So You Want to Create Your Own GAN?

A Presentation For:
CSC413/2516 - Neural Networks and Deep Learning

Bolin Gao

University of Toronto
Department of Electrical and Computer Engineering
Systems Control Group

March 25, 2021

Agenda

1. Finer Points of Original GAN
2. Evaluating Your GAN
3. Evolution of GAN Architectures
    - Practice: how to do better than DCGAN?
    - Theory: how to improve the original GAN framework?
4. Evolution of GAN Dynamics
5. Where to Go From Here?

Figure: Imaginary Cards Generated by Chimera Painter by Google AI

Finer Points of Original GAN

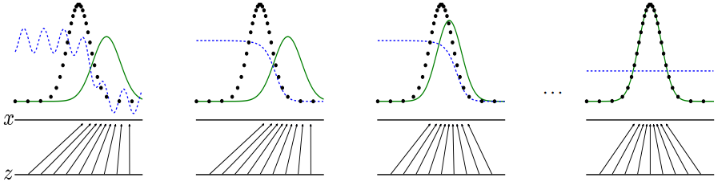


Figure: "Generative Adversarial Networks", I. Goodfellow et al., 2014

Original GAN

- Two interconnected multi-layer perceptrons (MLPs).
- **Generator**:

$$G_\theta : \mathbb{R}^m \to \mathbb{R}^n, z \mapsto G_\theta(z) \quad (1)$$

typically $m \ll n$, z is sampled from P, a Gaussian distribution (also write: $\mathbf{z} \sim P(z)$)

- **Discriminator**:

$$D_w : \mathbb{R}^n \to [0, 1], v \mapsto D_w(v) \quad (2)$$

$v \in \{x, G_\theta(z)\}$, x is a real sample from dataset $\mathcal{D}$, assumed to be sampled from (unknown) Q (also write: $\mathbf{x} \sim Q(x)$)

- Want $G_\theta(z)$ to be indistinguishable from real x for any z, z referred to as a noise/latent variable/latent code.

We assume both generator and discriminator are parametric models.
$\mathbf{z}$ is a random variable (RV), z is a realization of the RV. Same for $\mathbf{x}, x$.

Original GAN

- Two interconnected multi-layer perceptrons (MLPs).
- **Generator**:

$$G_\theta : \mathbb{R}^m \to \mathbb{R}^n, z \mapsto G_\theta(z) \tag{1}$$

typically $m \ll n$, z is sampled from P, a Gaussian distribution (also write: $\mathbf{z} \sim P(z)$)

- **Discriminator**:

$$D_w : \mathbb{R}^n \to [0, 1], v \mapsto D_w(v) \tag{2}$$

$v \in \{x, G_\theta(z)\}$, x is a real sample from dataset $\mathcal{D}$, assumed to be sampled from (unknown) Q (also write: $\mathbf{x} \sim Q(x)$)

- Want $G_\theta(z)$ to be indistinguishable from real x for any z, z referred to as a noise/latent variable/latent code.

We assume both generator and discriminator are parametric models.
$\mathbf{z}$ is a random variable (RV), z is a realization of the RV. Same for $\mathbf{x}, x$.

Original GAN

- Two interconnected multi-layer perceptrons (MLPs).
- **Generator**:

$$G_\theta : \mathbb{R}^m \to \mathbb{R}^n, z \mapsto G_\theta(z) \tag{1}$$

typically $m \ll n$, z is sampled from P, a Gaussian distribution (also write: $\mathbf{z} \sim P(z)$)

- **Discriminator**:

$$D_w : \mathbb{R}^n \to [0,1], v \mapsto D_w(v) \tag{2}$$

$v \in \{x, G_\theta(z)\}$, x is a real sample from dataset $\mathcal{D}$, assumed to be sampled from (unknown) Q (also write: $\mathbf{x} \sim Q(x)$)

- Want $G_\theta(z)$ to be indistinguishable from real x for any z, z referred to as a noise/latent variable/latent code.

We assume both generator and discriminator are parametric models.
$\mathbf{z}$ is a random variable (RV), z is a realization of the RV. Same for $\mathbf{x}, x$.

Original GAN

- Two interconnected multi-layer perceptrons (MLPs).
- **Generator**:

$$G_\theta : \mathbb{R}^m \to \mathbb{R}^n, z \mapsto G_\theta(z) \tag{1}$$

  typically $m \ll n$, z is sampled from P, a Gaussian distribution (also write: $\mathbf{z} \sim P(z)$)

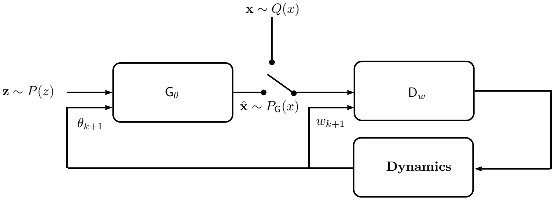
- **Discriminator**:

$$D_w : \mathbb{R}^n \to [0, 1], v \mapsto D_w(v) \tag{2}$$

  $v \in \{x, G_\theta(z)\}$, x is a real sample from dataset $\mathcal{D}$, assumed to be sampled from (unknown) Q (also write: $\mathbf{x} \sim Q(x)$)

- Want $G_\theta(z)$ to be indistinguishable from real x for any z, z referred to as a noise/latent variable/latent code.

We assume both generator and discriminator are parametric models.
$\mathbf{z}$ is a random variable (RV), z is a realization of the RV. Same for $\mathbf{x}, x$.



We wish to find θ, w of G_θ and D_w, by solving,

$$\min_\theta \max_w \mathbb{E}_{\mathbf{x}\sim Q(x)}[\log(D_w(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D_w(G_\theta(\mathbf{z})))] \quad (3)$$

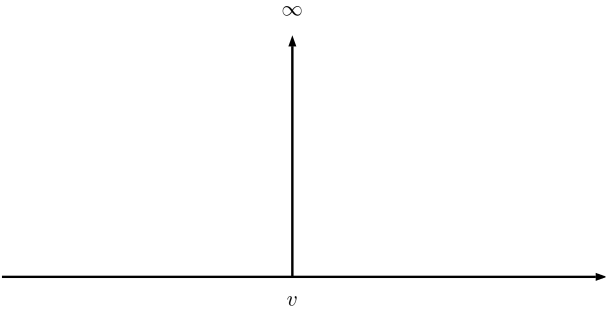
Sometimes write $\hat{\mathbf{x}} = G_\theta(\mathbf{z})$, has induced distribution $\hat{\mathbf{x}} \sim P_G(x)$.

We refer to the saddle function,

$$\mathcal{L}(\theta, w) = \mathbb{E}_{\mathbf{x}\sim Q(x)}[\log(D_w(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D_w(G_\theta(\mathbf{z})))] \quad (4)$$

as the (original) **GAN objective**.

A Simple Example: "Dirac GAN"

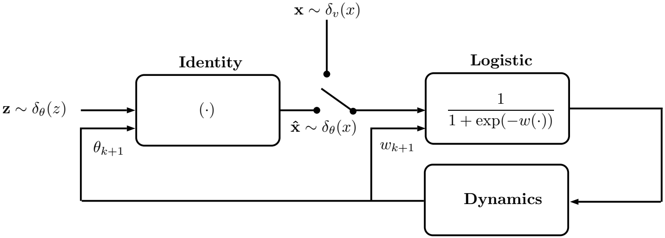


Shifted Dirac delta function:

$$\delta_v(x) = \begin{cases} +\infty & x = v \\ 0 & \text{else where} \end{cases} \tag{5}$$

Sifting property:

$$\int_{\mathbb{R}} \delta_v(x) f(x) \mathrm{d}x = f(v) \tag{6}$$



$$D_w(\mathbf{x}) = \frac{1}{1+\exp(-w\mathbf{x})}, \qquad \mathbf{x} \sim \delta_v(x)$$

$$G_\theta(\mathbf{z}) = \mathbf{z}, \qquad \mathbf{z} \sim \delta_\theta(z)$$

$$D_w(\mathbf{x}) = \frac{1}{1+\exp(-w\mathbf{x})}, \qquad \mathbf{x} \sim \delta_v(x)$$

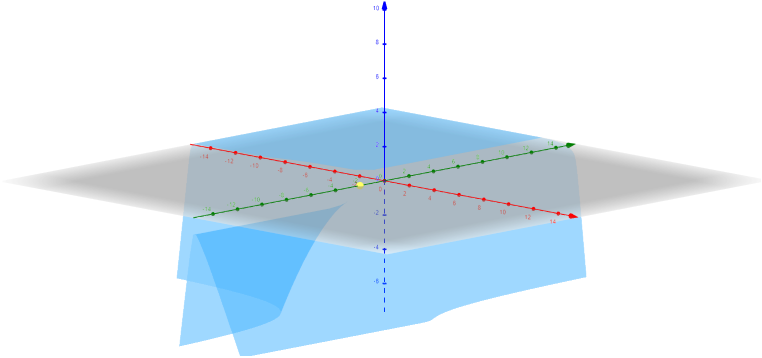
$$G_\theta(\mathbf{z}) = \mathbf{z}, \qquad \mathbf{z} \sim \delta_\theta(z)$$

$$\begin{aligned}
&\min_{\theta\in\mathbb{R}} \max_{w\in\mathbb{R}} \quad \mathbb{E}_{\mathbf{x}\sim\delta_v(x)}[\log(D_w(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim\delta_\theta(z)}[\log(1-D_w(G_\theta(\mathbf{z})))] \\
&= \min_{\theta\in\mathbb{R}} \max_{w\in\mathbb{R}} \int_{\mathbb{R}} \log(D_w(x))\delta_v(x)\mathrm{d}x + \int_{\mathbb{R}} \log(1-D_w(G_\theta(z)))\delta_\theta(z)\mathrm{d}z \\
&= \min_{\theta\in\mathbb{R}} \max_{w\in\mathbb{R}} \quad \log(\frac{1}{1+\exp(-wv)}) + \log(1-\frac{1}{1+\exp(-w\theta)}) \\
&= \min_{\theta\in\mathbb{R}} \max_{w\in\mathbb{R}} \quad \log(\frac{1}{1+\exp(-wv)}) + \log(\frac{1}{1+\exp(w\theta)}) \\
&= \min_{\theta\in\mathbb{R}} \max_{w\in\mathbb{R}} \quad -\log(1+\exp(-wv)) - \log(1+\exp(w\theta))
\end{aligned}$$

(Exercise: show this objective is concave in θ and concave in w)

Hard: because we want to minimize a concave (non-convex) function.

