Generative Adversarial Networks (part 2)

Benjamin Striner

CMU 11-785

March 26, 2018

Overview

Overview

- Last week
  - What makes generative networks unique?
  - What is a generative adversarial network (GAN)?
  - What kinds of problems can we apply GANs to?
- This week
  - How do we optimize GANs?
  - What problems do GANs have?
  - What current work is being done?

Recap

Generative vs. Discriminative Networks

- Discriminative networks require inputs X and labels Y and attempt to model the conditional distribution $P(Y \mid X)$.
- Generative networks do not require labels although they may be included. They variously attempt to model $P(X)$, $P(X \mid Y)$, $P(X, Y)$, etc.

VAE Recap

It is important to compare and contrast GANs with VAEs, which serve a similar purpose but came earlier.

Variational Autoencoders

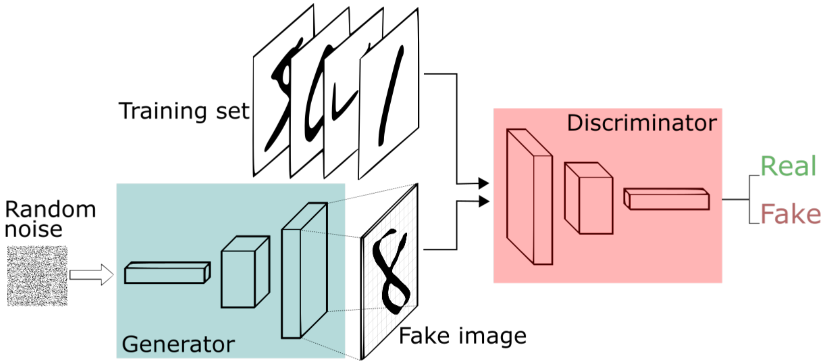
- Similar to autoencoders
  - Model an encoder $P(Z \mid X)$
  - Model a decoder $P(X \mid Z)$
- Model learns/infers Z; Z is not a part of the training data
- Regularized such that Z matches a prior (lets call it q)
- Since we can sample from Z directly, we can generate samples directly from the latent space

$$J = -\log p(X \mid Z) + KL(p(Z \mid X) \,||\, q(Z))$$

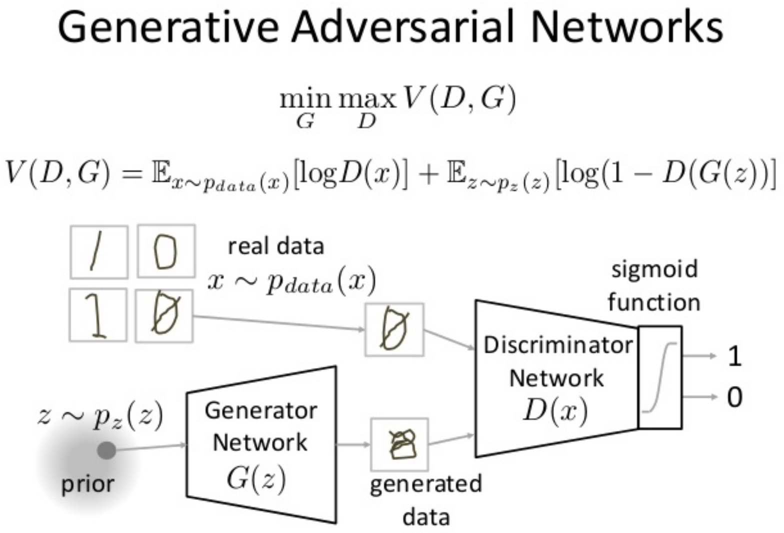
Generative Adversarial Networks

- Generative adversarial networks (GANs) are relatively new. They have spawned a flurry of activity and progress in recent years. Goodfellow et al. (2014)
- GANs are a new way to build generative models $P(X)$. GANs may have more flexibility and potential than VAEs. They produce sharper and cleaner results than VAEs. However, they are much harder to train and have their own set of issues.

Conceptual Diagram 1 (the simple one)



Conceptual Diagram 2 (the math one)

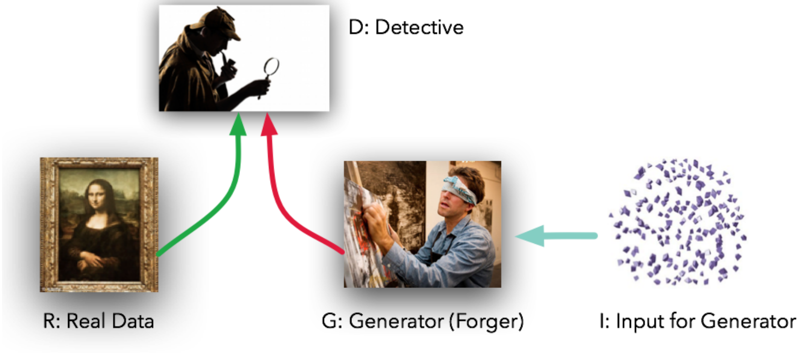


Generative Adversarial Networks

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

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

Conceptual Diagram 3 (the fun one)



