UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack:Divija L
Teaching Assistant

PES UNIVERSITY

GANs - Generative Adversarial Networks

Why Generative Models?

➢ **Discriminative models so far:**

- Given an image X, predict a label Y
- Estimates **P(Y|X)**

➢ **Discriminative models have several key limitations:**

- Can't model **P(X),** i.e. the probability of seeing a certain image
- Thus, can't sample from **P(X),** i.e. can't generate new images

SOLUTION:

Generative models (in general) can cope with all of the following:

➔ **Can model P(X)**

➔ **Can generate new images**

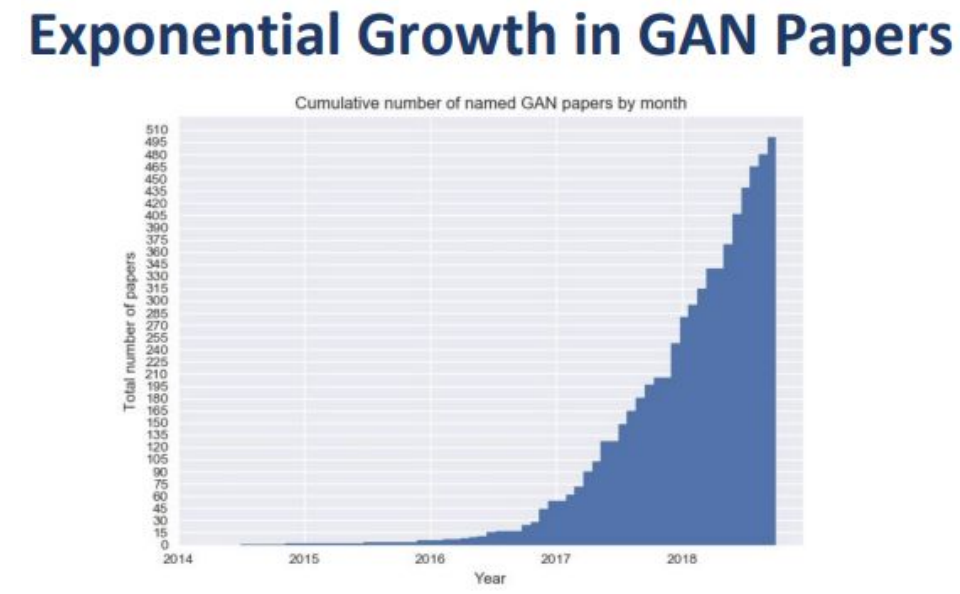
Why care about GANs?

- GANs are achieving state-of-the-art results in a large variety of image generation tasks.
- There's been a veritable explosion in GAN publications over the last few years – many people are very excited!
- GANs are stimulating new

  theoretical interest in min-max

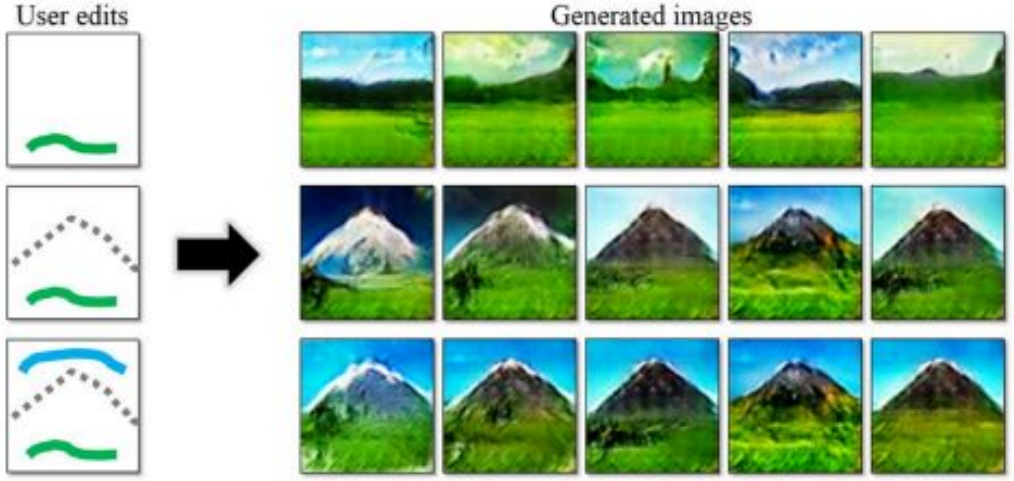
  optimization problems and

  "smooth games"