Let's plot $-\log(1 + \exp(-wv)) - \log(1 + \exp(w\theta)), v = -2$



Let $v = -2$, then it can be shown that, $(\theta^\star, w^\star) = (-2, 0)$.

Observe, $D^\star(v) = \left.\dfrac{1}{1 + \exp(-w(-2))}\right|_{w=w^\star} = \dfrac{1}{1 + \exp(0)} = \dfrac{1}{2}$

In general, the optimal discriminator $D^\star$ of original GAN is[1],

$$D^\star(\mathbf{x}) = \frac{Q(\mathbf{x})}{Q(\mathbf{x}) + P_G(\mathbf{x})}, Q(\mathbf{x}) = P_G(\mathbf{x})$$

Let $w^\star$ be optimal weight, then $\mathcal{L}(\theta, w^\star)$ is,

$$\begin{aligned}
&= \mathbb{E}_{\mathbf{x}\sim Q}[\log(D^\star(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D^\star(G_\theta(\mathbf{z})))] \\
&= \mathbb{E}_{\mathbf{x}\sim Q}\left[\log(D^\star(\mathbf{x}))\right] + \mathbb{E}_{\mathbf{z}\sim P}\left[\log(1 - \frac{Q(G_\theta(\mathbf{z}))}{Q(G_\theta(\mathbf{z})) + P_G(G_\theta(\mathbf{z}))})\right] \\
&= \mathbb{E}_{\mathbf{x}\sim Q}\left[\log(D^\star(\mathbf{x}))\right] + \mathbb{E}_{\mathbf{z}\sim P}\left[\log(\frac{P_G(G_\theta(\mathbf{z}))}{Q(G_\theta(\mathbf{z})) + P_G(G_\theta(\mathbf{z}))})\right] \\
&= \mathbb{E}_{\mathbf{x}\sim Q}\left[\log(\frac{Q(\mathbf{x})}{Q(\mathbf{x}) + P_G(\mathbf{x})})\right] + \mathbb{E}_{\hat{\mathbf{x}}\sim P_G}\left[\log(\frac{P_G(\mathbf{x})}{Q(\mathbf{x}) + P_G(\mathbf{x})})\right] \\
&= \int_{\mathbb{R}} Q(x)\log(\frac{Q(x)}{Q(x) + P_G(x)})\mathrm{d}x + \int_{\mathbb{R}} P_G(x)\log(\frac{P_G(x)}{Q(x) + P_G(x)})\mathrm{d}x
\end{aligned}$$

[1] "Generative Adversarial Nets", Goodfellow et al., 2014

$$\begin{aligned}\mathcal{L}(\theta, w^\star) &= D_{KL}(Q, Q + P_G) + D_{KL}(P_G, Q + PG) \\ &= -\log(4) + D_{KL}(Q, \frac{Q + P_G}{2}) + D_{KL}(P_G, \frac{Q + PG}{2}) \\ &= -\log(4) + 2D_{JS}(Q, P_G)\end{aligned}$$

- $D_{JS}(Q, P_G) := \frac{1}{2}(D_{KL}(Q, \frac{Q+P_G}{2}) + D_{KL}(P_G, \frac{Q+P_G}{2}))$ is the Jensen-Shannon divergence
- $D_{KL}(Q, P) := \int Q(x) \log(Q(x)/P(x)) \mathrm{d}x$ is the (continuous) Kullback-Leibler divergence.

When $D^\star$ is optimal, the GAN problem reduces to minimizing $D_{JS}(Q, P_G)$. The optimal value is $\mathcal{L}(\theta^\star, w^\star) = -\log(4)$.

Extremely important insight - spark of many research progress.

"Discriminator/Generator Loss"

Assume both G and D wish to **minimize** their loss. The discriminator and generator losses ("cost functions") are:

$$\begin{aligned}\mathcal{L}_D(w;\theta) &:= -\mathcal{L}(\theta, w)\\ &= -\mathbb{E}_{\mathbf{x}\sim Q(x)}[\log(D_w(\mathbf{x}))] - \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D_w(G_\theta(\mathbf{z})))]\end{aligned}$$

$$\begin{aligned}\mathcal{L}_G(\theta;w) &:= \mathcal{L}(\theta, w)\\ &= \mathbb{E}_{\mathbf{x}\sim Q(x)}[\log(D_w(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D_w(G_\theta(\mathbf{z})))]\end{aligned}$$

This is a two-player **zero-sum game**: $\mathcal{L}_D(w;\theta) + \mathcal{L}_G(\theta;w) = 0$

"Discriminator/Generator Loss"

Assume both G and D wish to **minimize** their loss. The discriminator and generator losses ("cost functions") are:

$$\begin{aligned}\mathcal{L}_D(w;\theta) &:= -\mathcal{L}(\theta, w) \\ &= -\mathbb{E}_{\mathbf{x}\sim Q(x)}[\log(D_w(\mathbf{x}))] - \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D_w(G_\theta(\mathbf{z})))]\end{aligned}$$

$$\begin{aligned}\mathcal{L}_G(\theta; w) &:= \mathcal{L}(\theta, w) \\ &= \mathbb{E}_{\mathbf{x}\sim Q(x)}[\log(D_w(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D_w(G_\theta(\mathbf{z})))]\end{aligned}$$

This is a two-player **zero-sum game**: $\mathcal{L}_D(w;\theta) + \mathcal{L}_G(\theta; w) = 0$

Note that GAN folks go a step further and removes the $\mathbb{E}_{\mathbf{x}\sim Q(x)}[\log(D_w(\mathbf{x}))]$ term from $\mathcal{L}_G$ (no θ dependence),

$$\mathcal{L}_G(\theta; w) := \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D_w(G_\theta(\mathbf{z})))] \tag{7}$$

Min-Max vs Non-Saturating

$$\mathcal{L}_D(w;\theta) = -\mathbb{E}_{\mathbf{x}\sim Q(x)}[\log(D_w(\mathbf{x}))] - \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D_w(G_\theta(\mathbf{z})))]$$
$$\mathcal{L}_G(\theta;w) = \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D_w(G_\theta(\mathbf{z})))]$$

We call this the **Min-Max GAN**.

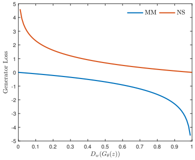
Min-Max vs Non-Saturating

$$\mathcal{L}_D(w;\theta) = -\mathbb{E}_{\mathbf{x}\sim Q(x)}[\log(D_w(\mathbf{x}))] - \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D_w(G_\theta(\mathbf{z})))]$$
$$\mathcal{L}_G(\theta;w) = \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D_w(G_\theta(\mathbf{z})))]$$

We call this the **Min-Max GAN**.

But, what was actually implemented is **Non-Saturating GAN**

$$\mathcal{L}_D(w;\theta) = -\mathbb{E}_{\mathbf{x}\sim Q(x)}[\log(D_w(\mathbf{x}))] - \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D_w(G_\theta(\mathbf{z})))]$$
$$\mathcal{L}_G(\theta;w) = -\mathbb{E}_{\mathbf{z}\sim P(z)}[\log(D_w(G_\theta(\mathbf{z})))]$$



This leads to quite different learning dynamics

Min-Max GAN

$$w_{k+1} = w_k + \frac{1}{B} \sum_{i=1}^{B} \nabla_w \log(D_w(x^{(i)})) + \log(1 - D_w(G_\theta(z^{(i)})))$$

$$\theta_{k+1} = \theta_k - \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \log(1 - D_w(G_\theta(z^{(i)}))), \ z^{(i)} \text{ resampled}$$

Non-Saturating GAN

$$w_{k+1} = w_k + \frac{1}{B} \sum_{i=1}^{B} \nabla_w \log(D_w(x^{(i)})) + \log(1 - D_w(G_\theta(z^{(i)})))$$

$$\theta_{k+1} = \theta_k + \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \log(D_w(G_\theta(z^{(i)}))), \ z^{(i)} \text{ resampled}$$

w is sometimes updated several times before θ is updated ("multi-looping").

In practice, $\mathcal{L}_D(w;\theta), \mathcal{L}_G(\theta;w)$ are built by using the binary cross entropy with logits loss (BCE-LL) with labels $\mathbf{0}$ (fake) or $\mathbf{1}$ (real).

In practice, $\mathcal{L}_D(w;\theta), \mathcal{L}_G(\theta;w)$ are built by using the binary cross entropy with logits loss (BCE-LL) with labels $\mathbf{0}$ (fake) or $\mathbf{1}$ (real).

Recall given two vectors $x = (x_n), y = (y_n) \in \mathbb{R}^n$, BCE-LL is,

$$\mathcal{H}(x,y) = -[y_n \log(\theta(x_n)) + (1 - y_n)\log(1 - \theta(x_n))] \qquad (8)$$

In practice, $\mathcal{L}_D(w;\theta), \mathcal{L}_G(\theta;w)$ are built by using the binary cross entropy with logits loss (BCE-LL) with labels $\mathbf{0}$ (fake) or $\mathbf{1}$ (real).

