What’s wrong with GANs?

Yossi Keshet

Bar-Ilan University

May 18, 2020
GAN: two-player game

Train jointly the generator network $G_{\theta_G}(z)$ and the discriminator network $D_{\theta_D}(x)$:

$$\min_{\theta_G} \max_{\theta_D} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_D}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_D}(G_{\theta_G}(z))) \right]$$
GAN: two-player game

The objective:

$$\min_{\theta_G} \max_{\theta_D} \left[ \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_D}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_D}(G_{\theta_G}(z))) \right]$$

is solved by alternating between

1. Gradient ascent on $\theta_D$

$$\max_{\theta_D} \left[ \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_D}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_D}(G_{\theta_G}(z))) \right]$$

2. Gradient decent on $\theta_G$

$$\min_{\theta_G} \left[ \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_D}(G_{\theta_G}(z))) \right]$$
\[
\min_{\theta_G} \left[ \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_D}(G_{\theta_G}(z))) \right]
\]

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!

Gradient signal dominated by region where sample is already good

\[
\max_{\theta_G} \left[ \mathbb{E}_{z \sim p(z)} \log(D_{\theta_D}(G_{\theta_G}(z))) \right]
\]

High gradients
Low gradients
GAN training algorithm

for number of training iterations do
  for $k$ steps do
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Sample minibatch of $m$ examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{data}(x)$.
    • Update the discriminator by ascending its stochastic gradient:
      \[
      \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D \left( x^{(i)} \right) + \log \left( 1 - D \left( G \left( z^{(i)} \right) \right) \right) \right].
      \]
  end for
  • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
  • Update the generator by descending its stochastic gradient:
    \[
    \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left( 1 - D \left( G \left( z^{(i)} \right) \right) \right).
    \]
end for

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

Arjovsky and Buttou (2017) raised a series of claims:

- Why is GAN training massively unstable?
- Why do updates get worse as the discriminator gets better? Both in the original and the new cost function.
- Is there a way to avoid some of these issues?

Arjovsky and Buttou, “Towards principled methods for training Generative Adversarial Networks”, 2017
Property 1: GAN is optimizing JS-divergence

We start by showing that minimizing the GAN objective function with an optimal discriminator is equivalent to minimizing the JS-divergence.

Recall that KL-Divergence is defined as:

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

The Jensen–Shannon divergence (JS-Divergence) is defined as:

$$D_{JS}(p||q) = \frac{1}{2} D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2} D_{KL}(q||\frac{p+q}{2})$$
Property 1: GAN is optimizing JS-divergence

Recall that the objective of GAN is the following

$$\min_{\theta_G} \max_{\theta_D} V(\theta_G, \theta_D) = \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_D}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_D}(G_{\theta_G}(z)))$$

We will prove that at the optimal point:

- \( p = q \)
- The discriminator cannot distinguish the real from the fake \( D_{\theta_D^*}(x) = 1/2 \)
- The objective \( V(\theta_G^*, \theta_D^*) \) equals \(-2 \log 2\).
Property 1: GAN is optimizing JS-divergence

\[
\min_{\theta_G} \max_{\theta_D} V(\theta_G, \theta_D) = \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_D}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_D}(G_{\theta_G}(z)))
\]

\[
= \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_D}(x) + \mathbb{E}_{x \sim p_{\text{gen}}} \log(1 - D_{\theta_D}(x))
\]

\[
= \int_x p_{\text{data}}(x) \log D_{\theta_D}(x) + p_{\text{gen}}(x) \log(1 - D_{\theta_D}(x)) \, dx
\]

If \( G \) is fixed, i.e., the parameters \( \theta_G \) are fixed, the optimal discriminator issues

\[
D_{\theta_D}^* = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)}
\]

The optimal \( y \) for \( a \log(y) + b \log(1 - y) \) is obtained by deriving and comparing to zero:

\[
\frac{d}{dy} \left[ a \log(y) + b \log(1 - y) \right] = \frac{a}{y} - \frac{b}{1 - y} = 0 \quad \Rightarrow \quad y^* = \frac{a}{a + b}
\]
Property 1: GAN is optimizing JS-divergence

