

# Generative Adversarial Networks (part 2)

Raphael Olivier<sup>1</sup>  
Slides from Benjamin Striner<sup>1</sup>

<sup>1</sup>Carnegie Mellon University

November 13, 2019

# Table of Contents

- 1 Recap
- 2 Understanding Optimization Issues
- 3 GAN Training and Stabilization
- 4 Take Aways

# Table of Contents

- 1 Recap
- 2 Understanding Optimization Issues
- 3 GAN Training and Stabilization
- 4 Take Aways

# Recap

What did we talk about so far?

- What is a GAN?
- How do GANs work theoretically?
- What kinds of problems can GANs address?

# Recap

## What is a GAN?

- Train a generator to produce samples from a target distribution
- Discriminator guides generator
- Generator and Discriminator are trained adversarially

# Recap

How do GANs work theoretically?

- Discriminator calculates a divergence between generated and target distributions
- Generator tries to minimize the divergence

# Pseudocode

How can you build one yourself?

- Define a generator network that takes random inputs and produces an image
- Define a discriminator network that takes images and produces a scalar
  - Draw a random batch of  $Z$  from prior
  - Draw a random batch of  $X$  from data
  - Gradient descent generator weights w.r.t. generator loss
  - Gradient descent discriminator weights w.r.t. discriminator loss
- See recitations and tutorials for details

# Recap

What kinds of problems can GANs address?

- Generation
- Conditional Generation
- Clustering
- Semi-supervised Learning
- Representation Learning
- Translation

Any traditional discriminative task can be approached with generative models



# Summary

- Powerful tool for generative modeling
- Lots of potential
- Limited by pragmatic issues (stability)

# Table of Contents

- 1 Recap
- 2 Understanding Optimization Issues**
- 3 GAN Training and Stabilization
- 4 Take Aways

# Common Failures

GAN training can be tricky to diagnose

- “Mode collapse” generates a small subspace but does not cover the entire distribution:  
<https://www.youtube.com/watch?v=ktxhiKhWoEE>
- Some errors harder to describe:  
<https://www.youtube.com/watch?v=D5akt32hsCQ>
- Cause can be unclear
  - Discriminator too complicated?
  - Discriminator not complicated enough?

# Causes of Optimization Issues

- Simultaneous updates require a careful balance between players
- In general, two player games are not guaranteed to converge to the global optimum
- There is a stationary point but no guarantee of reaching it
- Adversarial optimization is a more general, harder problem than single-player optimization

# Simultaneous Updates

Why are updates simultaneous? Can you just train an optimal discriminator?

- For any given discriminator, the optimal generator outputs  $\forall Z : G(Z) = \operatorname{argmax}_X D(X)$
- The optimal discriminator emits 0.5 for all inputs, so isn't useful for training anything
- Optimal discriminator conditional on current generator and vice-versa
- Cannot train generator without training discriminator first
- Therefore generator and discriminator must be trained together

# Adversarial Balance

What kind of balance is required?

- If discriminator is under-trained, it guides the generator in the wrong direction
- If discriminator is over-trained, it is too “hard” and generator can’t make progress
- If generator trains too quickly it will “overshoot” the loss that the discriminator learned

# Factors Affecting Balance

What affects the balance?

- Different optimizers and learning rates
- Different architectures, depths, and numbers of parameters
- Regularization
- Train  $D$  for  $k_D$  iterations, train  $G$  for  $k_G$  iterations, repeat
- Train  $D$  and  $G$  for dynamic  $k_D$  and  $k_G$  iterations, depending on some metrics

# Adversarial Balance

What does this look like in practice?

- Target distribution is stationary 2D point (green)
- Generator produces a single moving 2D point (blue)
- Discriminator is a 2D linear function, represented by the colored background
- Watch oscillations as generator overshoots discriminator
- <https://www.youtube.com/watch?v=ebMei6bYeWw>



# Two-Player Games

Even a simple game, Rock, Paper, Scissors, might not converge using alternating updates.