Recall given two vectors $x = (x_n), y = (y_n) \in \mathbb{R}^n$, BCE-LL is,

$$\mathcal{H}(x, y) = -[y_n \log(\theta(x_n)) + (1 - y_n)\log(1 - \theta(x_n))] \qquad (8)$$

Also, let the discriminator be decomposed as $D_w(x) = \theta(h_w(x))$.

$$\begin{aligned}
\mathcal{L}_D(w;\theta) &\approx \frac{1}{B}\sum_{i=1}^{B} \mathcal{H}(\mathbf{1}, h_w(x^{(i)})) + \mathcal{H}(\mathbf{0}, h_w(G_\theta(z^{(i)}))) \\
&= \frac{1}{B}\textstyle\sum_{i=1}^{B} -\log(\theta(h_w(x^{(i)}))) - \log(1 - \theta(h_w(G_\theta(z^{(i)})))) \\
&= \frac{1}{B}\textstyle\sum_{i=1}^{B} -\log(D_w(x^{(i)})) - \log(1 - D_w(G_\theta(z^{(i)})))
\end{aligned}$$

$$\mathcal{L}_G(\theta;w) \approx \frac{1}{B}\textstyle\sum_{i=1}^{B} \mathcal{H}(\mathbf{1}, \theta(h_w(G_\theta(z^{(i)})))) = \frac{1}{B}\textstyle\sum_{i=1}^{B} -\log(D_w(G_\theta(z^{(i)}))$$

Evaluating Your GAN

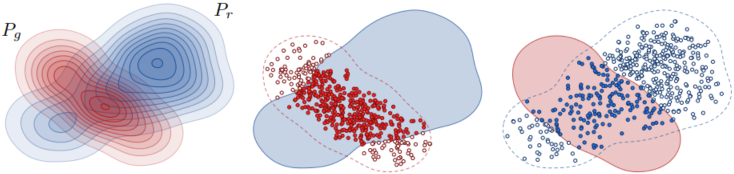


Figure: "Improved Precision and Recall Metric for Assessing Generative Models", T. Karras et al., 2019

Inception Score

- a metric that shown to correlate well with human scoring of the realism of generated images from the CIFAR-10 dataset.

  Generate a batch of images $\{\hat{x}^{(i)}\}_{i=1}^{B}$, run through Inception V3[2] to get $\{p(y|\hat{x}^{(i)})\}_{i=1}^{B}$, $p(y|\hat{x}^{(i)}) \in [0,1]^{1000}$. Calculate the

  inception score (IS) as,

  $$\mathrm{IS} = \exp\left[\frac{1}{B}\sum_{i=1}^{B} D_{\mathrm{KL}}(p(y|\hat{x}^{(i)}), \frac{1}{B}\sum_{i=1}^{B} p(y|\hat{x}^{(i)}))\right] \in [1, 1000]$$

  where $D_{KL}(v,w) = \sum_{n=1}^{N} v_n \log(v_n/w_n)$ is the “discrete” KL divergence.

- The higher the better.

---

[2] “Going Deeper with Convolutions”, Szegedy et al. 2015

The IS formula is taken from “A Note on the Inception Score”, Barratt and Sharma, 2018.

“Improved Techniques for Training GANs”, Salimans et al. 2016

Why Inception?

Inception V3 is trained on very a large set of (1000 classes) images, therefore, it should be able to tell if,

- there exists a single object in the image via the softmax probabilities
- the generator is able to generate a wide variety of images.

A higher score intuitively corresponds to achieving these objectives.

Figure: Samples generated by BigGAN at 512×512 resolution

$$\text{IS} = 241.5$$

$$\text{IS} = 900.15$$

Recent papers[3] have pointed out some problems with using IS

[3] Figure: "A Note on the Inception Score", Barratt and Sharma, 2018.

Frechet Inception Distance

- Draw a batch of real $\{x^{(i)}\}$ and generated images $\{\hat{x}^{(i)}\}$.
- Embed $\{x^{(i)}\}$ and $\{\hat{x}^{(i)}\}$ by using some specific layer of the Inception V3 (e.g., until after the last activation layer), then take statistics of each batch.

FID is the "Wasserstein-2 distance" between two multivariate Gaussians, $\mathcal{N}(\mu_1, \Sigma_1)$ and $\mathcal{N}(\mu_2, \Sigma_2)$.

$$\text{FID} = \|\mu_1 - \mu_2\|_2^2 + \text{tr}(\Sigma_1 + \Sigma_2 - 2\sqrt{\Sigma_1 \Sigma_2})$$

For 1D Gaussians, $\text{FID} = (\mu_1 - \mu_2)^2 + (\sigma_X^2 + \sigma_Y^2 - 2\sigma_X \sigma_Y)$.

Obviously closer to zero the better.

- Compares the fake vs real images based on their distributions (instead just looking at fake images as with IS).

"GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium", M. Heusel et al., 2018

FID Is Not Perfect Either

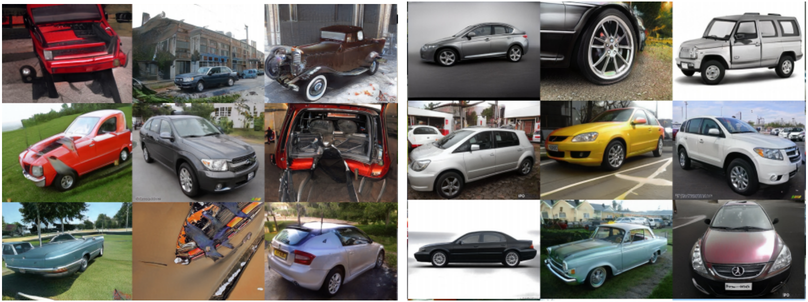


Figure: Both set of images have FID 3.27

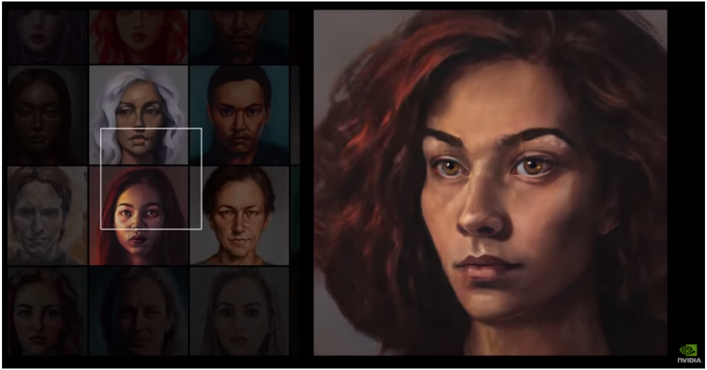
Other metrics you might want to try:

- Precision and Recall: kNN based approach
- Sliced Wasserstein Distance (lower = better)
- Perceptual path length scores (lower = better)

See "Pros and Cons of GAN Evaluation Measures", Borji, 2018, for a survey of metrics

Evolution of GAN Architectures

Part I: Practice



https://www.youtube.com/watch?v=9QuDh3W3lOY

DCGAN (Deep Convolutional GAN)[4]

- The original GAN only used MLPs. DCGAN is one of the first GANs utilizing CNN that worked well.
- List of best practices circa 2015,
  1. Replace pooling layers with strided convolutions (D_w) and fractional-strided convolutions (G_θ), so no pooling layers.
  2. Use batchnorm in both G_θ and D_w.
  3. Remove FC hidden layers for deeper architectures.
  4. ReLU in G_θ except for the output, which uses Tanh.
  5. LeakyReLU in D_w for all layers (except the output) and no activation at the output
- Essential idea is to avoid "sparse" gradients.
- In practical implementations, people relax one of more of the above (e.g., use regular convolution)

[4] "Unsupervised representation learning with deep convolutional generative adversarial networks", A. Radford, L. Metz, and S. Chintala, 2015

Figure: IS 6.69, FID 35.6

DCGAN performed well on small images, but how to generate larger more complicated images?

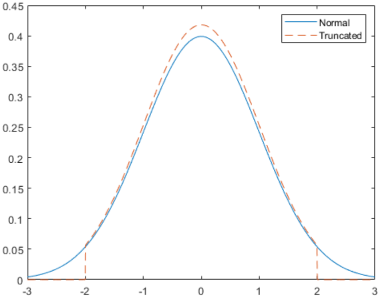
BigGAN

- A famous GAN that generated really good quality images

- List of best practices circa 2018,
  1. Use ResNet for both G_θ and D_w.
  2. Skip connections from latent to intermediate layers of G_θ ("skip-z")
  3. 2 D_w updates per G_θ update (Double-looping)
  4. Moving average for weights
  5. Orthogonal regularization or pairwise cosine similarity
  6. Non-local blocks
- Essential idea is to capture long-term dependencies.

"Large scale Gan Training For High Fidelity Natural Image Synthesis", A. Brock et al., 2018.

- BigGAN had an interesting idea called "truncation trick"
- Extremely simple idea: after GAN is trained, sample images from noise that are close to the mean of the distribution.



- This ensures generation from distribution that the generator performs good on.
- But this also means you have less variety in the generated images.

BigGAN produced nice images, but it cannot tackle the *unforgiving* task of human face generation

Figure: IS 232.5, FID 8.1

Your brain has a dedicated region for facial recognition, the fusiform gyrus. You are trying to fool millions of years of evolution.

Progressive Growing of GANs

Here is an idea: instead of learning all aspects of facial features at once, learn from coarse (4×4) to fine (1024×1024).