Plug the optimal $D_{\theta_D^*} = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)}$ back into the objective:

$$
\min_{\theta_G} V(\theta_G, \theta_D^*) = \int_x p_{\text{data}}(x) \log D_{\theta_D^*}(x) + p_{\text{gen}}(x) \log (1 - D_{\theta_D^*}(x)) \, dx
$$

$$
= \int_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)} + p_{\text{gen}}(x) \log \left(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)}\right) \, dx
$$

$$
= \int_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)} + p_{\text{gen}}(x) \log \frac{p_{\text{gen}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)} \, dx
$$

$$
= 2 \int_x \frac{1}{2} \left[ p_{\text{data}}(x) \log \frac{2p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)} - \log 2 \right]
$$

$$
\quad + \frac{1}{2} \left[ p_{\text{gen}}(x) \log \frac{2p_{\text{gen}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)} - \log 2 \right] \, dx
$$

$$
= 2D_{JS}(p_{\text{data}}||p_{\text{gen}}) - \log 4
$$
Property 1: GAN is optimizing JS-divergence

We got

1. \[ \min_{\theta_G} V(\theta_G, \theta_D^*) = 2D_{JS}(p_{\text{data}} \| p_{\text{gen}}) - \log 4 \]

2. Hence the optimal point is when \( p_{\text{data}} = p_{\text{gen}} \).

3. The discriminator at the optimum is

\[ D_{\theta_D^*} = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{gen}}(x)} = \frac{1}{2} \]

4. The objective at the optimum is

\[ \min_{\theta_G} \max_{\theta_D} V(\theta_G, \theta_D) = \mathbb{E}_{x \sim p_{\text{data}}} \log \frac{1}{2} + \mathbb{E}_{x \sim p_{\text{gen}}} \log(1 - \frac{1}{2}) = -2 \log 2 \]
Property 1: GAN is optimizing JS-divergence

Minimizing JS-Divergence creates some problem with the distance between $p_{\text{data}}$ and $p_{\text{gen}}$ is large. Below is a plot of a distribution $p$ and a distribution $q$ with different means:
Property 1: GAN is optimizing JS-divergence

The corresponding KL-divergence and JS-divergence between $p$ and $q$ with means. When both $p$ and $q$ are the same, the divergence is 0. As the mean of $q$ increases, the divergence increases. The gradient of the divergency will eventually diminish.

![Graphs showing KL-divergence and JS-divergence](image-url)
Property 2: Vanishing gradients on the generator

The paper shows that

\[
\lim_{\|D(\theta_D) - D(\theta_D^*)\| \to 0} \nabla_{\theta_G} \mathbb{E}_{z \sim p(z)} [\log (1 - D(G_{\theta_G}(z)))] = 0
\]

The gradient vanishes when the discriminator becomes optimal (\(D\) is close to \(D^*\)):

\[
\nabla_{\theta_G} \mathbb{E}_{z \sim p(z)} [\log (1 - D(G_{\theta_G}(z)))] \to 0
\]

What about the alternative loss function of the generator?
Property 2: Vanishing gradients on the generator

The paper shows that

$$\lim_{\|D(\theta_D) - D(\theta_D^*)\| \to 0} \nabla_{\theta_G} \mathbb{E}_{z \sim p(z)} [\log(1 - D(G_{\theta_G}(z))] = 0$$

The gradient vanishes when the discriminator becomes optimal ($D$ is close to $D^*$):

$$\nabla_{\theta_G} \mathbb{E}_{z \sim p(z)} [\log(1 - D(G_{\theta_G}(z))] \to 0$$

What about the alternative loss function of the generator?
Property 3: Instability of the generator gradient updates

First it can be shown that

\[ \mathbb{E}_{z \sim p(z)}[-\nabla_{\theta} \log(1 - D^*(G_{\theta}G(z))] = \nabla_{\theta}G \left[ D_{KL}(p_{\text{gen}} \| p_{\text{data}}) - 2D_{JS}(p_{\text{gen}} \| p_{\text{data}}) \right] \]

The reverse KL term assigns high cost in generating unnatural images while mode dropping is more acceptable. i.e. it generates more natural images but mode may collapse.

Second, it can be shown that \( \mathbb{E}_{z \sim p(z)}[-\nabla_{\theta} \log(1 - D^*(G_{\theta}G(z))] \) is centered Cauchy distribution with infinite expectation and variance. This means that the expected update will be 0, providing no feedback to the gradient.
Property 3: Instability of the generator gradient updates

First it can be shown that

$$\mathbb{E}_{z \sim p(z)}[-\nabla_{\theta} G \log (1 - D^*(G_{\theta} G(z)))] = \nabla_{\theta} G \left[ D_{KL}(p_{\text{gen}} \mid \mid p_{\text{data}}) - 2D_{JS}(p_{\text{gen}} \mid \mid p_{\text{data}}) \right]$$

The reverse KL term assigns high cost in generating unnatural images while mode dropping is more acceptable. i.e. it generates more natural images but mode may collapse.