See blog post (Nag (2017))

Implications of GANs

- Our metric by which we are comparing images is now defined by a neural network. That means any biases implicit in the network are now a part of our distance/loss.
- For example, imagine you have a discriminator that uses max-pooling and is therefore somewhat shift invariant. Your discriminator cannot discriminate based on small shifts, so your loss function is invariant to shifts. That should result in the network generating samples that are slight shifts of real data. Your CNN is really good at looking at local features but not as good at shifts in data.
- This is a new way of thinking about losses/distances/metrics. We can use neural networks instead of traditionally defined metrics.

GANs v VAEs

- VAEs are guaranteed to converge but GANs are not
- GANs potentially capture perceptual similarity and tend to generate sharper images than VAEs

What can we do with GANs?

We talked about some of the architectures that people have built around GANs. The key take-away is that there are many ways to work the core GAN idea into more complicated models.

- Conditional GAN: Mirza and Osindero (2014)
- LapGAN: Denton et al. (2015)
- DCGAN: Radford et al. (2015)
- CatGAN: Springenberg (2015)
- GRAN: Im et al. (2016)
- InfoGAN: Chen et al. (2016)
- AAE: Makhzani et al. (2015)
- BiGAN: Donahue et al. (2016)

Too good to be true?

The major current roadblocks with GANs are various optimization issues.

Why are GANs different?

GAN optimization is fundamentally different from other neural networks.

- Gradient descent is relatively well established
- Loss functions don't change much
- Most deep learning research has focused on new components to use within the standard single-player framework (dropout, batchnorm, relu, etc.)
- GANs are an area of research where the objectives and descent methods are still in flux

Causes of Optimization Issues

Causes of Optimization Issues

Many theories on why GANs are hard to train. Some contributing factors below.

- Two player games do not always converge using gradient descent
- Simultaneous updates require a careful balance between the two players
- There is a stationary point but no guarantee of reaching it
- Generated points tend to "herd" to probable regions, causing "mode collapse"
- Adversarial optimization is a more general, harder problem than single-player optimization
- Discriminator is highly nonlinear, gradient tends to be noisy or non-informative

Simultaneous Updates

- It is important to understand that both the generator and discriminator are trying to learn "moving targets". Both networks are trained simultaneously.
- The discriminator needs to update based on how well the generator is doing. The generator is constantly updating to improve performance on the discriminator. These two need to be balanced correctly to achieve stable learning instead of chaos.

Balance between D and G

- The discriminator tells the difference between real and generated data. Generator uses the discriminator to perform updates.
- If discriminator is not sufficiently trained, using it to train the generator will make the generator worse
- If discriminator is over-trained, tends to have no gradient or meaningless gradients

Balancing D and G in practice

- The balance between D and G is dependent on the complexity of the corresponding networks
- You can also control the balance by simply adjusting the learning rate
- Many papers ensure D is ahead of G by training using an imbalanced schedule. Something like train D 5 batches for each time G is trained on 1.
- Some papers recommend a dynamic schedule. For example, training D until the loss gets to some value before allowing G to train.
- Many factors contribute to how fast each learns relative to the other, such as learning rate, schedule, and regularization

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

What happens in practice?

- Simple 2 player game: rock, paper scissors. Shows that gradient descent does not always converge. `https://www.youtube.com/watch?v=JmON4S0kl04`
- Simplest GAN possible. Generator produces a single 2D point, discrminator is a single neuron, real data is a single point `https://www.youtube.com/watch?v=ebMei6bYeWw`
- 1-D GAN learning a normal distribution `https://www.youtube.com/watch?v=mObnwR-u8pc`
- Great video by Ian Goodfellow (40 mins but please watch if you have time) `https://www.youtube.com/watch?v=HN9NRhm9waY`

Common Failures

Common Failures

There are several common types of GAN failures that provide intuition into ways to make GANs better.

- "Mode collapse" GAN generates a subspace really well but doesn't cover the entire real distribution. For example, train on MNIST and it only generates threes and eights. `https://www.youtube.com/watch?v=ktxhiKhWoEE`
- Sometimes GANs enter into clear cycles. They seem to generate a single digit relatively well, then start generating a different digit, etc. Looks like "mode collapse" on a rotating set of samples, but it does not differentiate.
- Sometimes hard to describe failures, but videos like this are relatively typical. `https://www.youtube.com/watch?v=D5akt32hsCQ`

Optimization Techniques

Optimization Techniques

- The main problem with GANs is that they are tricky to train
- There are many tricks to train them better but some of those tricks do away with what makes GANs so special
- Not every trick works all the time or in combination with other tricks
- Most papers claim to have the golden bullet
- Best current solution is really a combination of techniques

Optimization Techniques

We will be discussing some proposed techniques for GAN training. Understand that this is still an active area of research. The important take-away is what kinds of things have been proposed and how different GANs are from traditional networks.