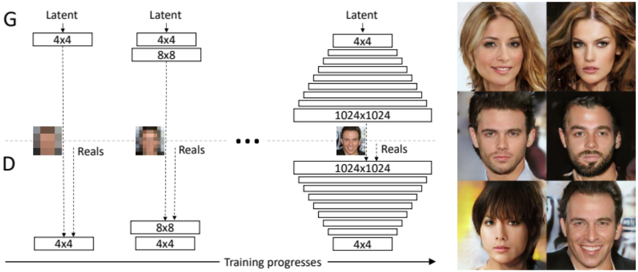
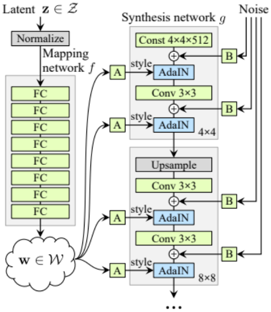


Figure: FID 7.79 on CelebA-HQ, 8.04 FFHQ

"Progressive Growing of GANs for Improved Quality, Stability and Variation", Karras, Aila, Laine, Lehtinen, 2018

StyleGAN

StyleGAN takes the progressive growing idea further through a *Style-based generator*.



Note that $4 \times 4 \times 512$ is a constant tensor of ones[5]. Not updated during training. It is like a growing canvas which we paint on.

[5] https://github.com/NVlabs/stylegan/blob/master/training/networks_stylegan.py line 507

"A Style-Based Generator Architecture for Generative Adversarial Networks", Karras et al, 2018.

Also uses truncation trick similar to BigGAN, but in $\mathcal{W}$ space:

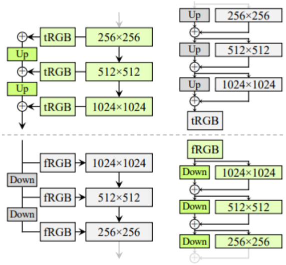
1. Calculate "average face" $\overline{\mathrm{w}} = \mathbb{E}_{\mathbf{z}\sim P(z)}[f(z)]$, f is the mapping network.
2. Scale deviation from average face $\widetilde{\mathrm{w}} = \overline{\mathrm{w}} + \psi(w - \overline{\mathrm{w}})$, where ψ is the truncation variable (default setting $\psi = 0.7$).

Similar effect as BigGAN, avoids generation from latent variables that were unseen (rarely seen) by generator.

Can You Spot the Artifacts?

StyleGAN2

- Authors found above artifacts (in particular, the "blobs") are due to the AdaIN and progressive growing architecture
- Added weight demodulation layer in place of AdaIN
- Added a generator with skip connection and a discriminator with residual blocks place of progressive training.



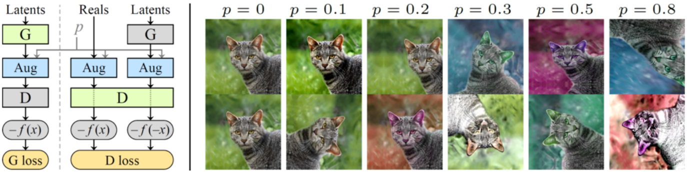
(b) Input/output skips (c) Residual nets

FFHQ	D original FID	D original PPL	D input skips FID	D input skips PPL	D residual FID	D residual PPL
G original	4.32	265	4.18	235	3.58	269
G output skips	4.33	169	3.77	127	**3.31**	**125**
G residual	4.35	203	3.96	229	3.79	243

LSUN Car	D original FID	D original PPL	D input skips FID	D input skips PPL	D residual FID	D residual PPL
G original	3.75	905	3.23	758	3.25	802
G output skips	3.77	544	3.86	**316**	3.19	471
G residual	3.93	981	3.40	667	**2.66**	645

"Analyzing and Improving the Image Quality of StyleGAN", T. Karras, 2020

StyleGAN2-Ada



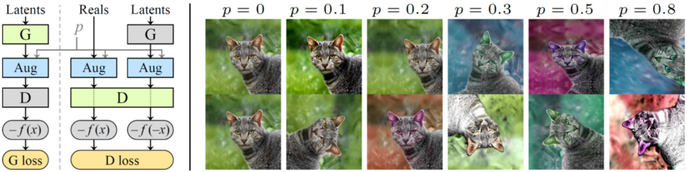
- StyleGAN requires order $\mathcal{O}(10^5) - \mathcal{O}(10^6)$ images to train.
- Data augmentation techniques (flipping, cropping, rotation, noise) prone to "leaking": generator will generate augmented images as well.
- Idea: Augment the real images AND the fake images.
  Let $\mathcal{T}$ be an augmentation, $\mathbf{x}$ real, $\hat{\mathbf{x}}$ fake, then, train so that

$$\mathcal{T}(\mathbf{x}) = \mathcal{T}(\hat{\mathbf{x}}) \tag{9}$$

  Suppose that $\mathcal{T}$ is invertible, then $\mathbf{x} = \hat{\mathbf{x}}$.

"Training Generative Adversarial Networks with Limited Data", T. Karras, et al. 2020

StyleGAN2-Ada



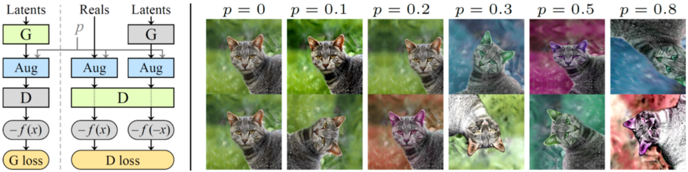
- StyleGAN requires order $\mathcal{O}(10^5) - \mathcal{O}(10^6)$ images to train.
- Data augmentation techniques (flipping, cropping, rotation, noise) prone to "leaking": generator will generate augmented images as well.
- Idea: Augment the real images AND the fake images.
  Let $\mathcal{T}$ be an augmentation, $\mathbf{x}$ real, $\hat{\mathbf{x}}$ fake, then, train so that

$$\mathcal{T}(\mathbf{x}) = \mathcal{T}(\hat{\mathbf{x}}) \tag{9}$$

  Suppose that $\mathcal{T}$ is invertible, then $\mathbf{x} = \hat{\mathbf{x}}$.

"Training Generative Adversarial Networks with Limited Data", T. Karras, et al. 2020

StyleGAN2-Ada

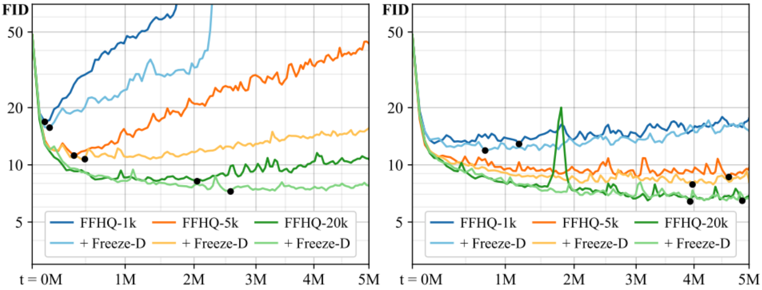


- StyleGAN requires order $\mathcal{O}(10^5) - \mathcal{O}(10^6)$ images to train.
- Data augmentation techniques (flipping, cropping, rotation, noise) prone to "leaking": generator will generate augmented images as well.
- Idea: Augment the real images AND the fake images.
  Let $\mathcal{T}$ be an augmentation, $\mathbf{x}$ real, $\hat{\mathbf{x}}$ fake, then, train so that

$$\mathcal{T}(\mathbf{x}) = \mathcal{T}(\hat{\mathbf{x}}) \tag{9}$$

  Suppose that $\mathcal{T}$ is invertible, then $\mathbf{x} = \hat{\mathbf{x}}$.

"Training Generative Adversarial Networks with Limited Data", T. Karras, et al. 2020



Solves two challenges:

- $\mathcal{T}$ non-invertible in general; can be made "invertible" through selective skipping, which is controlled by some (probability) $p \in [0, 1]$
- Tuning of p is not straightforward, instead make p adaptive (this is the "ada" part) based on the performance of D_w. If it is performing too well, make p larger, otherwise, p smaller.

Details are a bit technical, hear it directly from the authors:
`https://www.youtube.com/watch?v=h0x9NBwDkHY`

Evolution of GAN Architectures

Part II: Theory

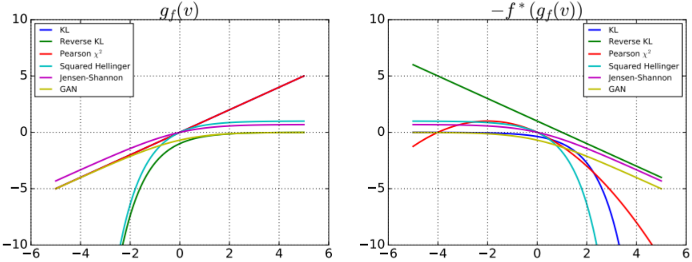
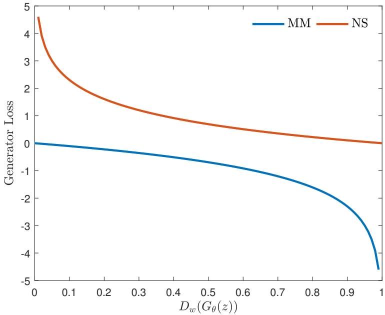


Figure: "f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization", S. Nowozin et al., 2016

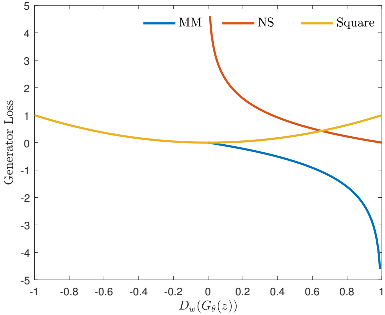
From your lecture, MM-GAN loss function is prone to vanishing gradients. NS-GAN fixes this issue a bit.



Observe the loss saturates when $D_w(G_\theta(z))) \to 1$, hence providing no feedback to account for correctly classified generated images that may look different than the real images.

Early authors sought to combat this saturation issue with new objective functions.

Least Squares GAN



$$\min_{w} \mathcal{L}_{D_w}(w;\theta) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim Q(x)}[(D_w(\mathbf{x})-1)^2] + \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim P(z)}[(D_w(G_\theta(\mathbf{z})))^2]$$

$$\min_{\theta} \mathcal{L}_{G_\theta}(\theta;w) = \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim P(z)}[(D_w(G_\theta(\mathbf{z}))-1)^2]$$

Penalize correctly generated samples if it doesn't quite look similar to the real images.