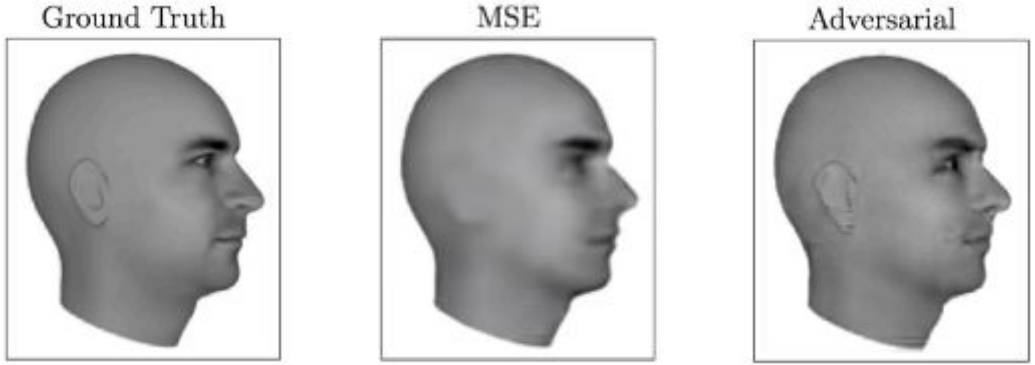
Adversarial Training

- We can generate adversarial samples to fool a discriminative model.
- We can use those adversarial samples to make models robust.
- We then require more effort to generate adversarial samples.
- Repeat this and we get better discriminative model.



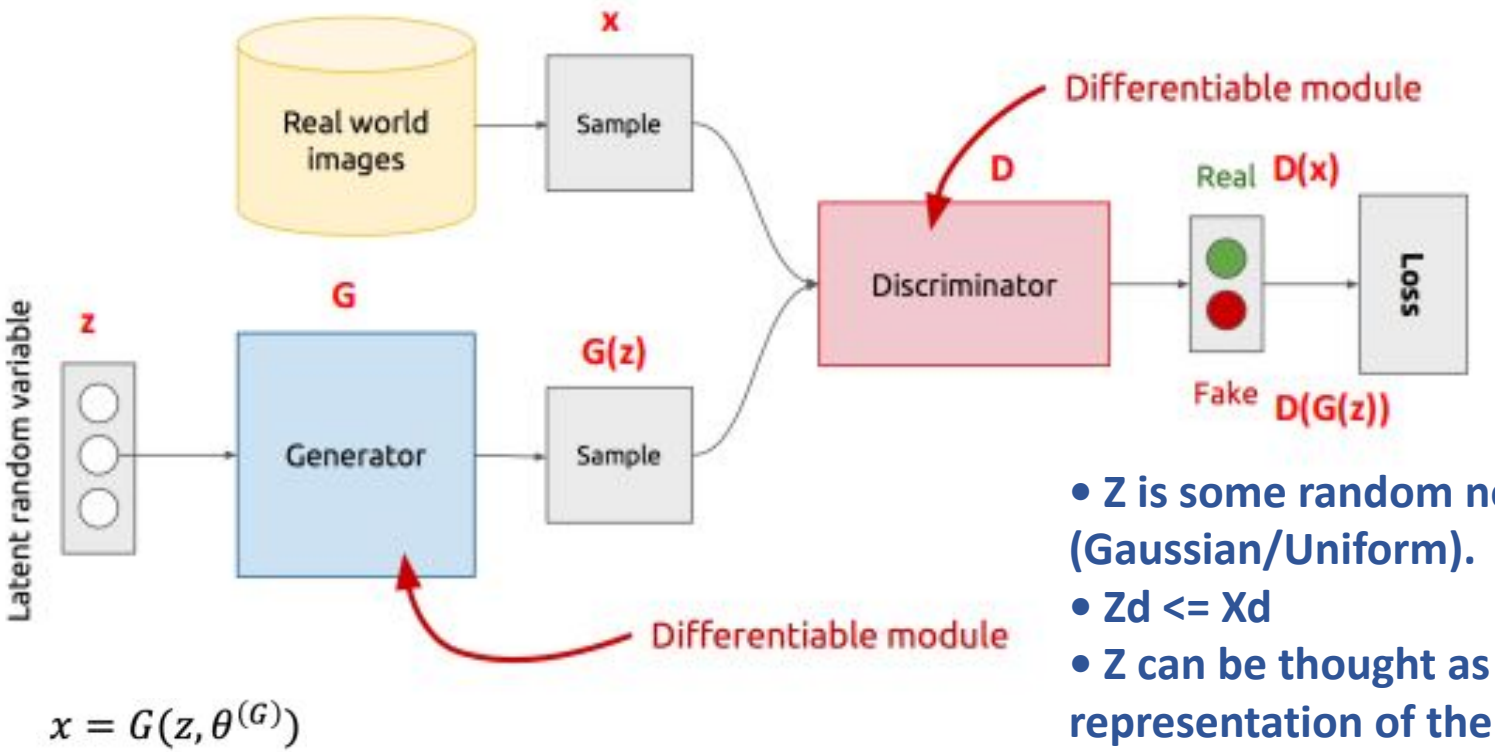
GANs extend that idea to generative models:

