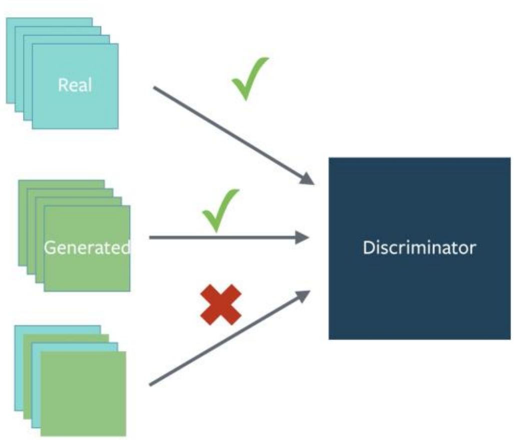
One-Sided Label Smoothing

- Generator is VERY sensitive to output from Discriminator
- Regulates Discriminator gradients
- Does-not reduce accuracy
- Increases confidence
- Only smooth *positive* samples

Feature Matching

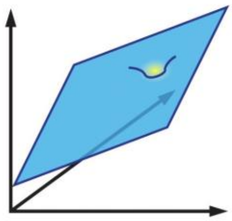
- Generated images must match statistics of real images
- Discriminator defines the statistics
- Generator is trained such that the expected value of statistics matches the expected value of real statistics
- Generator tries to minimize the L2 distance in expected values in some arbitrary space
- Discriminator defines that arbitrary space

$$\| \; E_{x \sim pdata} f(x) - E_{z \sim Pmodel} \; f\big(G(z)\big) \|_2^2$$



Batch Normalization

- Construct different mini-batches for real and fake
- Each mini-batch needs to contain only all real images or all generated images.
- Makes samples with-in a batch less dependent



$$\lambda \,.\, E_{x\sim pdata,\delta\sim N(0,cI)}\,[\,\|\Delta_x D(x+\delta)\| - k]^2$$

DRAGAN

Failed GANs typically have extreme gradients/sharp peaks around real data

Regularize GANs to reduce the gradient of the discriminator in region around real data

GAN Variations

- Conditional GAN
- LapGAN
- DCGAN
- CatGAN
- InfoGan
- AAE
- DRAGAN
- IRGAN
- ProGAN
- and more!

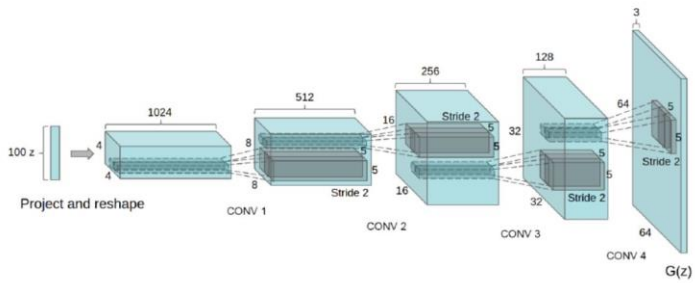
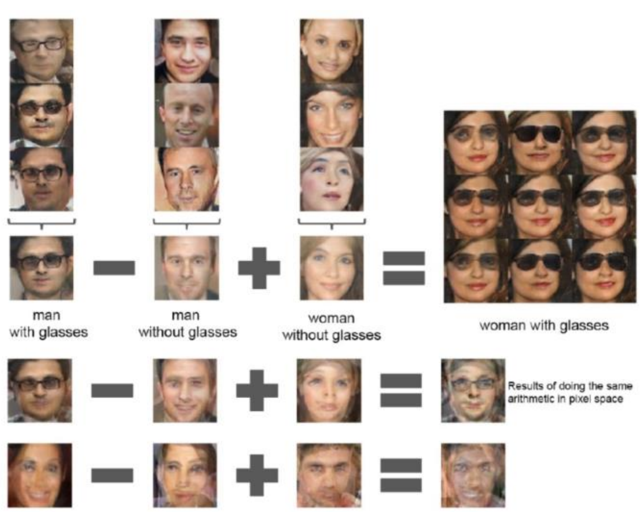


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

DCGAN

- Multiple Convolutional Layers
- Batch Normalization
- Strides with Convolution
- Leaky ReLUs



DCGAN

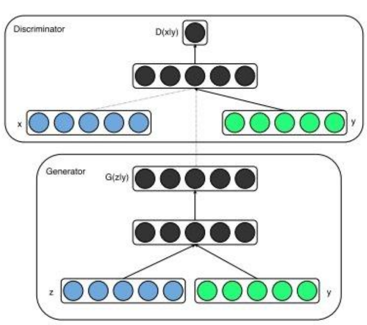
- Multiple Convolutional Layers
- Batch Normalization
- Strides with Convolution
- Leaky ReLUs

(a) Rotation (b) Width

Figure 4: **Manipulating latent codes on 3D Chairs:** In (a), we show that the continuous code captures the pose of the chair while preserving its shape, although the learned pose mapping varies across different types; in (b), we show that the continuous code can alternatively learn to capture the widths of different chair types, and smoothly interpolate between them. For each factor, we present the representation that most resembles prior supervised results [7] out of 5 random runs to provide direct comparison.