"On the Effectiveness of Least Squares Generative Adversarial Networks", X. Mao et al, 2018

LSGAN has an additional insight: at optimal $D^\star$ it is minimizing the Pearson Chi-square divergence (as opposed to the Jensen Shannon Divergence for the original GAN),

$$\begin{aligned}\mathcal{L}(\theta, w^\star) &= \frac{1}{2}\int_{\mathbb{R}} \frac{(2P_G(x) - (Q(x) + P_G(x)))^2}{P_G(x) + Q(x)}\mathrm{d}x \\ &= \frac{1}{2}\chi^2(P_G(x) + Q(x), 2P_G(x))\end{aligned}$$

What if we put divergence minimization front and center in our GAN design?

WGAN (Wasserstein GAN)

- Directly considers minimization of the Wasserstein-1 distance,

$$D(Q, P_G) = \min_{\gamma \in \Gamma} \mathbb{E}[\|\mathbf{x} - \hat{\mathbf{x}}\|], \mathbf{x} \sim Q(x), \hat{\mathbf{x}} \sim P_G(x) \qquad (10)$$

where Γ is the set of all joint distributions whose marginals are Q, P_G. This is also known as Earth Mover distance.

- By the so-called Kantorovich-Rubinstein duality,

$$D(Q, P_G) = \max_{w} \mathbb{E}_{\mathbf{x} \sim Q(x)}[D_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim P(z)}[D_w(G_\theta(\mathbf{z}))] \quad (11)$$

where the “critic” D_w is any 1-Lipschitz function.[6]

- Minimizing over θ yields the WGAN objective,

$$\mathcal{L}(\theta, w) = \min_{\theta} \max_{w} \mathbb{E}_{\mathbf{x} \sim Q(x)}[D_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim P(z)}[D_w(G_\theta(\mathbf{z}))] \quad (12)$$

- Also attempts to avoids the vanishing gradient problem.

[6] It is called a critic as opposed to a discriminator because it does not output $\{0, 1\}$.

“Wasserstein GAN”, M. Arjovsky, S. Chintala, L. Bottou, 2017. For more details about derivations: see https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

WGAN (Wasserstein GAN)

- Directly considers minimization of the Wasserstein-1 distance,

$$D(Q, P_G) = \min_{\gamma \in \Gamma} \mathbb{E}[\|\mathbf{x} - \hat{\mathbf{x}}\|], \mathbf{x} \sim Q(x), \hat{\mathbf{x}} \sim P_G(x) \quad (10)$$

  where Γ is the set of all joint distributions whose marginals are Q, P_G. This is also known as Earth Mover distance.
- By the so-called Kantorovich-Rubinstein duality,

$$D(Q, P_G) = \max_{w} \mathbb{E}_{\mathbf{x} \sim Q(x)}[D_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim P(z)}[D_w(G_\theta(\mathbf{z}))] \quad (11)$$

  where the "critic" D_w is any 1-Lipschitz function.[6]
- Minimizing over θ yields the WGAN objective,

$$\mathcal{L}(\theta, w) = \min_{\theta} \max_{w} \mathbb{E}_{\mathbf{x} \sim Q(x)}[D_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim P(z)}[D_w(G_\theta(\mathbf{z}))] \quad (12)$$

- Also attempts to avoids the vanishing gradient problem.

[6] It is called a critic as opposed to a discriminator because it does not output $\{0, 1\}$.

"Wasserstein GAN", M. Arjovsky, S. Chintala, L. Bottou, 2017. For more details about derivations: see https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

WGAN (Wasserstein GAN)

- Directly considers minimization of the Wasserstein-1 distance,

$$D(Q, P_G) = \min_{\gamma \in \Gamma} \mathbb{E}[\|\mathbf{x} - \hat{\mathbf{x}}\|], \mathbf{x} \sim Q(x), \hat{\mathbf{x}} \sim P_G(x) \qquad (10)$$

  where Γ is the set of all joint distributions whose marginals are Q, P_G. This is also known as Earth Mover distance.

- By the so-called Kantorovich-Rubinstein duality,

$$D(Q, P_G) = \max_{w} \mathbb{E}_{\mathbf{x} \sim Q(x)}[D_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim P(z)}[D_w(G_\theta(\mathbf{z}))] \quad (11)$$

  where the "critic" D_w is any 1-Lipschitz function.[6]

- Minimizing over θ yields the WGAN objective,

$$\mathcal{L}(\theta, w) = \min_{\theta} \max_{w} \mathbb{E}_{\mathbf{x} \sim Q(x)}[D_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim P(z)}[D_w(G_\theta(\mathbf{z}))] \quad (12)$$

- Also attempts to avoids the vanishing gradient problem.

[6] It is called a critic as opposed to a discriminator because it does not output $\{0, 1\}$.

"Wasserstein GAN", M. Arjovsky, S. Chintala, L. Bottou, 2017. For more details about derivations: see https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

WGAN (Wasserstein GAN)

- Directly considers minimization of the Wasserstein-1 distance,

$$D(Q, P_G) = \min_{\gamma \in \Gamma} \mathbb{E}[\|\mathbf{x} - \hat{\mathbf{x}}\|], \mathbf{x} \sim Q(x), \hat{\mathbf{x}} \sim P_G(x) \qquad (10)$$

  where Γ is the set of all joint distributions whose marginals are Q, P_G. This is also known as Earth Mover distance.

- By the so-called Kantorovich-Rubinstein duality,

$$D(Q, P_G) = \max_{w} \mathbb{E}_{\mathbf{x} \sim Q(x)}[D_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim P(z)}[D_w(G_\theta(\mathbf{z}))] \quad (11)$$

  where the "critic" D_w is any 1-Lipschitz function.[6]

- Minimizing over θ yields the WGAN objective,

$$\mathcal{L}(\theta, w) = \min_{\theta} \max_{w} \mathbb{E}_{\mathbf{x} \sim Q(x)}[D_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim P(z)}[D_w(G_\theta(\mathbf{z}))] \quad (12)$$

- Also attempts to avoids the vanishing gradient problem.

[6] It is called a critic as opposed to a discriminator because it does not output $\{0, 1\}$.

"Wasserstein GAN", M. Arjovsky, S. Chintala, L. Bottou, 2017. For more details about derivations: see `https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html`

- WGAN became a favorite among people who study games (e.g., me) because it leads to very tractable toy examples.

$$\min_{\theta} \max_{w} \mathbb{E}_{\mathbf{x} \sim Q(x)}[D_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim P(z)}[D_w(G_\theta(\mathbf{z}))] \tag{13}$$

Exercise: show D_w is 1-Lipschitz for $\|w\|_2 \leq 1$

This example was studied by "Training GANs with Optimism" by C. Daskalakis et al., 2018 in the WGAN context

- WGAN became a favorite among people who study games (e.g., me) because it leads to very tractable toy examples.

$$\min_\theta \max_w \mathbb{E}_{\mathbf{x}\sim Q(x)}[D_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim P(z)}[D_w(G_\theta(\mathbf{z}))] \tag{13}$$

Example

$$D_w(\mathbf{x}) = w^\top \mathbf{x}, \quad \mathbf{x} \sim \mathcal{N}(\mu, \Sigma)$$

$$G_\theta(\mathbf{z}) = \mathbf{z} - \theta, \quad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$$

$$\begin{aligned}\mathbb{E}_{\mathbf{x}}(D_w(\mathbf{x})) - \mathbb{E}_{\mathbf{z}}(D_w(G_\theta(\mathbf{z}))) &= w^\top \mathbb{E}_{\mathbf{x}\sim\mathcal{N}(\mu,\Sigma)}(\mathbf{x}) - \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}(w(\mathbf{z}-\theta)) \\ &= w(\mu+\theta)\end{aligned}$$

$$\implies \min_\theta \max_w w^\top(\mu+\theta), w^\star = \mathbf{0}, \theta^\star = \mu.$$

When $\mu = \mathbf{0}$,

$$\min_\theta \max_w w^\top \theta$$

We will come back to this important example.

Exercise: show D_w is 1-Lipschitz for $\|w\|_2 \leq 1$

This example was studied by "Training GANs with Optimism" by C. Daskalakis et al., 2018 in the WGAN context

WGAN-GP (Wasserstein GAN with Gradient Penalty)

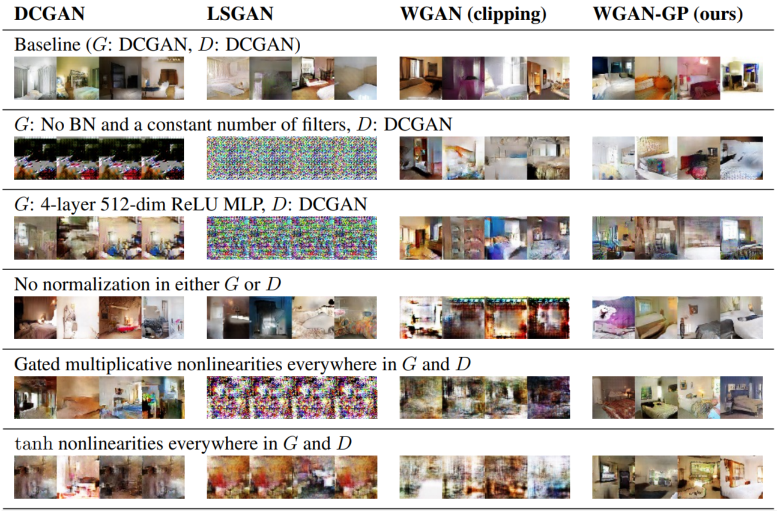
- WGAN encourages 1-Lipschitzness clipping all weights to lie in $[-c, c]$. As authors point out, this is not desirable.
- Instead, use Gradient Penalty, which penalizes norm that are greater than 1.

$$\min_{\theta} \max_{w} \mathbb{E}_{\mathbf{x} \sim Q(x)}[D_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim P(z)}[D_w(G_\theta(z))] + \lambda \mathbb{E}_{\bar{\mathbf{x}} \sim \overline{Q}(x)}[(\|\nabla D_w(\overline{x})\|_2 - 1)^2] \tag{14}$$

where $\bar{\mathbf{x}} = \epsilon \mathbf{x} + (1 - \epsilon) G_\theta(\mathbf{z}) \sim \overline{Q}(x), \epsilon \in [0, 1]$.