Second, it can be shown that $$\mathbb{E}_{z \sim p(z)}[-\nabla_{\theta} G \log (1 - D^*(G_{\theta} G(z)))]$$ is centered Cauchy distribution with infinite expectation and variance. This means that the expected update will be 0, providing no feedback to the gradient.
Property 3: Instability of the generator gradient updates

The norm of the gradient grows drastically as we train the discriminator closer to optimality, at any stage in training of a well stabilized DCGAN – except when it has already converged:
Possible remedy: adding noise to the input of the discriminator

Add noise (continuous) to the inputs of the discriminator to smoothen the data distribution of the probability mass.

The original GAN:
Possible remedy: adding noise to the input of the discriminator

Add noise (continuous) to the inputs of the discriminator to smoothen the data distribution of the probability mass.

The noisy GAN:

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

\[
- \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D(G(z^{(i)})) \right)
\]
Possible remedy: adding noise to the input of the discriminator

The new gradients, for adding noise $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$:

$$
\mathbb{E}_{z \sim p(z), \epsilon \sim \mathcal{N}(0, \sigma^2 I)} \left[ -\nabla_{\theta_G} \log(1 - D^*(G_{\theta_G}(z))) \right] = 2\nabla_{\theta_G} D_{JS}(p_{\text{data}} + \epsilon \parallel p_{\text{gen}} + \epsilon)
$$

which is not zero when $p_{\text{data}}$ and $p_{\text{gen}}$ are very different. This is excellent news because the generator’s gradient is not vanishing when the discriminator is optimal.
Wasserstein GAN

We saw that the standard GAN optimizes the JS-Divergence.

Wasserstein GAN minimizes Earth-Mover (EM) distance or Wasserstein Metric.

We will see how this solve some of the problems of GAN.
Earth-Mover (EM) distance
Earth-Mover (EM) distance

\[(6 + 6 + 6 + 6 + 2 \times 9 = 42)\]

\[\gamma_1\]

\[
\begin{array}{l}
1 & 1 & 0 & 0 & 2 \\
2 & 0 & 1 & 0 & 0 \\
3 & 0 & 0 & 2 & 0 \\
\end{array}
\]

\[(6 + 6 + 6 + 8 + 9 + 7 = 42)\]

\[\gamma_2\]

\[
\begin{array}{l}
1 & 1 & 0 & 1 & 1 \\
2 & 0 & 1 & 0 & 0 \\
3 & 0 & 0 & 1 & 1 \\
\end{array}
\]
The Wasserstein distance (or the EM distance) is the cost of the cheapest transport plan.

\[
\begin{array}{cccc}
1 & 1 & 2 & 2 \\
3 & 4 & 6 & 7 \\
\end{array}
\]

\[
\begin{array}{cccc}
2 & 1 & 2 & 1 \\
3 & 4 & 6 & 7 \\
\end{array}
\]
The Wasserstein distance is the minimum cost of transporting mass in converting the data distribution $q$ to the data distribution $p$:

$$W(p_{\text{data}}, p_{\text{gen}}) = \inf_{\gamma \in \Pi(p_{\text{data}}, p_{\text{gen}})} \mathbb{E}_{(x, \tilde{x}) \sim \gamma} [\|x - \tilde{x}\|]$$

where $\Pi(p_{\text{data}}, p_{\text{gen}})$ is the set of all disjoint distributions $\gamma(x, \tilde{x})$ whose margins are $p_{\text{data}}$ and $p_{\text{gen}}$, respectively.
**Wasserstein distance**

It can be shown that minimizing:

\[ W(p_{\text{data}}, p_{\text{gen}}) = \inf_{\gamma \in \Pi(p_{\text{data}}, p_{\text{gen}})} \mathbb{E}_{(x, \tilde{x}) \sim \gamma} \left[ \| x - \tilde{x} \| \right] \]

is equal to maximizing

\[ W(p_{\text{data}}, p_{\text{gen}}) = \sup_{\| f \|_L \leq 1} \mathbb{E}_{p_{\text{data}}} [f(x)] - \mathbb{E}_{p_{\text{gen}}} [f(x)] \]

It is called Kantorovich-Rubinstein duality.