1. **Generator:** generate fake samples, tries to fool the Discriminator
2. **Discriminator:** tries to distinguish between real and fake samples
3. Train them against each other
4. Repeat this and we get better Generator and Discriminator



- Sharp image
- Better estimation of Ear position
- Much crisp eyes

GAN's Architecture

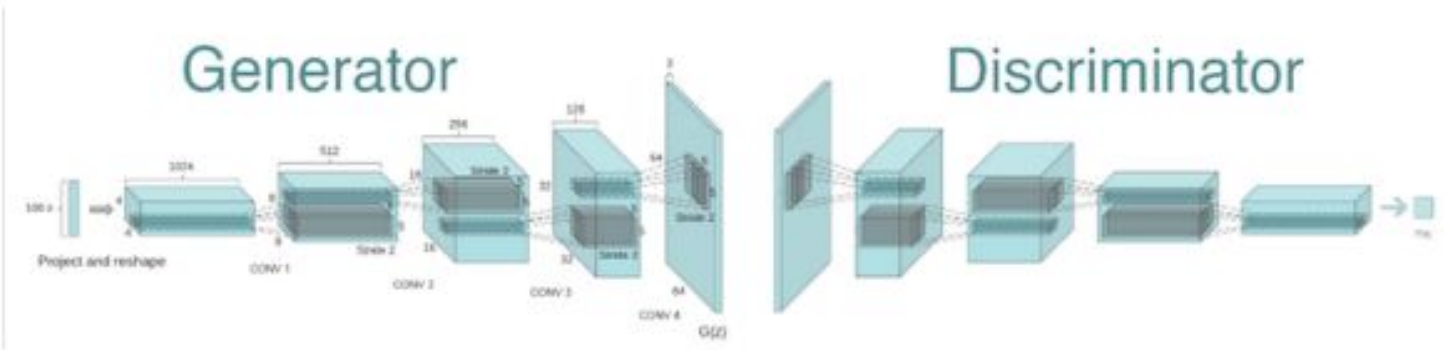


$$x = G(z, \theta^{(G)})$$

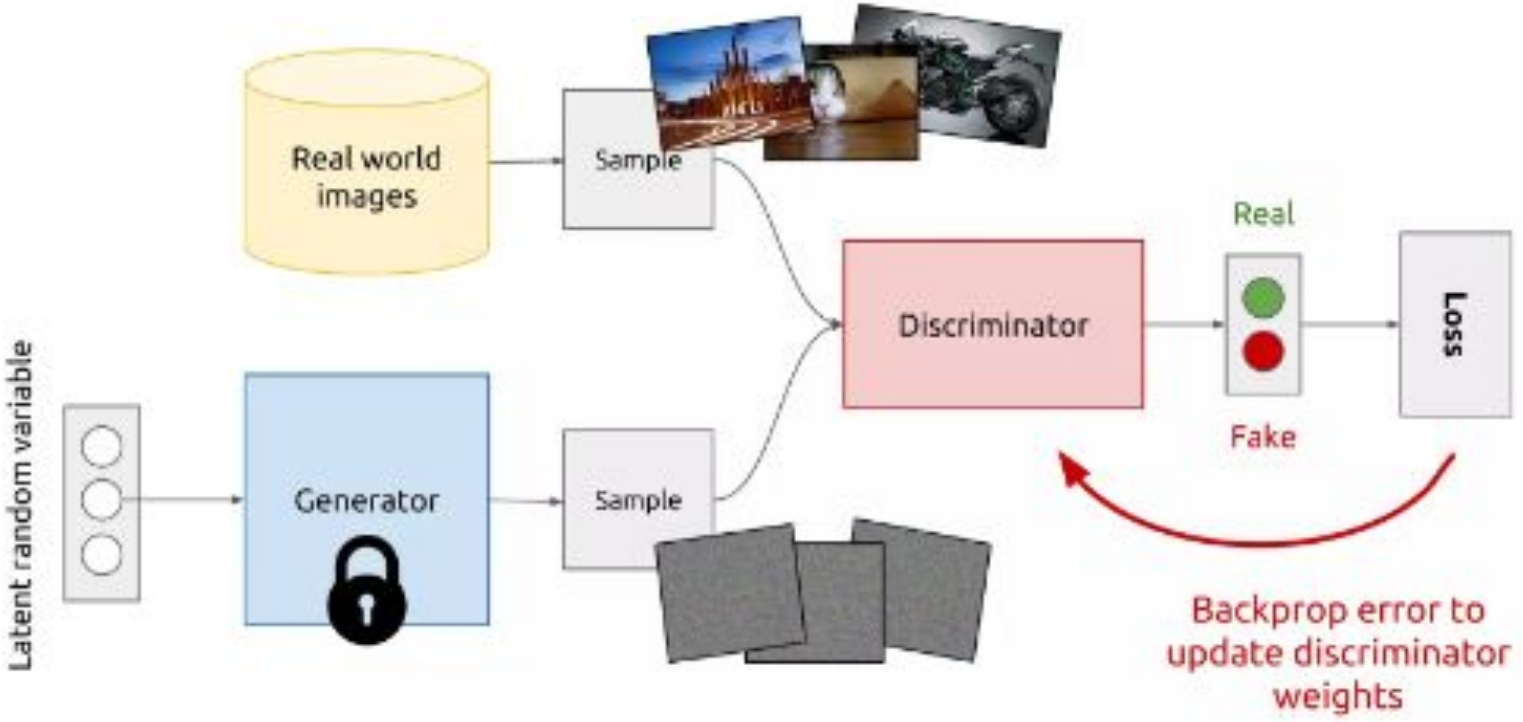
- Z is some random noise (Gaussian/Uniform).
- Zd <= Xd
- Z can be thought as the latent representation of the image.