- Unrolled GANs, Metz et al. (2016)
- Improved Techniques for Training GANs, Salimans et al. (2016)
- Least-Squares GAN, Mao et al. (2016)
- Amortised MAP Inference, Kaae Sønderby et al. (2016)
- EBGAN, Zhao et al. (2016)
- WGAN, Arjovsky et al. (2017)
- WGAN-GP, Gulrajani et al. (2017)
- GAN optimization is locally stable, Nagarajan and Kolter (2017)
- Numerics of GANs, Mescheder et al. (2017)
- DRAGAN, Kodali et al. (2017)
- Progressive GAN, Karras et al. (2017)
- Lagrangian GAN, Chen et al. (2018)

Unrolled Generative Adversarial Networks

GANs have trouble reaching a stationary state. By making decisions based on the opponents reactions, the model can achieve a steady state. Metz et al. (2016)

$$\theta_D^0 = \theta_D$$

$$\theta_D^{k+1} = \theta_D^k + \eta^k \frac{\mathrm{d}f\left(\theta_G, \theta_D^k\right)}{\mathrm{d}\theta_D^k}$$

$$\theta_D^* = \lim_{k \to \infty} \theta_D^k,$$

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

UGAN Diagram

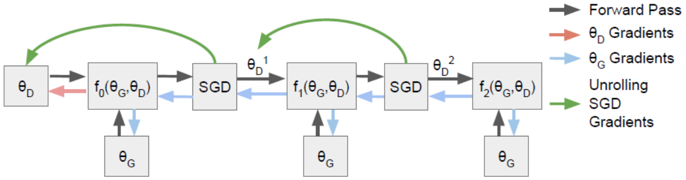


Figure 1: An illustration of the computation graph for an unrolled GAN with 3 unrolling steps. The generator update in Equation 10 involves backpropagating the generator gradient (blue arrows) through the unrolled optimization. Each step k in the unrolled optimization uses the gradients of f_k with respect to θ_D^k, as described in Equation 7 and indicated by the green arrows. The discriminator update in Equation 11 does not depend on the unrolled optimization (red arrow).

What does a UGAN do?

- Think of it like chess. You want to make the move with the best result after the opponent's move, not simply the best immediate reward.
- Generator makes update that is best for the next discriminator, not just current discriminator.

UGAN Toy Problem

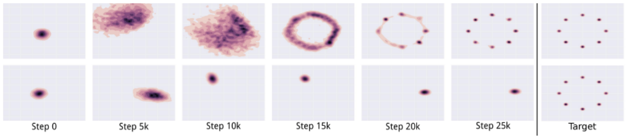


Figure 2: Unrolling the discriminator stabilizes GAN training on a toy 2D mixture of Gaussians dataset. Columns show a heatmap of the generator distribution after increasing numbers of training steps. The final column shows the data distribution. The top row shows training for a GAN with 10 unrolling 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.

UGAN MNIST

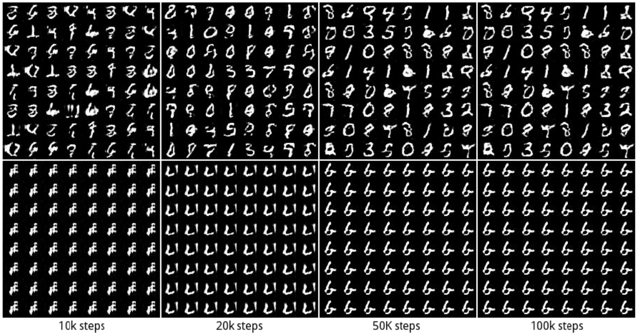


Figure 3: Unrolled GAN training increases stability for an RNN generator and convolutional discriminator trained on MNIST. The top row was run with 20 unrolling steps. The bottom row is a standard GAN, with 0 unrolling steps. Images are samples from the generator after the indicated number of training steps.

Improved Techniques for Training GANs

A large collection of techniques that can be used in combination to improve GAN training Salimans et al. (2016)

- Feature matching
- Minibatch Discrimination
- Historical Averaging
- One-sided Label Smoothing
- Virtual Batch Normalization

Improved Techniques: 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

$$||\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\mathbf{f}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\mathbf{f}(G(\boldsymbol{z}))||_2^2$$

Improved Techniques: Minibatch discrimination

- It is common for generators to collapse to a single point; need to encourage generators to span the entire real distribution
- A type of "herding". There is nothing telling generated data to spread out.
- Difficult for discriminator to represent a function that wants diversity. Either discriminator gives a high value and all of the generated points want to go there or it gives a low value and all of the generated points want to leave
- By making discriminator a function of multiple samples, it can encourage diversity