- Player A prefers rock by random initialization
- Player B should therefore play only paper
- Player A should play only scissors
- Player B should play only rock
- ...

# Rock, Paper, Scissors

Why is this so unstable?

$$\mathbb{E}l_A = A_R B_S + A_P B_R + A_S B_P$$

- Global optimum
  - Both players select uniformly w.p. 0.33
  - Both players win, lose or draw w.p. 0.33
- Local optimum
  - Say player B plays  $(R, P, S)$  w.p.  $(0.4, 0.3, 0.3)$
  - Player A should play  $(R, P, S)$  w.p.  $(0, 1, 0)$
  - Player B wins w.p. 0.4
- What happens if you use gradient descent?

<https://www.youtube.com/watch?v=JmON4S0k104>

# Stationary Points

The fact that there is a stationary point does not mean you will converge to it

- Gradients can circle or point away from the minima
- Stationary point may not be stable, no local “well”
- Some degree of smoothness to the discriminator is required
  - Even if discriminator correctly labels generated points 0 and real points 1
  - Does not mean the gradient of the discriminator is in the right direction
  - Does not mean area around generated points is 0 and area around real points is 1

# Table of Contents

- 1 Recap
- 2 Understanding Optimization Issues
- 3 GAN Training and Stabilization**
- 4 Take Aways

# GAN Training

Ongoing research into “best” GAN

- Likely no silver-bullet
- Combinations of techniques work well
- Getting better every year

# GAN Training Techniques

We will discuss a sample of training/stabilization techniques

- Will not cover every idea people have tried
- Goal is to understand the types of techniques and research
- Will cover some interesting or historical ideas that aren't that great
- I am not endorsing all of the following techniques

# GAN Training Techniques

- Unrolled Generative Adversarial Networks
- Gradient descent is locally stable
- DRAGAN
- Numerics of GANs
- Improved Techniques for GANs
- Least-Squares GAN
- Instance Noise
- EBGAN
- WGAN
- WGAN-GP
- Spectral Normalized GAN
- Fisher GAN
- LapGAN
- ProgGAN

# Unrolled Generative Adversarial Networks

Optimize future loss, not current loss [MPPS16]

- Calculate the discriminator after a few SGD steps
- Find the generator that has the best loss on the future discriminator
- Differentiate through gradient descent



# UGAN Definition

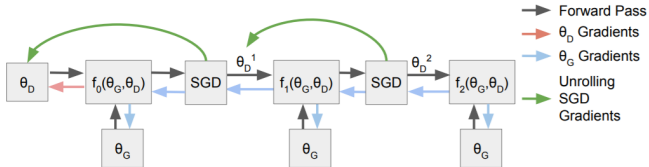
Think of it like chess. Make move that gives the best result after the opponent's move, not the best immediate reward.

$$\theta_D^0 = \theta_D$$

$$\theta_D^{k+1} = \theta_D^k + \eta^k \frac{\partial f(\theta_G, \theta_D^k)}{\partial \theta_D^k}$$

$$f_K(\theta_G, \theta_D) = f(\theta_G, \theta_D^K(\theta_G, \theta_D))$$

# UGAN Diagram



# UGAN Results

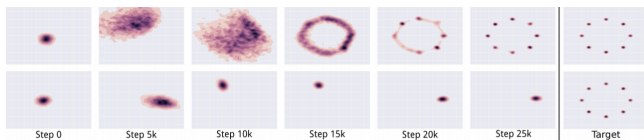


Figure 2: Unrolling the discriminator stabilizes GAN training on a toy 2D mixture of Gaussians dataset. Columns show a heatmap of the generator distribution after increasing numbers of training steps. The final column shows the data distribution. The top row shows training for a GAN with 10 unrolling steps. Its generator quickly spreads out and converges to the target distribution. The bottom row shows standard GAN training. The generator rotates through the modes of the data distribution. It never converges to a fixed distribution, and only ever assigns significant probability mass to a single data mode at once.