GANs: Goodfellow et al. (2014)

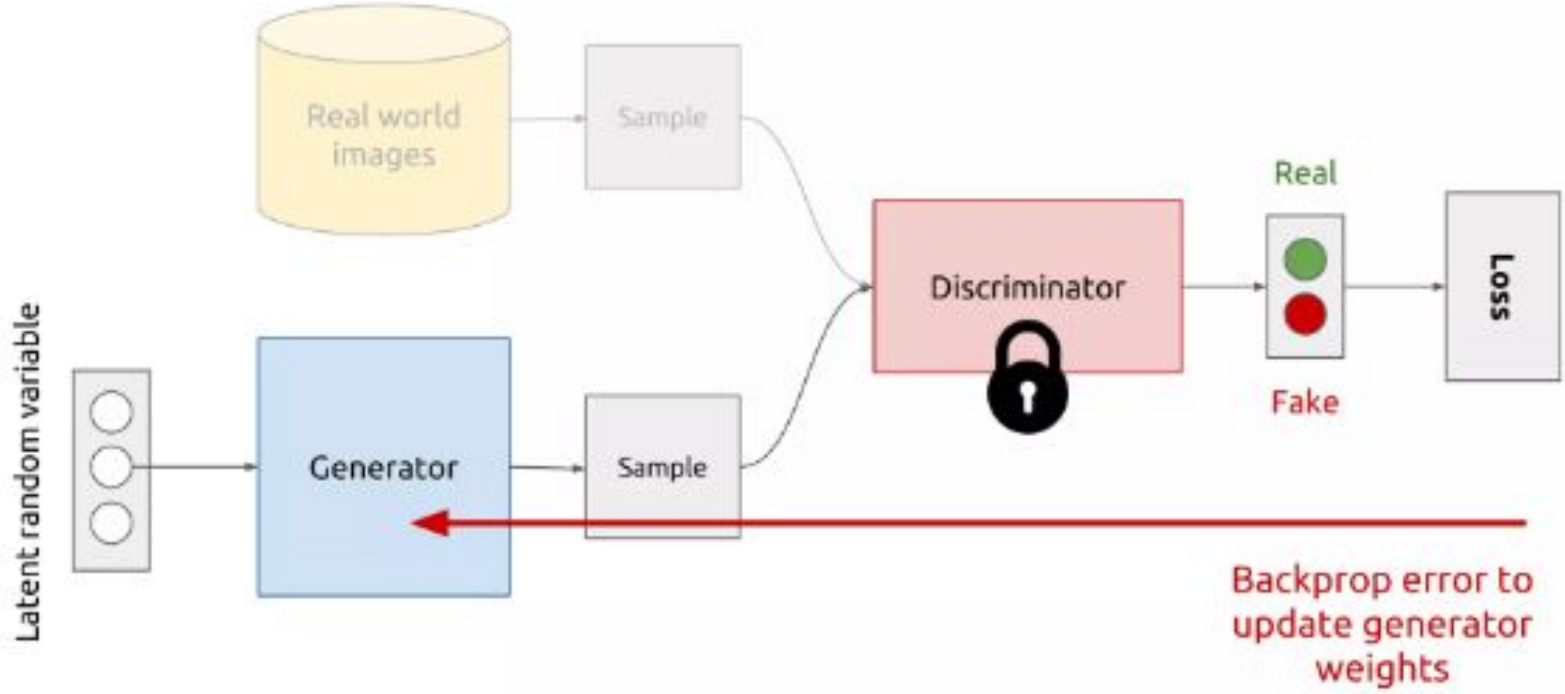
- Let $\mathbf{z} \in \mathbb{R}^m$ and $p_{\mathbf{z}}(\mathbf{z})$ be a simple base distribution.
- The generator $G_{\theta_g}(\mathbf{z}) : \mathbb{R}^m \rightarrow \tilde{\mathcal{D}}$ is a deep neural network.
  - $\tilde{\mathcal{D}}$ is the manifold of generated examples.
- The discriminator $D_{\theta_d}(\mathbf{x}) : \mathcal{D} \cup \tilde{\mathcal{D}} \rightarrow (0, 1)$ is also a deep neural network.