Improved Techniques: Historical Averaging

- Another common failure of GANs is creating a cycle
- Add a term that reduces oscillations

$$||\theta - \frac{1}{t}\sum_{i=1}^{t}\theta[i]||^2$$

Improved Techniques: One-sided label smoothing

- Label smoothing has been shown to improve performance in any number of situations
- Smoothing has been shown to improve GAN performance as well
- However, if you smooth the negative labels, there is some mass wherever there are generated samples, so generated samples have less incentive to move to other regions.
- Smoothing just the positive labels prevents over-saturation but avoids some drawbacks of smoothing both labels

Improved Techniques: Virtual Batch Normalization

- Desire to make samples within a batch less dependent
- Statistics collected based on a fixed reference batch and applied to other batch
- Expensive; requires running forward pass on both reference batch and training batch
- Only applied to generator

Least-Squares GAN (LSGAN)

Simple idea to avoid saturation. Train to minimize L2 distance instead of cross-entropy. Mao et al. (2016)

$$\begin{aligned}\min_D V_{\text{LSGAN}}(D) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right]\\ \min_G V_{\text{LSGAN}}(G) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right],\end{aligned} \tag{2}$$

Amortised MAP Inference for Image Super-resolution

Several techniques for GAN training. One simple but effective idea is adding noise to generated and real images. Seems to smooth the function learned by the discriminator, at least in a small region around the real and generated images. Kaae Sønderby et al. (2016)

Energy-based Generative Adversarial Network

Much simpler objective function based on energy instead of probability. Roughly a predecessor to WGAN which is currently very popular. Linear functions tend to behave better than nonlinear functions. Zhao et al. (2016)

$$\mathcal{L}_D(x,z) = D(x) + [m - D(G(z))]^+$$
$$\mathcal{L}_G(z) = D(G(z))$$

Wasserstein GAN

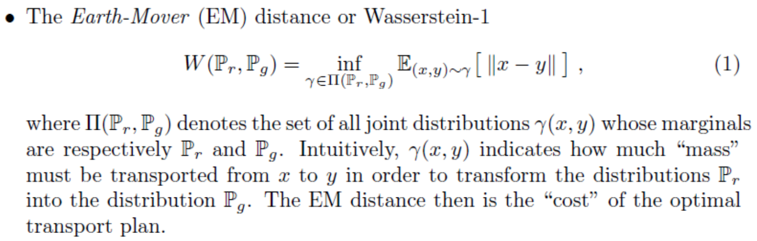
A theoretically grounded improvement on GANs. Makes connections between GANs and the dual of the earth-mover distance. Arjovsky et al. (2017)

- Uses a linear loss instead of a logarithmic loss; much simpler equations as a result and avoids certain issues with saturation
- Discriminator outputs are bounded by imposing constraints on the discriminator (weight clipping)
- Provides a smoother gradient to the generator than traditional GAN

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \le 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

What is Wasserstein/Earth Mover distance?

Earth Mover is the total $\sum$ *mass* $*$ *distance* required to transform one distribution to another. Analogous to how much work to move piles of earth.



- The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big] \,, \tag{1}$$

  where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the "cost" of the optimal transport plan.

Comparison with other distances

Example 1 (Learning parallel lines). Let $Z \sim U[0,1]$ the uniform distribution on the unit interval. Let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$ (a 0 on the x-axis and the random variable Z on the y-axis), uniform on a straight vertical line passing through the origin. Now let $g_\theta(z) = (\theta, z)$ with θ a single real parameter. It is easy to see that in this case,

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|$,
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \, , \\ 0 & \text{if } \theta = 0 \, , \end{cases}$
- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0 \, , \\ 0 & \text{if } \theta = 0 \, , \end{cases}$
- and $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0 \, , \\ 0 & \text{if } \theta = 0 \, . \end{cases}$

EM vs JS

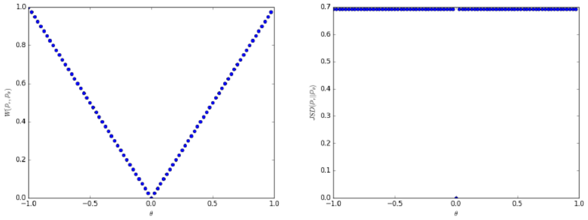


Figure 1: These plots show $\rho(\mathbb{P}_\theta, \mathbb{P}_0)$ as a function of θ when ρ is the EM distance (left plot) or the JS divergence (right plot). The EM plot is continuous and provides a usable gradient everywhere. The JS plot is not continuous and does not provide a usable gradient.