Gradient descent GAN optimization is locally stable

## Gradient descent GAN optimization is locally stable

Minimize the original objective as well as the gradient of the objective [NK17]

- Gradient of the discriminator is how much discriminator can improve
- If generator makes improvement, but discriminator gradient is large, discriminator can undo that improvement
- If generator makes improvement in a way that the discriminator gradient is zero, discriminator cannot undo that improvement

$$\ell_G = \ell_{G,0} + \eta \|\nabla \ell_D\|^2$$

# How to train your DRAGAN

Minimize the norm of the gradient in a region around real data  
[KAHK17]

- Minimize the norm of the gradient makes a function smoother
- Smooth in a random region around real data to smooth the discriminator

$$\lambda \mathbb{E}_{X, \epsilon} \max(0, \|\nabla D(X + \epsilon)\|^2 - k)$$

# The Numerics of GANs

GANs define a vector field [MNG17]

- Consider the joint field of generator and discriminator parameters
- Stationary points are where the gradient of each player w.r.t. its own parameters is 0
- Find these regions by minimizing the norm of the gradient of each player
- Regularization parameter balances between adversarial objective and consensus objective

$$L = L_0 + \lambda \|\nabla L_0\|_2^2$$

# Consensus

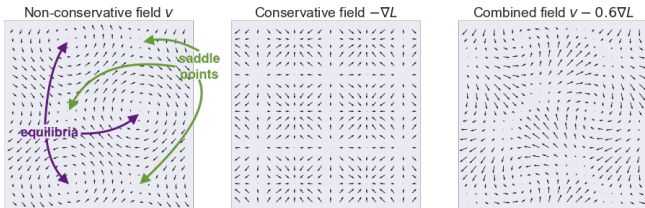
## Why “consensus”?

- Generator tries to minimize discriminator gradients as well as its own
- Discriminator tries to minimize generator gradients as well as its own
- Mutual de-escalation
- Encourages towards minima, maxima, and saddle points
- Small regularization parameter so hopefully finds the minima

# Vector Fields

https:

[//www.inference.vc/my-notes-on-the-numerics-of-gans/](https://www.inference.vc/my-notes-on-the-numerics-of-gans/)





# Improved Techniques for Training GANs

A collection of interesting techniques and experiments

- Feature matching
- Minibatch discrimination
- Historical averaging
- One-sided label smoothing
- Virtual batch normalization

# Feature Matching

Statistics of generated images should match statistics of real images

- Discriminator produces multidimensional output, a “statistic” of the data
- Generator trained to minimize  $L_2$  between real and generated data
- Discriminator trained to maximize  $L_2$  between real and generated data

$$\|\mathbb{E}_X D(X) - \mathbb{E}_Z D(G(Z))\|_2^2$$

# Minibatch Discrimination

Discriminator can look at multiple inputs at once and decide if those inputs come from the real or generated distribution

- GANs frequently collapse to a single point
- Discriminator needs to differentiate between two distributions
- Easier task if looking at multiple samples

# Historical Averaging

Dampen oscillations by encouraging updates to converge to a mean

- GANs frequently create a cycle or experience oscillations
- Add a term to reduce oscillations that encourages the current parameters to be near a moving average of the parameters

$$\left\| \theta - \frac{1}{t} \sum_i^t \theta_i \right\|_2^2$$

# One-sided label smoothing

Don't over-penalize generated images

- Label smoothing is a common and easy technique that improves performance across many domains
  - Sigmoid tries hard to saturate to 0 or 1 but can never quite reach that goal
  - Provide targets that are  $\epsilon$  or  $1 - \epsilon$  so the sigmoid doesn't saturate and overtrain
- Experimentally, smooth the real targets but do not smooth the generated targets when training the discriminator

# Virtual Batch Normalization

Use batch normalization to accelerate convergence

- Batch normalization accelerates convergence
- However, hard to apply in an adversarial setting
- Collect statistics on a fixed batch of real data and use to normalize other data

# Least-Squares GAN

Use an  $L_2$  loss instead of cross-entropy [MLX<sup>+</sup>16]