1. Fix generator weights , draw samples from both real world and generated images.
2. Train Discrimator to distinguish between real world and generated images



1. Fix discriminator weights
2. Sample from Generator
3. Backdrop error through discriminator to update generator weights



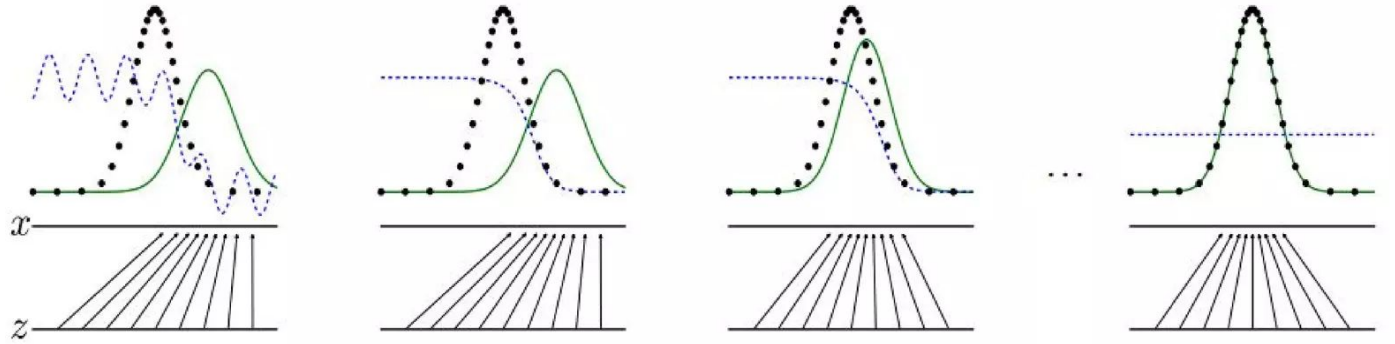
PES UNIVERSITY

Training GANs

Iterate these two steps until convergence (which may not happen)

- Updating the discriminator should make it better at discriminating between real images and generated ones (**discriminator improves**)
- Updating the generator makes it better at fooling the current discriminator (**generator improves**)

Eventually (we hope) that the generator gets so good that it is impossible for the discriminator to tell the difference between real and generated images. Discriminator accuracy = 0.5



GANs: Training Procedure

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

Discriminator updates

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

Generator updates

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Min-max Game Approach

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

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

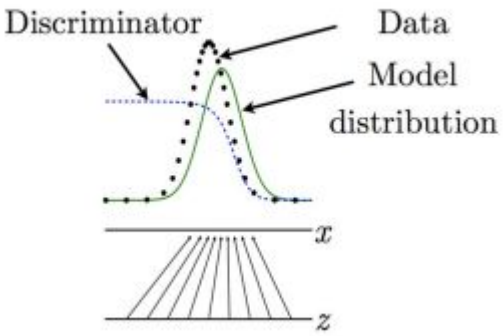
$$J^G = -J^D$$

Discriminator output for real data x

Discriminator output for fake data G(z)

- Generator minimizes the log-probability of the discriminator being correct
- Resembles Jensen-Shannon divergence
- Saddle Point of discriminators loss

Min-max Game Approach



Nash Equilibrium / Saddle Point

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

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

$D^*(x) = ½ \quad \forall x$

- **Generator minimizes the log-probability of the discriminator being correct**
- **Resembles Jensen-Shannon divergence**
- **Saddle Point of discriminators loss**

Problems with GANs

1. **Vanishing gradients:**

   the discriminator becomes ”too good” and the generator gradient vanishes.