GAN v WGAN Discriminator

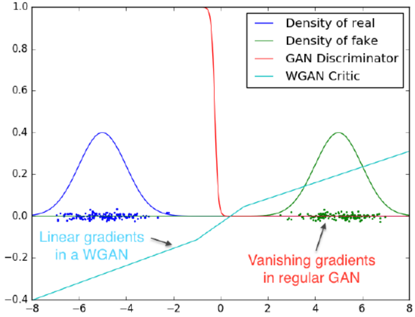


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

Connection between Lipshitz and Wasserstein

- Solving the dual of the Wasserstein metric involves optimization over the space of functions with Lipschitz constant of 1
- GANs are closely related to duals/Lagrangians
    - Early GANs did not make the connection
    - Recent literature makes more parallels

Wasserstein GAN

- Claims more stability than traditional GAN
- Loss value is more meaningful and corresponds to how good samples are
- Less tuning required in terms of generator/discriminator balance
- In practice, doesn't actually work so well (see WGAN-GP)

Improved Training of Wasserstein GANs

An improvement on WGANs that is much more stable. Variously called IWGAN or WGAN-GP. Gulrajani et al. (2017)

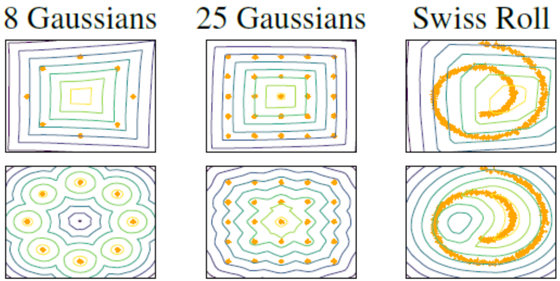
- Similar to WGAN
- Adds a gradient penalty such that the gradient of the output w.r.t. the input should be about 1
- WGAN attempts to make Lipschitz constant of discriminator bounded in a round-about way, WGAN-GP directly regularizes the Lipschitz constant.
- WGAN does not require sampling, WGAN-GP requires sampling
- Regularize the gradient at samples which are linear interpolations between real and generated data

$$L = \underbrace{\underset{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}{\mathbb{E}} [D(\tilde{\boldsymbol{x}})] - \underset{\boldsymbol{x} \sim \mathbb{P}_r}{\mathbb{E}} [D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \underset{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}{\mathbb{E}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]}_{\text{Our gradient penalty}} .$$

WGAN-GP Implementation

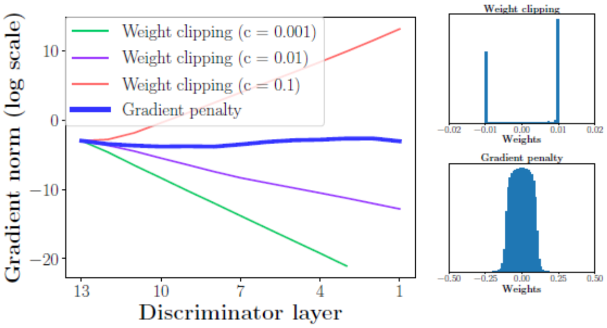
- Train discriminator on generated and real samples as with WGAN
- Sample data that is at random linear interpolations between generated and real samples
- Calculate gradient of discriminator w.r.t. inputs at those points
- Add regularization term such that L2 of gradient should be about 1

IWGAN v WGAN Results



(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.

IWGAN v WGAN Behind-the-scenes



(b) (left) Gradient norms of deep WGAN critics during training on the Swiss Roll dataset either explode or vanish when using weight clipping, but not when using a gradient penalty. (right) Weight clipping (top) pushes weights towards two values (the extremes of the clipping range), unlike gradient penalty (bottom).

Gradient descent GAN optimization is locally stable

Adds an additional regularization term where the generator tries to minimize the gradient that the discriminator can work with. Very similar in spirit to the unrolled GAN with only one step and to the numerics of GANs discussed next. Nagarajan and Kolter (2017)

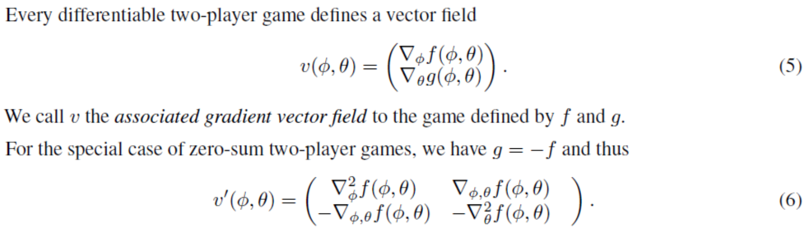
$$\boldsymbol{\theta}_{\mathrm{G}} := \boldsymbol{\theta}_{\mathrm{G}} - \alpha \nabla_{\boldsymbol{\theta}_{\mathrm{G}}} \left(V(D_{\boldsymbol{\theta}_{\mathrm{D}}}, G_{\boldsymbol{\theta}_{\mathrm{G}}}) + \eta \|\nabla_{\boldsymbol{\theta}_{\mathrm{D}}} V(D_{\boldsymbol{\theta}_{\mathrm{D}}}, G_{\boldsymbol{\theta}_{\mathrm{G}}})\|^2 \right)$$

The Numerics of GANs

An alternative strategy for optimizing GANs. Attempts to reduce the gradients of the loss. Intuitively, the parameters are regularized towards stable regions. Mescheder et al. (2017)