- Interpolation allows for a broader range of Lipschitz enforcement.

"Improved Training of Wasserstein GANs", Gulrajani et al., 2017.


DCGAN
LSGAN
WGAN (clipping)
WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)
G: No BN and a constant number of filters, D: DCGAN
G: 4-layer 512-dim ReLU MLP, D: DCGAN
No normalization in either G or D
Gated multiplicative nonlinearities everywhere in G and D
tanh nonlinearities everywhere in G and D

f-GAN

- Instead of minimizing a particular divergence function, minimize a general one: the f-divergence[7],

$$D_f(Q, P_G) = \int P_G(x) f\left[\frac{Q(x)}{P_G(x)}\right] \mathrm{d}x \tag{15}$$

where $f : \mathbb{R}_{>0} \to \mathbb{R}$ is a convex, continuous, $f(1) = 0$. This captures KL, Chi-square, Jensen-Shannon divergence, etc.

- Then it can be shown,

$$D_f(Q, P_G) \geq \max_w \mathbb{E}_{\mathbf{x}\sim Q(x)}[D_w(x)] + \mathbb{E}_{\mathbf{z}\sim P(z)}[-f^\star(D_w(G_\theta(\mathbf{z})))]$$

for some function w, where $f^\star$ is the Fenchel conjugate of f.

[7] f stands for Fenchel, a famous mathematician. The Fenchel conjugate is one of the most useful and used ideas in all of mathematics. Read "WHAT IS a Fenchel conjugate?" by H. H. Bauschke and Y. Lucet, 2012.

For your interest, the Fenchel conjugate is defined as $f^\star(x) = \max_{u\in\mathrm{domain}(f)} u^\top x - f(u)$

f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization, Nowozin, 2016

f-GAN

- Instead of minimizing a particular divergence function, minimize a general one: the f-divergence[7],

$$D_f(Q, P_G) = \int P_G(x) f\left[\frac{Q(x)}{P_G(x)}\right] \mathrm{d}x \tag{15}$$

  where $f : \mathbb{R}_{>0} \to \mathbb{R}$ is a convex, continuous, $f(1) = 0$. This captures KL, Chi-square, Jensen-Shannon divergence, etc.

- Then it can be shown,

$$D_f(Q, P_G) \geq \max_w \mathbb{E}_{\mathbf{x} \sim Q(x)}[D_w(x)] + \mathbb{E}_{\mathbf{z} \sim P(z)}[-f^\star(D_w(G_\theta(\mathbf{z})))]$$

  for some function w, where $f^\star$ is the Fenchel conjugate of f.

---

[7] f stands for Fenchel, a famous mathematician. The Fenchel conjugate is one of the most useful and used ideas in all of mathematics. Read "WHAT IS a Fenchel conjugate?" by H. H. Bauschke and Y. Lucet, 2012.

For your interest, the Fenchel conjugate is defined as $f^\star(x) = \max_{u \in \mathrm{domain}(f)} u^\top x - f(u)$

f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization, Nowozin, 2016

$$D_f(Q, P_G) \geq \max_w \mathbb{E}_{\mathbf{x}\sim Q(x)}[D_w(x)] + \mathbb{E}_{\mathbf{z}\sim P(z)}[-f^\star(D_w(G_\theta(\mathbf{z})))]$$

Observe minimizing over θ yields a generalized GAN objective:

- Let $D_w(x) = \theta(h_w(x))$, where θ is an arbitrary activation function of the last layer, h_w be the rest of the network.
- f-**GAN objective**,

$$\mathcal{L}(\theta, w) = \mathbb{E}_{\mathbf{x}\sim Q(x)}[\theta(h_w(x))] + \mathbb{E}_{\mathbf{z}\sim P(z)}[-f^\star(\theta(h_w(G_\theta(\mathbf{z}))))]$$

- For $\theta(v) = -\log(1+\exp(-v))$, $f^\star(t) = -\log(1-\exp(t))$, then (try show this),

$$\mathbb{E}_{\mathbf{x}\sim Q(x)}[-\log(1+\exp(-h_w(\mathbf{x})))] - \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1+\exp(h_w(G_\theta(\mathbf{z}))))]$$

But this is exactly our GAN objective (show this).

$$D_f(Q, P_G) \geq \max_w \mathbb{E}_{\mathbf{x}\sim Q(x)}[D_w(x)] + \mathbb{E}_{\mathbf{z}\sim P(z)}[-f^\star(D_w(G_\theta(\mathbf{z})))]$$

Observe minimizing over θ yields a generalized GAN objective:

- Let $D_w(x) = \theta(h_w(x))$, where θ is an arbitrary activation function of the last layer, h_w be the rest of the network.
- f-**GAN objective**,

$$\mathcal{L}(\theta, w) = \mathbb{E}_{\mathbf{x}\sim Q(x)}[\theta(h_w(x))] + \mathbb{E}_{\mathbf{z}\sim P(z)}[-f^\star(\theta(h_w(G_\theta(\mathbf{z}))))]$$

- For $\theta(v) = -\log(1+\exp(-v))$, $f^\star(t) = -\log(1-\exp(t))$, then (try show this),

$$\mathbb{E}_{\mathbf{x}\sim Q(x)}[-\log(1+\exp(-h_w(\mathbf{x})))] - \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1+\exp(h_w(G_\theta(\mathbf{z}))))]$$

But this is exactly our GAN objective (show this).

$$D_f(Q, P_G) \geq \max_w \mathbb{E}_{\mathbf{x}\sim Q(x)}[D_w(x)] + \mathbb{E}_{\mathbf{z}\sim P(z)}[-f^\star(D_w(G_\theta(\mathbf{z})))]$$

Observe minimizing over θ yields a generalized GAN objective:

- Let $D_w(x) = \theta(h_w(x))$, where θ is an arbitrary activation function of the last layer, h_w be the rest of the network.
- f-**GAN objective**,

$$\mathcal{L}(\theta, w) = \mathbb{E}_{\mathbf{x}\sim Q(x)}[\theta(h_w(x))] + \mathbb{E}_{\mathbf{z}\sim P(z)}[-f^\star(\theta(h_w(G_\theta(\mathbf{z}))))]$$

- For $\theta(v) = -\log(1 + \exp(-v))$, $f^\star(t) = -\log(1 - \exp(t))$, then (try show this),

$$\mathbb{E}_{\mathbf{x}\sim Q(x)}[-\log(1+\exp(-h_w(\mathbf{x})))] - \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1+\exp(h_w(G_\theta(\mathbf{z})))]$$

But this is exactly our GAN objective (show this).

The GAN Zoo

Name	**Saddle Functions (min θ, max w)**
MMGAN	$\mathbb{E}_{\mathbf{x}\sim Q(x)}[\log(D_w(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D_w(G_\theta(\mathbf{z})))]$
WGAN	$\mathbb{E}_{\mathbf{x}\sim Q(x)}[D_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim P(z)}[D_w(G_\theta(\mathbf{z}))] = \mathcal{W}$
WGAN-GP	$\mathcal{W} + \lambda \mathbb{E}_{\overline{\mathbf{x}}\sim \overline{Q}(x)}(\lVert \nabla D_w(\overline{\mathbf{x}})\rVert_2 - 1)^2, \overline{\mathbf{x}} = \epsilon \mathbf{x} + (1-\epsilon) G_\theta(\mathbf{z})$
f-GAN	$\mathbb{E}_{\mathbf{x}\sim Q(x)}[\theta(h_w(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim P(z)}[-f^\star(\theta(h_w(G(\mathbf{z}))))]$

Name	**Individual Losses $\mathcal{L}_D, \mathcal{L}_G$**
NSGAN	$\mathcal{L}_D = -\mathbb{E}_{\mathbf{x}\sim Q(x)}[\log(D_w(\mathbf{x}))] - \mathbb{E}_{\mathbf{z}\sim P(z)}[\log(1 - D_w(G_\theta(\mathbf{z})))]$
	$\mathcal{L}_G = -\mathbb{E}_{\mathbf{z}\sim P(z)}[\log(D_w(G_\theta(\mathbf{z})))]$
LSGAN	$\mathcal{L}_D = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim Q(x)}[(D_w(\mathbf{x}) - 1)^2] + \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim P(z)}[(D_w(G_\theta(\mathbf{z})))^2]$
	$\mathcal{L}_G = \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim P(z)}[(D_w(G_\theta(\mathbf{z})) - 1)^2]$

For more examples, see: "Are GANs Created Equal? A Large-Scale Study", Lucic et al., 2018

Evolution of GAN Dynamics

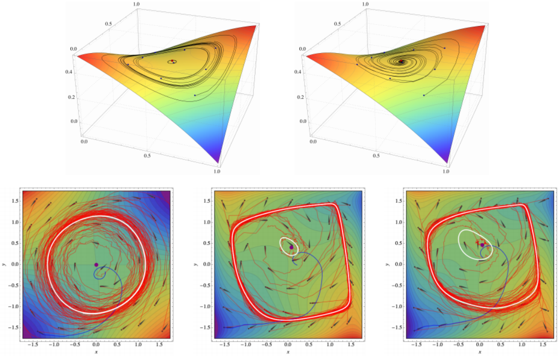
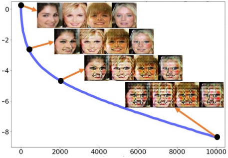


Figure: (Top) "Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile" - Mertikopoulos et al., 2018 (Bottom) "The limits of min-max optimization algorithms: convergence to spurious non-critical sets" - Hsieh et al., 2020.