- Generator tries to minimize  $L_2$  loss  $\|a - D(G(Z))\|_2^2$
- Generator tries to minimize  $L_2$  loss  $\|b - D(G(Z))\|_2^2 + \|c - D(G(Z))\|_2^2$
- For example,  $a = 1$ ,  $b = 0$ ,  $c = 1$

Less squashing and large values than cross-entropy

# Amortised MAP Inference for Image Super-Resolution

Part of a larger paper with several modifications and experiments  
[SCT<sup>+</sup>16]

- “Instance Noise”
- Add noise to generated and real images
- Smooth the function learned by the discriminator

Simple and effective



# Energy-based Generative Adversarial Network

Simplify the loss function, removing powers and exponents  
[ZML16]

- No longer easily described using JS divergence or something similar
- Greatly simplified and linear
- Gradients are neither squashed nor explode

$$L_D = \mathbb{E}_X D(X) + \mathbb{E}_Z [m - D(G(Z))]^+$$

$$L_G = D(G(z))$$

# Wasserstein GAN

Further simplified and theoretically grounded [ACB17]

- Solve Wasserstein “earth mover” distance
- Provide a smoother gradient
- Constrain Lipschitz of discriminator to 1
- Harder to overtrain discriminator

$$L_D = \mathbb{E}_X D(X) - \mathbb{E}_Z D(G(Z))$$

$$L_G = \mathbb{E}_Z D(G(Z))$$

# Wasserstein Distance

The total  $\sum \text{mass} \times \text{distance}$  required to transform one distribution to another.

- Intuitive, symmetric measure of divergence
- Hard to calculate because requires solving “optimal transport”
- You have some distribution of products at some warehouses and you need to make some other distribution of products at those warehouses
- Move the products to the target distribution minimizing  $\sum \text{mass} \times \text{distance}$
- Creating the optimal “transport plan” can be NP-hard

# KL Example

$$KL(p||q) = \int_x p(x) \log \frac{p(x)}{q(x)}$$

- Real data is a point mass at 0
- Generated data is a point mass at  $\theta$
- If  $\theta \neq 0$ ,  $p(0) \log \frac{p(0)}{q(0)} = 1 \frac{1}{0} = \infty$
- If  $\theta = 0$ ,  $1 \log \frac{1}{1} = 0$
- **Not differentiable w.r.t.  $\theta$**

## JS Example

Calculate the average distribution and calculate the average KL to the average.

$$m(x) = \frac{p(x) + q(x)}{2}$$
$$JS(p||q)$$

- Real data is a point mass at 0
- Generated data is a point mass at  $\theta$
- If  $\theta \neq 0$ ,  $\frac{1}{2} [1 \log \frac{1}{0.5} + 1 \log \frac{1}{0.5}] = \log 4$
- If  $\theta = 0$ ,  $\frac{1}{2} [1 \log \frac{1}{1} + 1 \log \frac{1}{1}] = 0$
- **Not differentiable w.r.t.  $\theta$**

# Wasserstein Example

Calculate the mass times the distance in one dimension

- Real data is a point mass at 0
- Generated data is a point mass at  $\theta$
- EM is  $|\theta|$
- **Differentiable w.r.t.  $\theta$ !**

# Kantorovich Dual

Result showing the EM can be calculated using a dual method

$$\sup_{\phi, \psi} \left( \int_X \phi(x) d\mu(x) + \int_Y \psi(y) d\nu(y) \right)$$

$$\phi(x) - \psi(y) \leq c(x, y)$$

Provide justification for GAN techniques.

## EM v JS Visualized

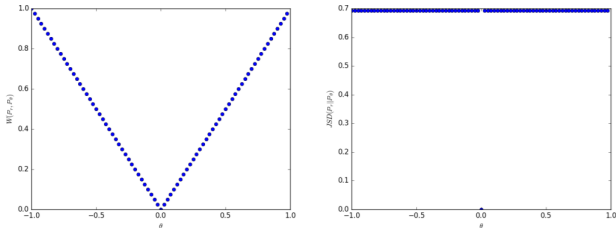


Figure 1: These plots show  $\rho(\mathbb{P}_\theta, \mathbb{P}_0)$  as a function of  $\theta$  when  $\rho$  is the EM distance (left plot) or the JS divergence (right plot). The EM plot is continuous and provides a usable gradient everywhere. The JS plot is not continuous and does not provide a usable gradient.