- Stable regions are where the gradient of each player w.r.t. its own weights is 0
- Can find these regions my minimizing the norm of the gradient of each player
- Regularization parameter balances between playing the two-player game (adversarial) and searching for a stable region (consensus)

The Numerics of GANs: Games define a vector field



Every differentiable two-player game defines a vector field

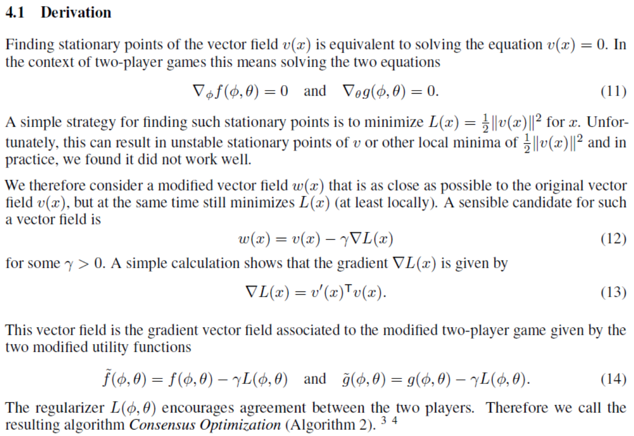
$$v(\phi, \theta) = \begin{pmatrix} \nabla_\phi f(\phi, \theta) \\ \nabla_\theta g(\phi, \theta) \end{pmatrix}. \tag{5}$$

We call v the *associated gradient vector field* to the game defined by f and g.

For the special case of zero-sum two-player games, we have $g = -f$ and thus

$$v'(\phi, \theta) = \begin{pmatrix} \nabla^2_\phi f(\phi, \theta) & \nabla_{\phi,\theta} f(\phi, \theta) \\ -\nabla_{\phi,\theta} f(\phi, \theta) & -\nabla^2_\theta f(\phi, \theta) \end{pmatrix}. \tag{6}$$

The Numerics of GANs: Find stationary points in the field



4.1 Derivation

Finding stationary points of the vector field $v(x)$ is equivalent to solving the equation $v(x) = 0$. In the context of two-player games this means solving the two equations

$$\nabla_\phi f(\phi, \theta) = 0 \quad \text{and} \quad \nabla_\theta g(\phi, \theta) = 0. \tag{11}$$

A simple strategy for finding such stationary points is to minimize $L(x) = \frac{1}{2}\|v(x)\|^2$ for x. Unfortunately, this can result in unstable stationary points of v or other local minima of $\frac{1}{2}\|v(x)\|^2$ and in practice, we found it did not work well.

We therefore consider a modified vector field $w(x)$ that is as close as possible to the original vector field $v(x)$, but at the same time still minimizes $L(x)$ (at least locally). A sensible candidate for such a vector field is

$$w(x) = v(x) - \gamma \nabla L(x) \tag{12}$$

for some $\gamma > 0$. A simple calculation shows that the gradient $\nabla L(x)$ is given by

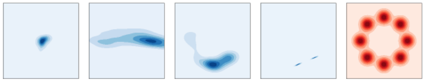
$$\nabla L(x) = v'(x)^\mathsf{T} v(x). \tag{13}$$

This vector field is the gradient vector field associated to the modified two-player game given by the two modified utility functions

$$\tilde{f}(\phi, \theta) = f(\phi, \theta) - \gamma L(\phi, \theta) \quad \text{and} \quad \tilde{g}(\phi, \theta) = g(\phi, \theta) - \gamma L(\phi, \theta). \tag{14}$$

The regularizer $L(\phi, \theta)$ encourages agreement between the two players. Therefore we call the resulting algorithm *Consensus Optimization* (Algorithm 2). [3] [4]

The Numerics of GANs: Results


(a) Simultaneous Gradient Ascent

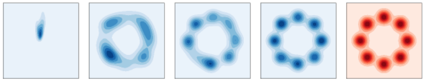
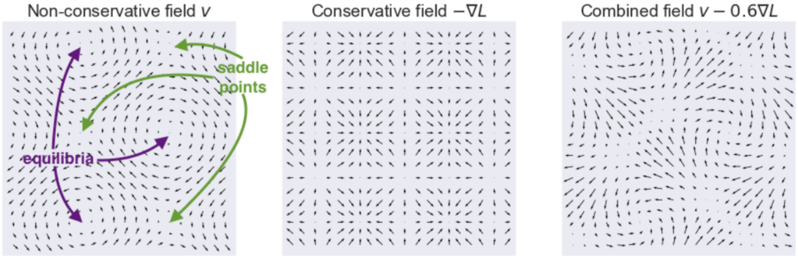

(b) Consensus optimization

Figure 2: Comparison of Simultaneous Gradient Ascent and Consensus optimization on a circular mixture of Gaussians. The images depict from left to right the resulting densities of the algorithm after 0, 5000, 10000 and 20000 iterations as well as the target density (in red).