Problems with Training GAN

- **Mode collapse**: generator produces from a localized region of the true targets distribution

- **Instability**: "The more you train, the worse it gets"



Obviously has something to do with weight update. How to fix these issues?

(Top) "Least Squares Generative Adversarial Networks", Mao et al., 2017, (Bottom) "GANs May Have No Nash Equilibria", Farnia, et al., 2020

Back to the Basics: The Bilinear Zero-Sum Game

$$\min_{\theta \in \mathbb{R}} \max_{w \in \mathbb{R}} \mathcal{L}(\theta, w) = \theta^\top w \tag{16}$$

Exists a unique saddle point at $(\theta^\star, w^\star) = (0, 0)$

- Coincides with some WGAN formulation (as we have seen)
- Many neural networks at the last layer is linear
- If dynamics oscillates/diverges in simple game, then intuitively it oscillates/diverges in more complicated games

Gradient descent ascent (GDA)[8] ($\approx$ Algo for Training Min-Max GAN): ϵ is the step-size.

$$\theta_{k+1} = \theta_k - \epsilon \nabla_\theta \mathcal{L}(\theta_k, w_k)$$
$$w_{k+1} = w_k + \epsilon \nabla_w \mathcal{L}(\theta_k, w_k)$$

For us, $\nabla_\theta \mathcal{L} = w, \nabla_w \mathcal{L} = \theta$

[8] "Iterative methods for concave programming", H. Uzawa, 1958

$$\theta_{k+1} = \theta_k - \epsilon w_k \qquad w_{k+1} = w_k + \epsilon \theta_k$$

For GDA (show this):

$$\|\theta_{k+1}\|_2^2 + \|w_{k+1}\|_2^2 = (1+\epsilon^2)(\|\theta_k\|_2^2 + \|w_k\|_2^2)$$

If $(1+\epsilon^2) > 1$, then the magnitude of θ_{k+1}, w_{k+1} blows up

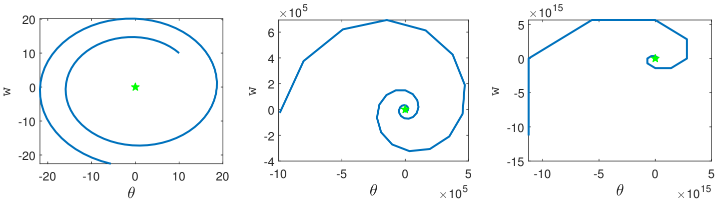


Figure: GDA: $(\theta_0, w_0) = (10, 10)$ (Left) $\epsilon = 0.1$ (Center) $\epsilon = 0.5$ (Right) $\epsilon = 1$. All divergent, Spiraling Out.

What if We “Alternate”?

Instead of,

$$\theta_{k+1} = \theta_k - \epsilon w_k$$
$$w_{k+1} = w_k + \epsilon \theta_k$$

Do this (alternating),

$$\theta_{k+1} = \theta_k - \epsilon w_k$$
$$w_{k+1} = w_k + \epsilon \theta_{k+1}$$

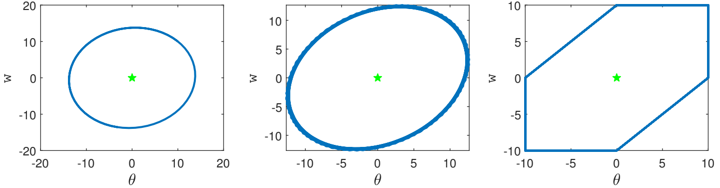


Figure: Alternating GDA: $(\theta_0, w_0) = (10, 10)$ (Left) $\epsilon = 0.1$ (Center) $\epsilon = 0.5$ (Right) $\epsilon = 1$. Periodic orbits or limit cycles (Poincaré, 1882).

A Simple Technique: Averaging

Moving Average (off-line) [9]:

$$w_k^{\mathrm{MA}} = \frac{1}{k} \sum_{j=1}^{k} w_j \tag{17}$$

Moving Average (online):

$$w_k^{\mathrm{MA}} = \frac{k-1}{k} w_{k-1}^{\mathrm{MA}} + \frac{1}{k} w_k \tag{18}$$

Exponential Moving Average (online):

$$w_k^{\mathrm{MA}} = \beta w_{k-1}^{\mathrm{MA}} + (1-\beta) w_k, \beta > 0 \tag{19}$$

[9] Introduced by Bruck, Nemirovskii, et al. in the optimization literature, independently and *extensively* studied in game theory, e.g., "Learning in games via reinforcement learning and regularization", Mertikopoulos and Sandholm, 2016, "Time-Averaged Replicator and Best-Reply Dynamics", Hofbauer et al., 2009, "Time Averages for Heteroclinic Attractors", Gaunersdorfer, et al. 1992.

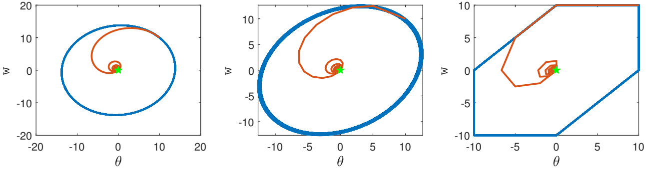


Figure: MA for Alternating GDA: $(\theta_0, w_0) = (10, 10)$ (Left) $\epsilon = 0.1$ (Center) $\epsilon = 0.5$ (Right) $\epsilon = 1$

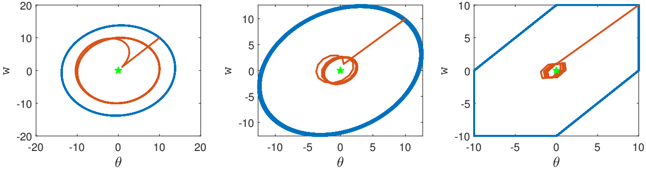


Figure: EMA for Alternating GDA: $(\theta_0, w_0) = (10, 10), \beta = 0.9$ (Left) $\epsilon = 0.1$ (Center) $\epsilon = 0.5$ (Right) $\epsilon = 1$

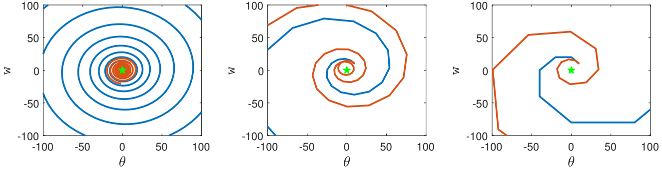


Figure: MA for Simultaneous GDA (Orange) vs Original Trajectories (Blue): $(\theta_0, w_0) = (10, 10)$ (Left) $\epsilon = 0.1$ (Center) $\epsilon = 0.5$ (Right) $\epsilon = 1$. All divergent.

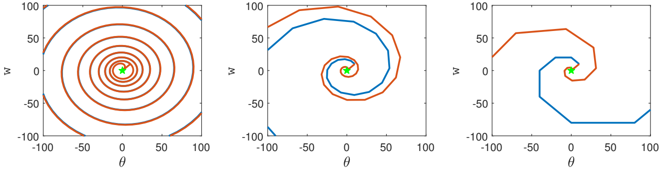


Figure: EMA for Simultaneous GDA: $(\theta_0, w_0) = (10, 10), \beta = 0.9$ (Left) $\epsilon = 0.1$ (Center) $\epsilon = 0.5$ (Right) $\epsilon = 1$. All divergent.

For non-convex, non-concave problems, couple it with ADAM.

$$\begin{cases} w_k, \theta_k & \leftarrow \text{ADAM}(\mathcal{L}_D, \mathcal{L}_G) \\ w_k^{\text{EMA}} & = \beta w_{k-1}^{\text{EMA}} + (1-\beta) w_k \quad \theta_k^{\text{EMA}} = \beta \theta_{k-1}^{\text{EMA}} + (1-\beta)\theta_k \end{cases}$$

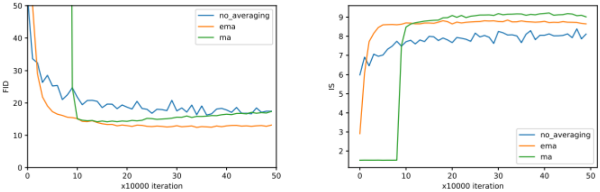


Figure: CIFAR-10 FID and IS score for Original GAN[10].

Has been used in BigGAN, Progressive Growing of GAN, etc.

But how do we get convergence of the actual trajectory?

10 "The Unusual Effectiveness of Averaging in GAN Training" - Yazıcı et al., 2018

Proximal Point Update

Proximal Point update[11] is,

$$\theta_{k+1} = \theta_k - \epsilon \nabla_w \mathcal{L}(\theta_{k+1}, w_{k+1}) \quad w_{k+1} = w_k + \epsilon \nabla_\theta \mathcal{L}(\theta_{k+1}, w_{k+1})$$

Note that the gradient is taken at $k+1$ step, so we need to invert the equation to calculate θ_{k+1}, w_{k+1}.

For bilinear ZS game, $\nabla_w \mathcal{L} = \theta, \nabla_\theta \mathcal{L} = w$, (show the following),

$$\theta_{k+1} = \theta_k - \epsilon w_{k+1} \implies \theta_{k+1} = \frac{1}{1+\epsilon^2}(\theta_k - \epsilon w_k)$$
$$w_{k+1} = w_k + \epsilon \theta_{k+1} \implies w_{k+1} = \frac{1}{1+\epsilon^2}(w_k + \epsilon \theta_k)$$

[11] "Brève communication. Régularisation d'inéquations variationnelles par approximations successives", B. Martinet, 1970

$$\theta_{k+1} = \frac{1}{1+\epsilon^2}(\theta_k - \epsilon w_k) \quad w_{k+1} = \frac{1}{1+\epsilon^2}(w_k + \epsilon\theta_k)$$

For Proximal Point update (show this):

$$\|\theta_{k+1}\|_2^2 + \|w_{k+1}\|_2^2 = \frac{1}{1+\epsilon^2}(\|\theta_k\|_2^2 + \|w_k\|_2^2)$$

We have convergence for $1/(1+\epsilon^2) < 1$!