DCGAN

- Multiple Convolutional Layers
- Batch Normalization
- Strides with Convolution
- Leaky ReLUs



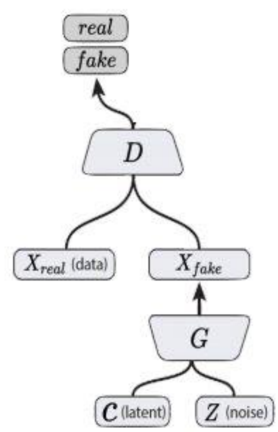
Conditional GANs *P(X|Y)*

- Generator Learns P(X|Z,Y)
- Discriminator Learns P(L|X,Y)

$$J^D = -\frac{1}{2} E_{x\sim P_{data}} \log D(x|y) - \frac{1}{2} E_z \log\left(1 - D(G(z|y))\right)$$

InfoGAN

- Rewards Disentanglement
  - (individual dimensions capturing key attributes of images)
- Z – partitioned into two parts
  - z – capture slight variation in the images
  - y – captures the main attributes of the images
- Mutual Information
  - maximizing mutual information Between the code and generator output



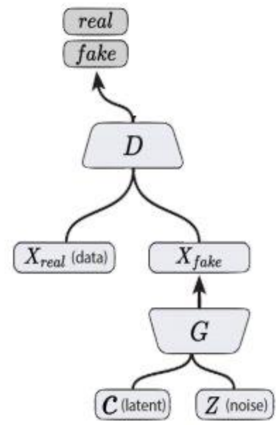
InfoGAN

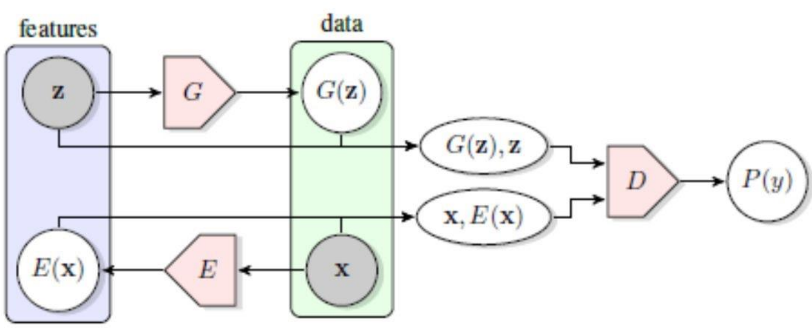
$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$

$$I(c; G(z,c)) = H(c) - H(c|G(z,c))$$

$$= E_{x\sim G(z,c)}[\, D_{KL}\,(P\,||Q) + E_{c'\sim p(c|x)}\ [\log Q(c'|x)]\,] + H(c)$$

$$\geq E_{x\sim G(z,c), c\sim p(c)}, [\log Q(c|x)]\ + H(c)$$



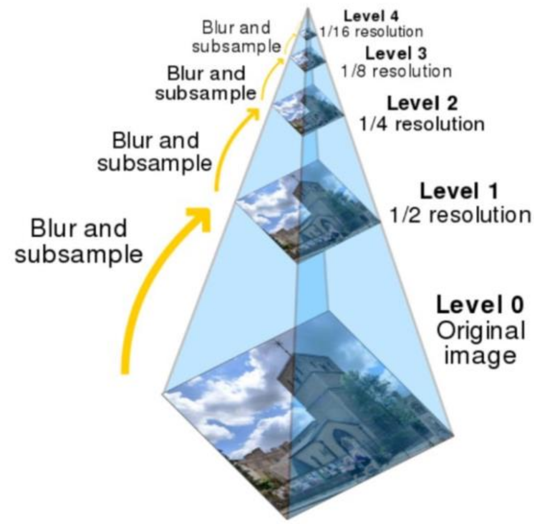


BiGAN

- Encoder
- Decoder
- Discriminator

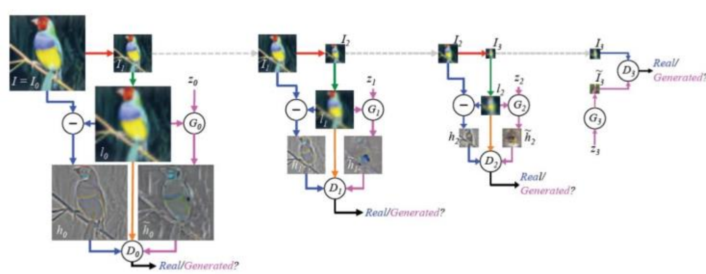
LapGAN

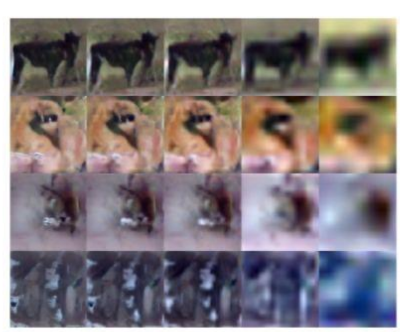
- Scale GANs for large images
- Laplacian pyramid function is used to generate different scales of image