The Numerics of GANs: Visualization



How to Train Your DRAGAN

Another approach to smoothing the function learned by GANs. Kodali et al. (2017)

- Failed GANs typically have extreme gradients/sharp peaks around real data
- Regularize GANs to reduce the gradient of the discriminator in a region around real data

How to Train Your DRAGAN: Equations

Additional regularization terms encourage discriminator to have a relatively flat surface around real data.

$$\lambda \cdot \mathbb{E}_{x \sim P_{real}, \delta \sim N_d(0,cI)} \left[\|\nabla_{\mathbf{x}} D_\theta(x+\delta)\|^2 \right]$$

$$\lambda \cdot \mathbb{E}_{x \sim P_{real}, \delta \sim N_d(0,cI)} \left[\max\left(0, \|\nabla_{\mathbf{x}} D_\theta(x+\delta)\|^2 - k\right) \right]$$

$$\lambda \cdot \mathbb{E}_{x \sim P_{real}, \delta \sim N_d(0,cI)} \left[\|\nabla_{\mathbf{x}} D_\theta(x+\delta)\| - k \right]^2$$

Progressive Growing of GANs for Improved Quality, Stability, and Variation

Very impressive results with a very simple idea. Progressively grow the generator and discriminator from low resolution to high. Similar intuition to LapGAN. Karras et al. (2017)

Progressive GANs: Architecture

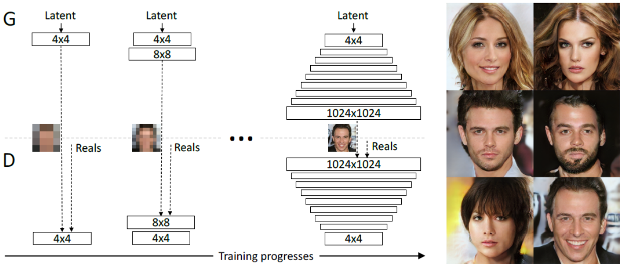


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout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

Progressive GANs: Results

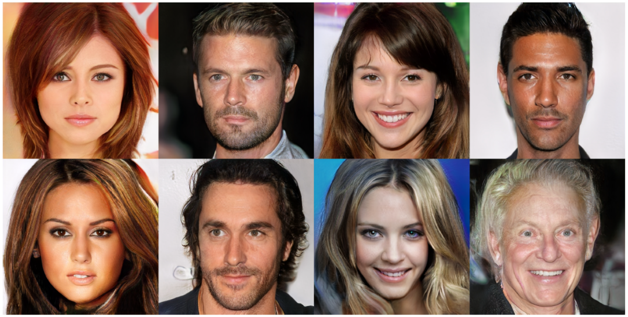


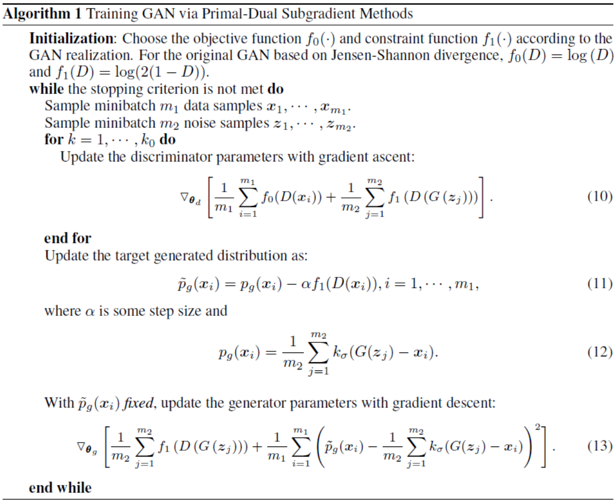
Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

https://www.youtube.com/watch?v=XOxxPcy5Gr4

Training Generative Adversarial Networks via Primal-Dual Subgradient Methods: A Lagrangian Perspective on GAN

Provides some analysis of GANs as a Lagrangian Dual. Roughly speaking, uses a kernel to encourage GAN generators to generate real samples. Chen et al. (2018)