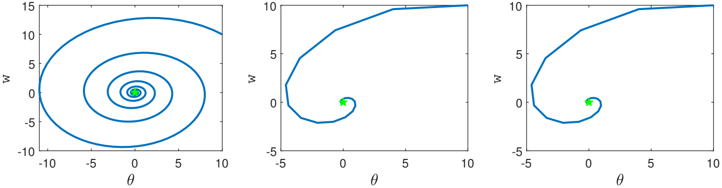


Figure: $(\theta_0, w_0) = (10, 10)$ (Left) $\epsilon = 0.1$ (Center) $\epsilon = 0.5$ (Right) $\epsilon = 1$

But in general, the inversion procedure is hard. How to relax?

Optimistic GDA/GDA with Negative Momentum

Given the **Proximal Point** update,

$$\theta_{k+1} = \theta_k - \epsilon \nabla_\theta \mathcal{L}(\theta_{k+1}, w_{k+1})$$
$$w_{k+1} = w_k + \epsilon \nabla_w \mathcal{L}(\theta_{k+1}, w_{k+1})$$

Make the approximation $\nabla \mathcal{L}_{k+1} \approx \nabla \mathcal{L}_k + (\nabla \mathcal{L}_k - \nabla \mathcal{L}_{k-1})$, we obtain the **Optimistic gradient descent ascent** (OGDA)[12]:

$$\theta_{k+1} = \theta_k - \epsilon \nabla_\theta \mathcal{L}(\theta_k, w_k) + \epsilon(\nabla_\theta \mathcal{L}(\theta_{k-1}, w_{k-1}) - \nabla_\theta \mathcal{L}(\theta_k, w_k))$$
$$w_{k+1} = w_k + \epsilon \nabla_w \mathcal{L}(\theta_k, w_k) - \epsilon(\nabla_w \mathcal{L}(\theta_{k-1}, w_{k-1}) - \nabla_w \mathcal{L}(\theta_k, w_k)),$$

$\theta_{-1} = \theta_0, w_{-1} = w_0.$

For our bilinear zero-sum game, OGDA becomes,

$$\theta_{k+1} = \theta_k - 2\epsilon w_k + \epsilon w_{k-1} \quad w_{k+1} = w_k + 2\epsilon\theta_k - \epsilon\theta_{k-1}$$

[12] "A modification of the Arrow-Hurwicz method for search of saddle point", L. D. Popov, 1980

Extragradient

Given the **Proximal Point** update,

$$\theta_{k+1} = \theta_k - \epsilon \nabla_\theta \mathcal{L}(\theta_{k+1}, w_{k+1})$$
$$w_{k+1} = w_k + \epsilon \nabla_w \mathcal{L}(\theta_{k+1}, w_{k+1})$$

Can re-write as,

$$\theta_{k+1} = \theta_k - \epsilon \nabla_\theta \mathcal{L}(\theta_k - \epsilon \nabla_\theta \mathcal{L}(\theta_{k+1}, w_{k+1}), w_k + \epsilon \nabla_w \mathcal{L}(\theta_{k+1}, w_{k+1}))$$
$$w_{k+1} = w_k + \epsilon \nabla_w \mathcal{L}(\theta_k - \epsilon \nabla_\theta \mathcal{L}(\theta_{k+1}, w_{k+1}), w_k + \epsilon \nabla_w \mathcal{L}(\theta_{k+1}, w_{k+1}))$$

Make the approximation $\nabla \mathcal{L}_{k+1} \approx \nabla \mathcal{L}_k$,

$$\theta_{k+1} = \theta_k - \epsilon \nabla_\theta \mathcal{L}(\theta_k - \epsilon \nabla_\theta \mathcal{L}(\theta_k, w_k), w_k + \epsilon \nabla_w \mathcal{L}(\theta_k, w_k))$$
$$w_{k+1} = w_k + \epsilon \nabla_w \mathcal{L}(\theta_k - \epsilon \nabla_\theta \mathcal{L}(\theta_k, w_k), w_k + \epsilon \nabla_w \mathcal{L}(\theta_k, w_k))$$

This is the **extragradient algorithm** (EG) [13]

[13] "The extragradient method for finding saddle points and other problems", G. M. Korpelevich, 1976

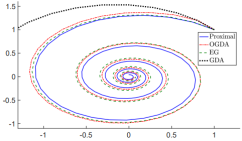


Figure: Bilinear Zero-Sum Game

Remember our "hard" Dirac GAN game from the beginning.

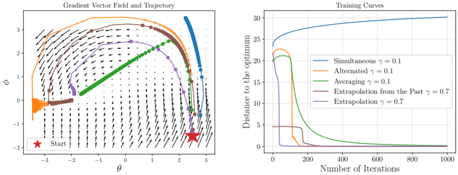


Figure: Dirac GAN

Figure: (Top) "A Unified Analysis of Extra-gradient and Optimistic Gradient Methods for Saddle Point Problems: Proximal Point Approach", A. Mokhtari, A. Ozdaglar, S. Pattathil, 2019. (Bottom) "A Variational Inequality Perspective on Generative Adversarial Networks", G. Gidel, H. Berard, G. Vignoud, P. Vincent, S. Lacoste-Julien, 2018

Other Dynamics/Techniques

- (Optimistic) Mirror Descent[14]: essentially a generalization of GDA to constrained setting.
- "Two Time-Scale Method"[15]: GDA but with added noise.
- Methods based on Game Hessian/Game Jacobian, e.g., consensus optimization[16]. CS/ML folks may have the most impact here.

Open question: is it true that more complicated the game $\implies$ more complicated dynamics?

---

[14] "Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile", P. Mertikopoulos et al., 2018

See also: "Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems", A. Nemirovski, 2004

[15] "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium", M. Heusel et al., 2018

See also: "Stochastic Approximation with Two Time Scales", V. Borkar, 1997

[16] "The Numerics of GANs", Mescheder, et al., 2018

Where do I go from here?

Some ideas:

- Combine idea with computer graphics (heavily explored)
- Come up with a new objective function that makes sense (heavily explored, e.g., f-GAN.)
- Come up with a new training technique that work well in non-convex/non-concave domain (hard)
- Pick an "outside of the box" application, particle physics (large Hadron collider), dentistry (Personalized GANufacturing) (plenty of opportunities left here)
- Combine GAN with other domains of learning, e.g., imitation learning (GAIL) (a bit unexplored here)
- Put GAN on firm game-theoretic grounds (very under-explored)

So What Should You Name Your GAN?

Some ideas that are already taken:

- VEEGAN `https://arxiv.org/abs/1705.07761`
- (Lady) GAGAN `http://jeankossaifi.com/pages/gagan.html`
- DRAGAN "How to train your DRAGAN" `https://arxiv.org/abs/1705.07215v1`
- GANdalf `https://arxiv.org/abs/2008.04396`
- GANGs `https://arxiv.org/abs/1712.00679`

Potential ideas (?):

- GANGNam, LOGAN, PAGAN, SLOGAN

The catch: if you name it, you will have to present it.

Parting Message

- Tremendous challenges + opportunities still remain
- Be thorough with literature review: GANs are new but *game and dynamical system theory* are ancient

  "...the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity." - R. Tyrrell Rockafellar, 1993

- So, to You: "What is the great watershed in games + GAN?"

  I view the current era as focused on pattern recognition...So what's next? I would call this the era of markets. So it's not just one agent making decisions in the classical AI sense...but a huge interconnected, planetary scale web of data, agents and decisions. – Michael I. Jordan, Prof. UC Berkeley, 2020

Good luck!

Appendix

Figure: Imaginary Image Generated by Chimera Painter by Google AI

Appendix: Other Cool GANs

Cool GANs Architectures

- LapGAN, Coupled GAN, BiGAN, Big BiGAN, SAGAN, ProGAN

Cool GAN Objectives

- CycleGAN, DiscoGAN, PairedCycleGAN, StarGAN, ComboGAN, SRGAN, OTGAN, SNGAN

GAN for Video Generation

- Temporal GAN (TGAN), VGAN, MoCoGAN, TGANv2

GAN That Generates What You Tell It To

- Conditional GAN, Auxiliary Classifier GAN, Info GAN, Pix2Pix

GAN That Addresses Mode Collapse

- Unrolled GAN, VEEGAN, PacGAN

Appendix: "Weird" / "Cool" GAN Websites

StyleGAN Related

https://www.whichfaceisreal.com/
https://www.thispersondoesnotexist.com
http://thesecatsdonotexist.com/
https://www.thiswaifudoesnotexist.net/
https://www.artbreeder.com/

Sketch To Painting

https://affinelayer.com/pixsrv/
http://gandissect.res.ibm.com/ganpaint.html https://storage.googleapis.com/chimera-painter/index.html
http://nvidia-research-mingyuliu.com/gaugan/

Digital Art Using GAN

http://www.obvious-art.com/ukiyo/index.html
https://refikanadol.com/

Appendix: Some Resources if You Are Interested in Games

Conference and Workshops:

1. NeurIPS: Smooth Games Optimization and Machine Learning Workshop, Bridging Game Theory & Deep Learning
2. Fields Institute Workshop on Dynamics, Optimization and Variational Analysis in Applied Games
3. Games, Dynamics and Optimization (GDO)
4. IEEE Control and Decision Conference (CDC)