- The **minimax** objective saturates when D_{θ_d} is close to perfect:

$$V(\theta_d, \theta_g) = \mathbb{E}_{p_{\text{data}}} [\log D_{\theta_d}(\mathbf{x})] + \mathbb{E}_{p_z(\mathbf{z})} \left[\log \left(1 - D_{\theta_d}(G_{\theta_g}(\mathbf{z}))\right)\right].$$

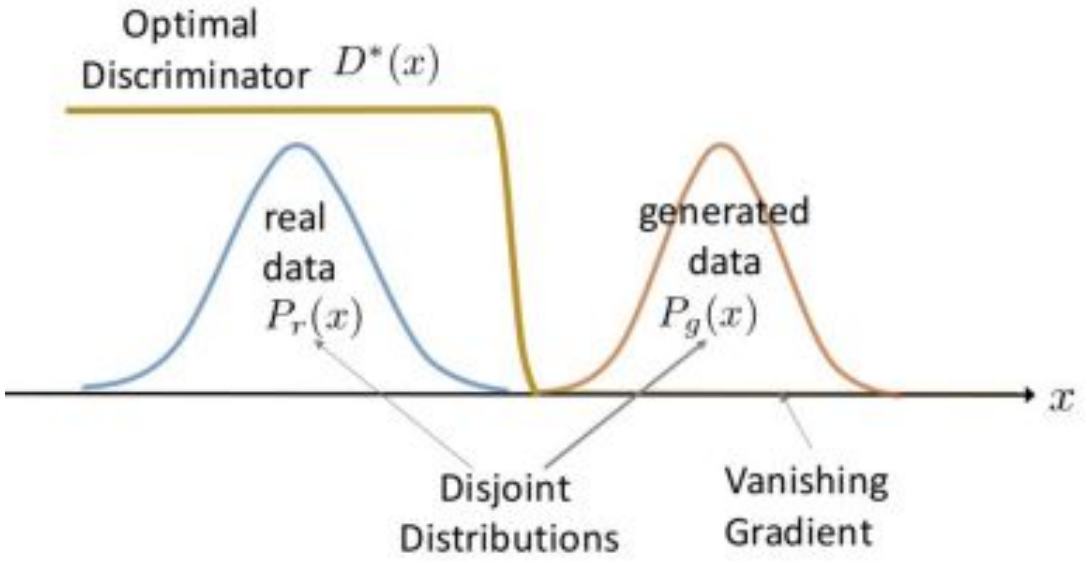
Vanishing Gradient Problem with Generator

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

Gradient goes to 0 if D is confident , ie 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



Heuristic Non Saturating Game

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

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

-> Generator maximizes the log probability of the discriminator's mistake

->Does not change when discriminator is successful

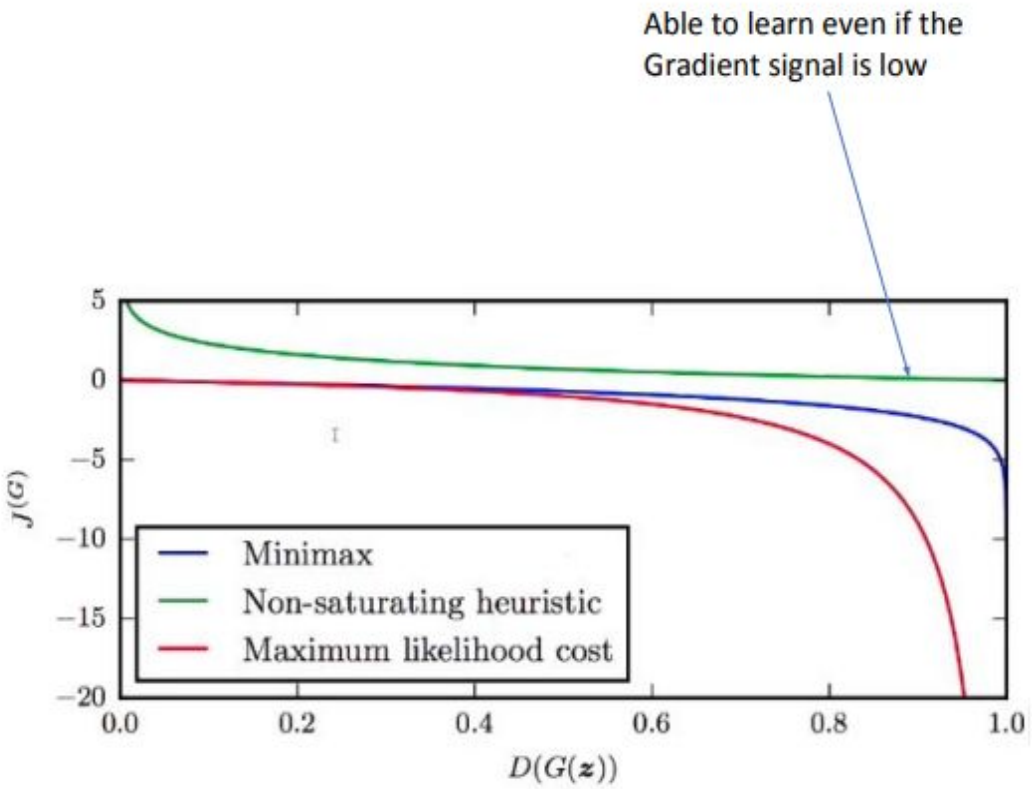
Comparison of Generator Losses

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

$$J^G = -\frac{1}{2}\,\mathrm{E}_z \log \mathrm{D}(\mathrm{G}(z))$$

$$J^G = -\frac{1}{2}\,\mathrm{E}_z \exp(\sigma^{-1}\mathrm{D}(\mathrm{G}(z)))$$

-> Generators cost is a function D(G(z))



Solutions:

- **Change Objectives:**

use the non-saturating heuristic objective, maximum-likelihood cost, etc.