Dual Method for GANs



Algorithm 1 Training GAN via Primal-Dual Subgradient Methods

```
**Initialization**: Choose the objective function $f_0(\cdot)$ and constraint function $f_1(\cdot)$ according to the GAN realization. For the original GAN based on Jensen-Shannon divergence, $f_0(D) = \log(D)$ and $f_1(D) = \log(2(1-D))$.
**while** the stopping criterion is not met **do**
    Sample minibatch $m_1$ data samples $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{m_1}$.
    Sample minibatch $m_2$ noise samples $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_{m_2}$.
    **for** $k = 1, \cdots, k_0$ **do**
        Update the discriminator parameters with gradient ascent:
            $\nabla_{\boldsymbol{\theta}_d} \left[ \frac{1}{m_1} \sum_{i=1}^{m_1} f_0(D(\boldsymbol{x}_i)) + \frac{1}{m_2} \sum_{j=1}^{m_2} f_1\left(D\left(G\left(\boldsymbol{z}_j\right)\right)\right) \right].$   (10)
    **end for**
    Update the target generated distribution as:
        $\tilde{p}_g(\boldsymbol{x}_i) = p_g(\boldsymbol{x}_i) - \alpha f_1(D(\boldsymbol{x}_i)), i = 1, \cdots, m_1,$   (11)
    where $\alpha$ is some step size and
        $p_g(\boldsymbol{x}_i) = \frac{1}{m_2} \sum_{j=1}^{m_2} k_\sigma(G(\boldsymbol{z}_j) - \boldsymbol{x}_i).$   (12)
    With $\tilde{p}_g(\boldsymbol{x}_i)$ *fixed*, update the generator parameters with gradient descent:
        $\nabla_{\boldsymbol{\theta}_g} \left[ \frac{1}{m_2} \sum_{j=1}^{m_2} f_1\left(D\left(G\left(\boldsymbol{z}_j\right)\right)\right) + \frac{1}{m_1} \sum_{i=1}^{m_1} \left( \tilde{p}_g(\boldsymbol{x}_i) - \frac{1}{m_2} \sum_{j=1}^{m_2} k_\sigma(G(\boldsymbol{z}_j) - \boldsymbol{x}_i) \right)^2 \right].$   (13)
**end while**
```

Assessing GAN Quality

Assessing Generative Models

In general, it is difficult to objectively assess GAN quality. There is no specific loss we are trying to optimize. Part of the reason GANs are complicated is that there is no easy way to compare distributions over images in a meaningful way.

- Most direct methods put humans in the loop. Something like mechanical Turk to identify real/fake images. Expensive and time-consuming but do provide a very convincing argument
- Some seemingly straightforward analytic methods, but with their own set of issues. For example, generate samples from the generator, make a Gaussian mixture given those points, and then calculate the test set log likelihood.
- Some methods that piggyback on existing models. For example, "inception score" is a calculation of whether an inception network gives your images a low entropy label but your images cover a variety of labels.

Even the latter two methods are relatively time-consuming to get accurate results. Typically generate in the range of 10,000 to 1,000,000 images.

Conclusion

Take-Away Messages

- GANs have a lot of potential
    - Flexible to support a variety of problems
    - Potential to represent meaningful metrics
- GAN optimization, architecture, etc. is still an active area of research
    - Many strategies for stabilizing, mostly related to simplifying the objective or regularizing the game towards a stationary point
    - Even given current knowledge, results are impressive
    - Current state-of the art changes frequently
    - Best current approach is a combination of techniques and extensive hyperparameter tuning

References

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

Dev Nag. Generative adversarial networks (gans) in 50 lines of code (pytorch). `https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-py` 2017.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014. URL `http://arxiv.org/abs/1411.1784`.

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. URL `http://arxiv.org/abs/1506.05751`.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. URL `http://arxiv.org/abs/1511.06434`.

J. T. Springenberg. Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks. *ArXiv e-prints*, November 2015.

Daniel Jiwoong Im, Chris Dongjoo Kim, Hui Jiang, and Roland Memisevic. Generating images with recurrent adversarial networks. *CoRR*, abs/1602.05110, 2016. URL `http://arxiv.org/abs/1602.05110`.

X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *ArXiv e-prints*, June 2016.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015. URL `http://arxiv.org/abs/1511.05644`.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016. URL `http://arxiv.org/abs/1605.09782`.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *CoRR*, abs/1611.02163, 2016. URL `http://arxiv.org/abs/1611.02163`.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016. URL `http://arxiv.org/abs/1606.03498`.

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. URL `http://arxiv.org/abs/1611.04076`.

Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. Energy-based generative adversarial network. *CoRR*, abs/1609.03126, 2016. URL `http://arxiv.org/abs/1609.03126`.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *ArXiv e-prints*, January 2017.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved Training of Wasserstein GANs. *ArXiv e-prints*, March 2017.

C. Kaae Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár. Amortised MAP Inference for Image Super-resolution. *ArXiv e-prints*, October 2016.

Vaishnavh Nagarajan and J. Zico Kolter. Gradient descent GAN optimization is locally stable. *CoRR*, abs/1706.04156, 2017. URL http://arxiv.org/abs/1706.04156.

Lars M. Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. *CoRR*, abs/1705.10461, 2017. URL http://arxiv.org/abs/1705.10461.

Naveen Kodali, Jacob D. Abernethy, James Hays, and Zsolt Kira. How to train your DRAGAN. *CoRR*, abs/1705.07215, 2017. URL http://arxiv.org/abs/1705.07215.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017. URL http://arxiv.org/abs/1710.10196.

X. Chen, J. Wang, and H. Ge. Training Generative Adversarial Networks via Primal-Dual Subgradient Methods: A Lagrangian Perspective on GAN. *ArXiv e-prints*, February 2018.