More details can be found here:

https://medium.com/@jonathan_hui/gan-spectral-normalization-893b6a4e8f53
Wasserstein distance

We need to find $f$ which is a 1-Lipschitz function that follows the constraint

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|$$

and maximizes

$$\mathbb{E}_{p_{data}}[f(x)] - \mathbb{E}_{p_{gen}}[f(x)]$$

More details can be found here:
https://medium.com/@jonathan_hui/gan-spectral-normalization-893b6a4e8f53
Wasserstein GAN

Wasserstein GAN diagram is composed of a Critic (instead of a discriminator):

<table>
<thead>
<tr>
<th>Discriminator/Critic</th>
<th>Generator</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GAN</strong></td>
<td></td>
</tr>
<tr>
<td>[ \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D(x^{(i)}) + \log \left( 1 - D(G(z^{(i)})) \right) \right] ]</td>
<td>[ \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left( D(G(z^{(i)})) \right) ]</td>
</tr>
<tr>
<td><strong>WGAN</strong></td>
<td></td>
</tr>
<tr>
<td>[ \nabla_{w} \frac{1}{m} \sum_{i=1}^{m} \left[ f(x^{(i)}) - f(G(z^{(i)})) \right] ]</td>
<td>[ \nabla_{\theta} \frac{1}{m} \sum_{i=1}^{m} f(G(z^{(i)})) ]</td>
</tr>
</tbody>
</table>

Because it can’t really discriminate between real and fake, the WGAN discriminator is actually called a *critic* instead of a *discriminator*. This distinction has theoretical importance, but for practical purposes we can treat it as an acknowledgement that the inputs to the loss functions don’t have to be probabilities.
Wasserstein GAN

Wasserstein GAN diagram is composed of a Critic (instead of a discriminator):

\[ \nabla_w \left[ \frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)})) \right] \]

\[ \nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)})) \]
Wasserstein GAN diagram is composed of a Critic (instead of a discriminator):

\[
\begin{align*}
\text{GAN} & \quad \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D(x^{(i)}) + \log (1 - D(G(z^{(i)}))) \right] \\
\text{WGAN} & \quad \nabla_{w} \frac{1}{m} \sum_{i=1}^{m} \left[ f(x^{(i)}) - f(G(z^{(i)})) \right] \\
\text{Generator} & \quad \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log (D(G(z^{(i)})))
\end{align*}
\]

For WGAN \( f \) has to be a 1-Lipschitz function.
Lipschitz continuity

A real function $f : \mathbb{R} \rightarrow \mathbb{R}$ is Lipschitz continuous if

$$|f(x_1) - f(x_2)| \leq k|x_1 - x_2|$$

where $k$ is called the Lipschitz constant of the function.

Lipschitz constant equals the maximum absolute value of the derivatives.

Graphically, we can construct a double cone with slope $k$ and $-k$. If we move its origin along the graph, the graph will always stay outside the cone.
Wasserstein GAN

Wasserstein GAN diagram is composed of a Critic (instead of a discriminator):

\[
\begin{align*}
\text{GAN} & \quad \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D(x^{(i)}) + \log (1 - D(G(z^{(i)}))) \right] \\
\text{WGAN} & \quad \nabla_w \frac{1}{m} \sum_{i=1}^{m} \left[ f(x^{(i)}) - f(G(z^{(i)})) \right] \\
\text{Generator} & \quad \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log (D(G(z^{(i)}))) \\
\text{Discriminator/Critic} & \quad \nabla_{\theta} \frac{1}{m} \sum_{i=1}^{m} f(G(z^{(i)}))
\end{align*}
\]

For WGAN \( f \) has to be a 1-Lipschitz function.

To enforce the constraint, WGAN applies a very simple clipping to restrict the maximum weight value in \( f \), i.e. the weights of the discriminator must be within a certain range controlled by the hyperparameters \( c \).
Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size. $n_{\text{critic}}$, the number of iterations of the critic per generator iteration.

Require: $w_0$, initial critic parameters. $\theta_0$, initial generator’s parameters.

1: while $\theta$ has not converged do
2:     for $t = 0, \ldots, n_{\text{critic}}$ do
3:         Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
4:         Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
5:         $g_w \leftarrow \nabla_w \left[ \frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)})) \right]$
6:         $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
7:         $w \leftarrow \text{clip}(w, -c, c)$
8:     end for
9:     Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
10:    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
11:    $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
12: end while
Wasserstein GAN: Examples