- **Limit Discriminator:**

restrict the capacity of the discriminator.

- **Schedule Learning:**

try to balance training $D_{\theta d}$ and $G_{\theta g}$.

Problems with GANs

2. Non-Convergence

-> The generator and discriminator oscillate without reaching an equilibrium.

-> Simultaneous gradient descent is not guaranteed to converge for

minimax objectives.

- Goodfellow et al. only showed convergence when updates are made in the function space.
- The parameterization of $D_{\theta d}$ and $G_{\theta g}$ results in highly non-convex objective.
- In practice, training tends to oscillate – updates “undo” each other.

Solution for Non-Convergence:

6: Use Soft and Noisy Labels

- Label Smoothing, i.e. if you have two target labels: Real=1 and Fake=0, then for each incoming sample, if it is real, then replace the label with a random number between 0.7 and 1.2, and if it is a fake sample, replace it with 0.0 and 0.3 (for example).
  - Salimans et. al. 2016
- make the labels the noisy for the discriminator: occasionally flip the labels when training the discriminator

7: DCGAN / Hybrid Models

- Use DCGAN when you can. It works!
- if you cant use DCGANs and no model is stable, use a hybrid model : KL + GAN or VAE + GAN

Solution for Non-Convergence:

8: Use stability tricks from RL

- Experience Replay
  - Keep a replay buffer of past generations and occassionally show them
  - Keep checkpoints from the past of G and D and occassionaly swap them out for a few iterations
- All stability tricks that work for deep deterministic policy gradients
- See Pfau & Vinyals (2016)

9: Use the ADAM Optimizer

- optim.Adam rules!
  - See Radford et. al. 2015
- Use SGD for discriminator and ADAM for generator

Problems with GANs

3. Mode Collapse:

the generator distribution collapses to a small set of examples.

4. Mode Dropping:

the generator distribution doesn’t fully cover the data distribution.

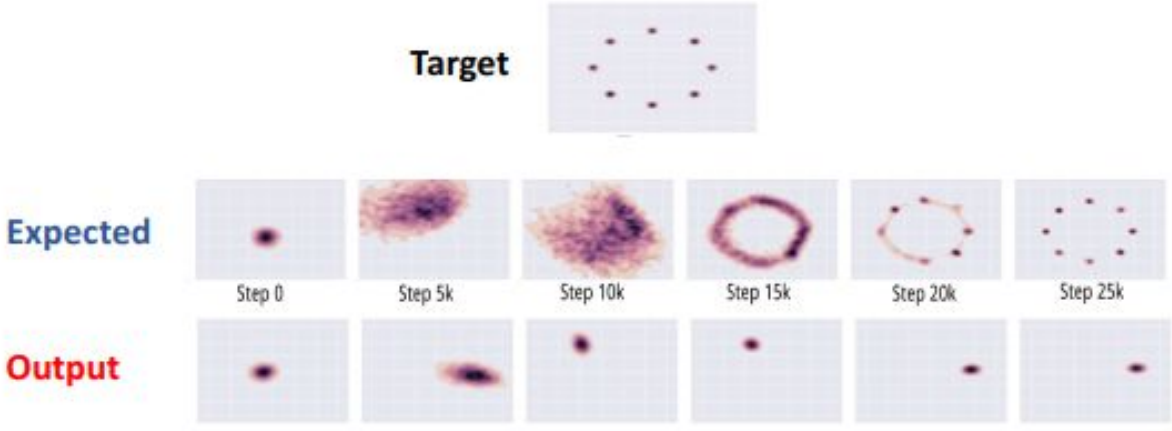
Problems: Mode Collapse and Mode Dropping

Generator Fails To Output Diverse samples

One Explanation: SGD may optimize the max-min objective

$$\max_{\theta_d} \min_{\theta_g} \mathbb{E}_{p_{\text{data}}} [\log D_{\theta_d}(\mathbf{x})] + \mathbb{E}_{p_z(\mathbf{z})} [\log (1 - D_{\theta_d}(G_{\theta_g}(\mathbf{z})))]$$