## GAN vs WGAN

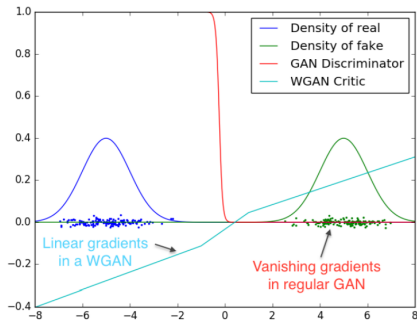


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

# So what's the catch?

Constrain the Lipschitz of an arbitrary function

- Function is nonlinear
- Cannot calculate  $\frac{\|D(x)-D(y)\|_2}{\|x-y\|_2}$  for all pairs
- Just clip the weights to some value
- Constraining the  $L_\infty$  of all weight matrices upper bounds the Lipschitz of the function
- Bound is not great
- In practice, leads to poor discriminators

# Wasserstein GAN-GP

Demonstrate the flaws with gradient clipping [GAA<sup>+</sup>17]

- Propose an alternative method of constraining the Lipschitz
- Lipschitz should be 1 almost everywhere
- Calculate the gradient of  $D$  at random samples
- Add a penalty of the mean squared distance between the gradient and 1

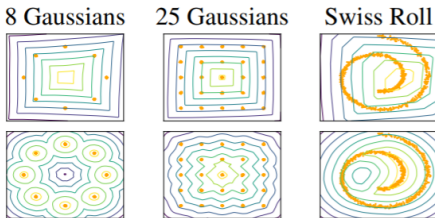
$$L = \mathbb{E}_X(D(X)) - \mathbb{E}_Z(D(G(Z))) + \lambda \mathbb{E}_{X'} (\|\nabla D(X')\|_2 - 1)^2$$

# Where to sample?

## How to sample?

- Cannot constrain gradient everywhere
- Can only penalize gradient at some samples
- Use samples that are random linear interpolations between real and fake data
- Keeps discriminator smooth over the relevant region

# Discriminator Visualization



(a) Value surfaces of WGAN critics trained to optimality on toy datasets using (top) weight clipping and (bottom) gradient penalty. Critics trained with weight clipping fail to capture higher moments of the data distribution. The ‘generator’ is held fixed at the real data plus Gaussian noise.

# Spectral Normalized GAN

Bound the Lipschitz using the spectral norm of each layer  
[MKKY18]

- Lipschitz of a single matrix multiplication can be calculated using a power iteration

$$\sup_x \frac{\|xA\|_2}{\|x\|_2} = \|A\|_2^*$$

- Lipschitz of an MLP can be upper bounded by product of each layer
- Constrain each layer by calculating  $\hat{W} = \frac{W}{\|W\|_2^*}$

# Fisher GAN

Fisher IPM does not require constraining the discriminator [MS17]

- Constraint is imposed by the loss function itself
- Difference between  $D(G(Z))$  and  $D(X)$  is scaled by  $D(G(Z))$  and  $D(X)$

$$\sup_f \frac{\mathbb{E}_X D(X) - \mathbb{E}_Z D(G(Z))}{\sqrt{\frac{1}{2}\mathbb{E}_X D(X)^2 + \frac{1}{2}\mathbb{E}_Z D(G(Z))^2}}$$

# LapGAN

Break image generation into smaller chunks [DCSF15]

- GAN generates small, blurry image
- CGAN sharpens and enlarges image slightly
- Repeat step 2 until desired size
- Separate, small GANs simplify training



# Progressive Growing of GANs

Gradually expand networks, like a curriculum [KALL17]

- Simple generator learns with simple discriminator
- Gradually add layers to generator and discriminator to produce larger images