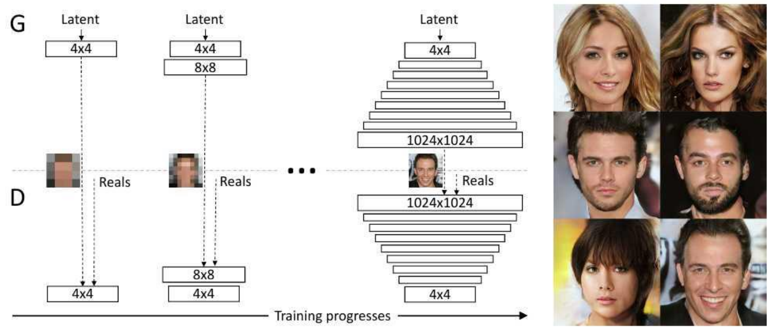


LapGAN

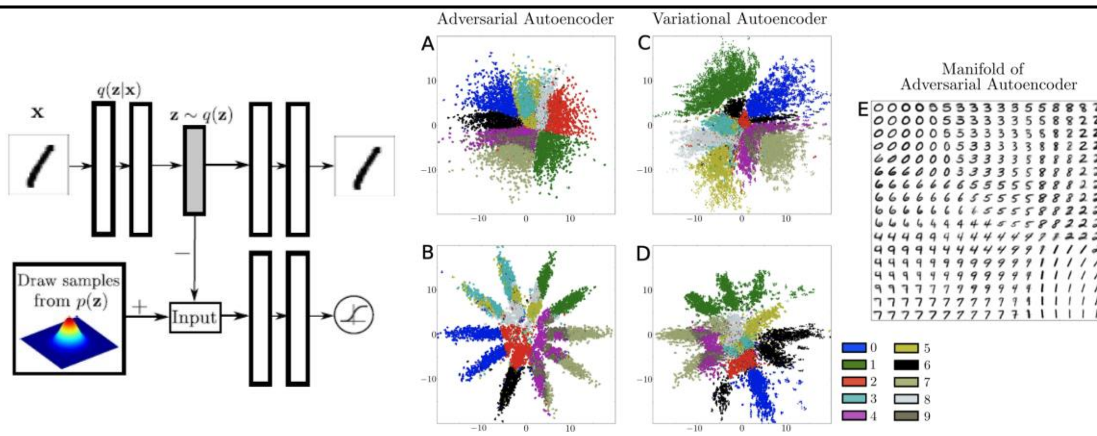
- Scale GANs for large images
- Laplacian pyramid function is used to generate different scales of image







PROGAN



ADVERSARIAL AUTOENCODER (GAN + VAE)

Conclusion

- GAN is still an active area of research
- GAN framework is flexible to support variety of learning problems
- GAN does not guarantee to converge
- GAN can capture perceptual similarity and generates better images than VAE
- Needs a lot of work in theoretic foundation of Network
- Evaluation of GAN is still an open research (Theis et. al)

Important Papers to dig into GAN

- **NIPS 2016 Tutorial:** - Ian Goodfellow
- Arjovsky, Martin, and Léon Bottou. "Towards principled methods for training generative adversarial networks." arXiv preprint arXiv:1701.04862 (2017).
- Roth, Kevin, et al. "Stabilizing training of generative adversarial networks through regularization." Advances in Neural Information Processing Systems. 2017.
- Li, Jerry, et al. "Towards understanding the dynamics of generative adversarial networks." arXiv preprint arXiv:1706.09884 (2017).
- Kodali, Naveen, et al. "On convergence and stability of GANs." arXiv preprint arXiv:1705.07215 (2017).
- Fedus, William, et al. "Many Paths to Equilibrium: GANs Do Not Need to Decrease aDivergence At Every Step." arXiv preprint arXiv:1710.08446 (2017).
- https://github.com/soumith/ganhacks#authors
- http://www.inference.vc/instance-noise-a-trick-for-stabilising-gan-training/
- https://www.araya.org/archives/1183

Software

- https://github.com/eriklindernoren/Keras-GAN
- https://github.com/eriklindernoren/PyTorch-GAN
- https://github.com/znxlwm/tensorflow-MNIST-cGAN-cDCGAN

TensorFlow

References

- Deep Learning Book
- GAN paper: https://arxiv.org/abs/1701.00160
- GAN slides: http://slazebni.cs.illinois.edu/spring17/lec11_gan.pd
- GAN Tutorial: https://www.youtube.com/watch?v=HGYYEUSm-0Q


THANK YOU!