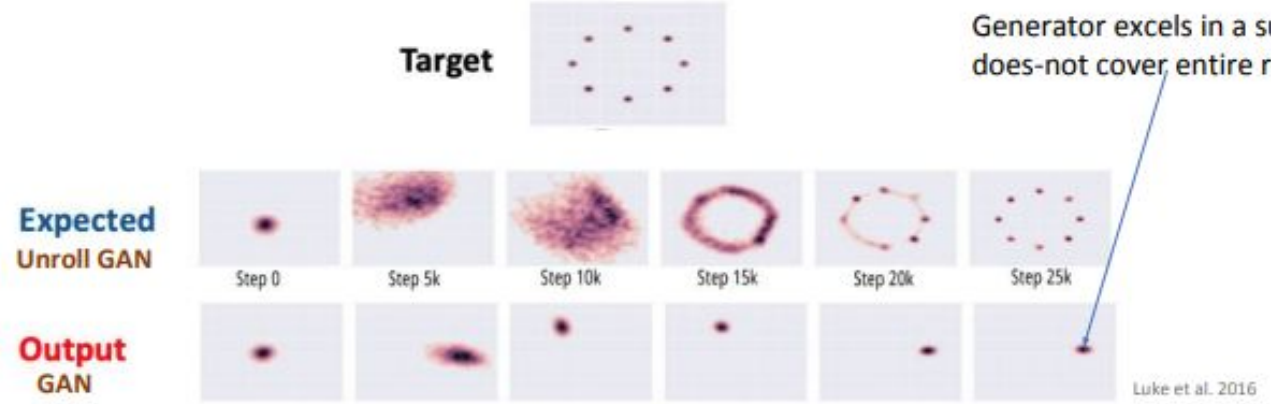
Intuition: the generator maps all $\mathbf{z}$ values to the $\hat{\mathbf{x}}$ that is mostly likely to fool the discriminator.



https://arxiv.org/abs/1701.0016

Problems: Mode Collapse and Mode Dropping

Generator excels in a subspace but does-not cover entire real distribution



Luke et al. 2016

Solution for Mode Collapse and Mode Dropping

Alternative Divergences

There are a large variety of divergence measures for distributions:

- **f-Divergences:** (e.g. Jensen-Shannon, Kullback-Leibler)

$$D_f\ (P\ ||Q) = \int_{\chi} q(\mathbf{x}) f(\frac{p(\mathbf{x})}{q(\mathbf{x})}) d\mathbf{x}$$

  - GANs [2], f-GANs [7], and more.
- **Integral Probability Metrics:** (e.g. Earth Movers Distance, Maximum Mean Discrepancy)

$$\gamma_F\ (P\ ||Q) = \sup_{f \in F} \left| \int f dP - \int f dQ \right|$$

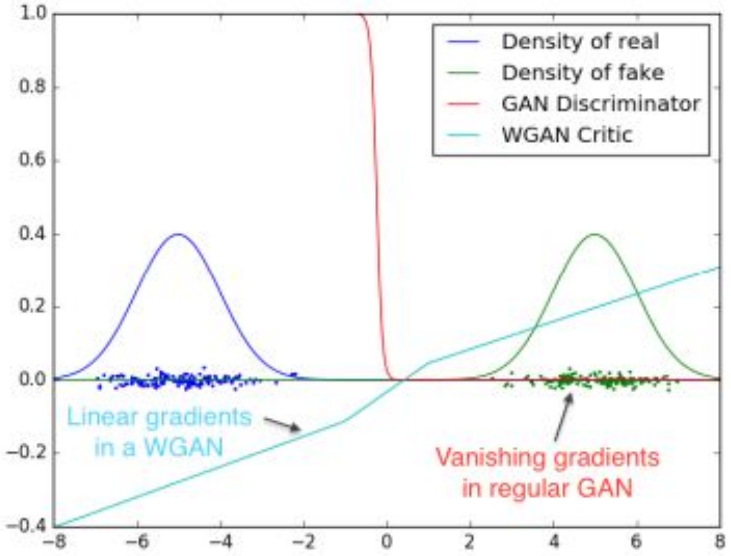
  - Wasserstein GANs [1], Fisher GANs [6], Sobolev GANs [5] and more.

Solution for Mode Collapse and Mode Dropping

Wasserstein GANs

Wasserstein GANs: Strong theory and excellent empirical results

- "In no experiment did we see evidence of mode collapse for the WGAN algorithm." [1]



https://arxiv.org/abs/1701.07875

References:

- https://github.com/nowozin/mlss2018-madrid-gan
- https://arxiv.org/abs/1701.00160
- https://www.alexirpan.com/2017/02/22/wasserstein-gan.html
- https://slazebni.cs.illinois.edu/spring17/lec11_gan.pdf
- https://people.cs.pitt.edu/~milos/courses/cs3750/lectures/class23.pdf
- http://www.youtube.com/watch?v=ktxhiKhWoEE&t=0m30s

Thank You

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack:Divija L
Teaching Assistant