# Progressive GAN Visualization

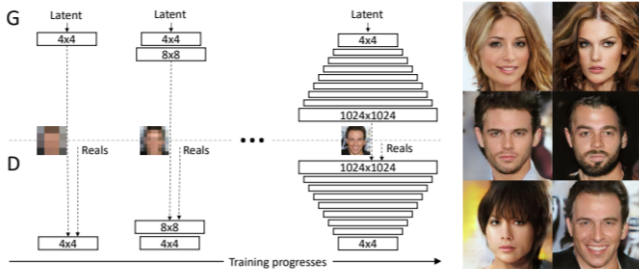


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of  $4 \times 4$  pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here  $N \times N$  refers to convolutional layers operating on  $N \times N$  spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. On the right we show six example images generated using progressive growing at  $1024 \times 1024$ .

# Table of Contents

- 1 Recap
- 2 Understanding Optimization Issues
- 3 GAN Training and Stabilization
- 4 Take Aways**

## Common “Gotchas”

- You can only reduce an NP problem into another NP problem
- Unrolling or second gradients are computationally expensive (UGAN, CGAN)
- Sampling-based methods can be unreliable (WGAN-GP)
- Bound-based methods can easily learn poor local optima (WGAN, SNGAN)

# Lessons

Some major trends in the research

- Regularize and smooth the discriminator, especially in some region around the real and generated points
- Update the players towards some sort of consensus or future stable region, not just the immediate greedy update
- Simplify networks and losses to eliminate nonlinearities
- Constrain the “complexity” of the discriminator so you can train reliably without overtraining
- Grow or chain GANs to reduce complexity, learn a curriculum

# Where is current research?

Current techniques have impressive results

- Lots of GPUs and tuning required
- But the results are constantly improving
- Better techniques allow larger models, always pushing the limits
- Mostly images but other modalities are in-progress

# Where is this going?

Ultimately, discriminator is calculating a divergence/loss function




- Neural networks are universal function approximators
- Loss functions are functions
- How can we measure/constrain the “complexity” of a neural network?
- How can we architect a neural network to learn a meaningful loss function?

# References I





-  Martin Arjovsky, Soumith Chintala, and Léon Bottou, *Wasserstein GAN*, arXiv e-prints (2017), arXiv:1701.07875.
-  Emily L. Denton, Soumith Chintala, Arthur Szlam, and Robert Fergus, *Deep generative image models using a laplacian pyramid of adversarial networks*, CoRR **abs/1506.05751** (2015).
-  Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville, *Improved training of wasserstein gans*, CoRR **abs/1704.00028** (2017).






## References II

-  Naveen Kodali, Jacob D. Abernethy, James Hays, and Zolt Kira, *How to train your DRAGAN*, CoRR [abs/1705.07215](#) (2017).
-  Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen, *Progressive growing of gans for improved quality, stability, and variation*, CoRR [abs/1710.10196](#) (2017).
-  Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida, *Spectral normalization for generative adversarial networks*, CoRR [abs/1802.05957](#) (2018).

## References III

-  Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang, *Multi-class generative adversarial networks with the L2 loss function*, CoRR [abs/1611.04076](#) (2016).
-  Lars M. Mescheder, Sebastian Nowozin, and Andreas Geiger, *The numerics of gans*, CoRR [abs/1705.10461](#) (2017).
-  Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein, *Unrolled generative adversarial networks*, CoRR [abs/1611.02163](#) (2016).
-  Youssef Mroueh and Tom Sercu, *Fisher GAN*, CoRR [abs/1705.09675](#) (2017).

## References IV

-  Vaishnavh Nagarajan and J. Zico Kolter, *Gradient descent GAN optimization is locally stable*, CoRR [abs/1706.04156](https://arxiv.org/abs/1706.04156) (2017).
-  Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár, *Amortised MAP inference for image super-resolution*, CoRR [abs/1610.04490](https://arxiv.org/abs/1610.04490) (2016).
-  Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun, *Energy-based generative adversarial network*, CoRR [abs/1609.03126](https://arxiv.org/abs/1609.03126) (2016).