Improving Generative Adversarial Network
Improving GAN

- Quick Review of GAN
- Unified framework of GAN
- Selecting a better divergence: W-GAN
- Evaluation
Review
Basic Idea of GAN

- The data we want to generate has a distribution $P_{data}(x)$

$P_{data}(x)$

High Probability

Low Probability

Image Space
Basic Idea of GAN

• A generator G is a network. The network defines a probability distribution.

\[
\begin{align*}
\text{Normal Distribution} & \quad \rightarrow \quad \text{generator } G \\
\text{generator } G & \quad \rightarrow \quad x = G(z) \\
\end{align*}
\]

It is difficult to compute \( P_G(x) \).
We do not know what the distribution looks like.

https://blog.openai.com/generative-models/
Basic Idea of GAN

Normal Distribution

\[ \text{Normal Distribution} \]

\[ \text{NN Generator v1} \]

\[ P_{data}(x) \]

\[ P_G(x) \]

It can be proofed that the **loss the discriminator** related to **JS divergence**.
Basic Idea of GAN

• **Next step:**
  • Updating the parameters of generator
  • To minimize the JS divergence

The output be classified as “real” (as close to 1 as possible)

Generator + Discriminator = a network

Using gradient descent to update the parameters in the generator, but fix the discriminator
Unified Framework

Original GAN using binary classifier to evaluate JS divergence of two distributions

Actually, we can evaluate any f-divergence.

**f-divergence**  

$P$ and $Q$ are two distributions. $p(x)$ and $q(x)$ are the probability of sampling $x$.  

$$D_f(\mathcal{P} \parallel \mathcal{Q}) = \int q(x) f \left( \frac{p(x)}{q(x)} \right) dx$$  

$f$ is convex  

$f(1) = 0$  

If $p(x) = q(x)$ for all $x$  

$$D_f(\mathcal{P} \parallel \mathcal{Q}) = \int q(x) f \left( \frac{p(x)}{q(x)} \right) dx = 0$$  

Because $f$ is convex  

$$\geq f \left( \int q(x) \frac{p(x)}{q(x)} dx \right)$$  

$$= f(1) = 0$$  

When $P$ and $Q$ are the same distribution,  

$D_f(\mathcal{P} \parallel \mathcal{Q})$ has the smallest value, which is $0$. 
**f-divergence**

\[ D_f(P \| Q) = \int q(x) f \left( \frac{p(x)}{q(x)} \right) dx \]

\( f \) is convex

\( f(1) = 0 \)

\( f(x) = x \log x \)

\[ D_f(P \| Q) = \int q(x) \frac{p(x)}{q(x)} \log \left( \frac{p(x)}{q(x)} \right) dx = \int p(x) \log \left( \frac{p(x)}{q(x)} \right) dx \]

\( f(x) = -\log x \)

\[ D_f(P \| Q) = \int q(x) \left( -\log \left( \frac{p(x)}{q(x)} \right) \right) dx = \int q(x) \log \left( \frac{q(x)}{p(x)} \right) dx \]

\( f(x) = (x - 1)^2 \)

\[ D_f(P \| Q) = \int q(x) \left( \frac{p(x)}{q(x)} - 1 \right)^2 dx = \int \frac{(p(x) - q(x))^2}{q(x)} dx \]
Fenchel Conjugate

- Every convex function $f$ has a conjugate function $f^*$

$$f^*(t) = \max_{x \in \text{dom}(f)} \{xt - f(x)\}$$

$x$ is given, $t$ is variable

\[ D_f(P||Q) = \int_x q(x)f\left(\frac{p(x)}{q(x)}\right)dx \]

$f$ is convex, $f(1) = 0$
Fenchel Conjugate

• Every convex function $f$ has a conjugate $f^*$, and $(f^*)^* = f$

$$f^*(t) = \sup_{x \in \text{dom}(f)} \{ xt - f(x) \}$$

$f(x) = x \log x$

Something like exponential?

$$f^*(t) = \exp(t - 1)$$

$$0.1t - 0.1 \log 0.1$$
$$1t - 0$$
$$10t - 10 \log 10$$
Fenchel Conjugate

• Every convex function $f$ has a conjugate $f^*$, and $(f^*)^* = f$

$$f^*(t) = \sup_{x \in \text{dom}(f)} \{xt - f(x)\}$$

$$f(x) = x \log x \quad \leftrightarrow \quad f^*(t) = \exp(t - 1)$$

$$f^*(t) = \sup_{x \in \text{dom}(f)} \{xt - x \log x\}$$

$$g(x) = xt - x \log x \quad \text{Given } t, \text{ find } x \text{ maximizing } g(x)$$

$$t - \log x - 1 = 0 \quad x = \exp(t - 1)$$

$$f^*(t) = \exp(t - 1) \times t - \exp(t - 1) \times (t - 1)$$
**Connection with GAN**

\[ f^*(t) = \sup_{x \in \text{dom}(f)} \{ xt - f(x) \} \quad \leftrightarrow \quad f(x) = \sup_{t \in \text{dom}(f^*)} \{ xt - f^*(t) \} \]

\[ D_f(P \| Q) = \int q(x) f \left( \frac{p(x)}{q(x)} \right) dx \]

\[ = \int q(x) \left( \sup_{t \in \text{dom}(f^*)} \left\{ \frac{p(x)}{q(x)} t - f^*(t) \right\} \right) dx \]

\[ \geq \int q(x) \left( \max_D \left\{ \frac{p(x)}{q(x)} D(x) - f^*(D(x)) \right\} \right) dx \quad \text{D is a function whose input is x, and output is t} \]

\[ = \max_D \int q(x) \left( \frac{p(x)}{q(x)} D(x) - f^*(D(x)) \right) dx \]

\[ = \max_D \int p(x) D(x) dx + \int q(x) f^*(D(x)) dx \]
Connection with GAN

\[ D_f(P || Q) \approx \max_D \int p(x)D(x)dx + \int q(x)f^*(D(x))dx \]

\[ = \max_D \{E_{x \sim P}[D(x)] + E_{x \sim Q}[f^*(D(x))]\} \]

Samples from P               Samples from Q

\[ D_f(P_{data} || P_G) = \max_D \{E_{x \sim P_{data}}[D(x)] + E_{x \sim P_G}[f^*(D(x))]\} \]

\[ G^* = \arg \min_G \max_D V(G, D) \]

Now you can minimize any f-divergence

Original GAN minimizes something related to JS
\[ D_f(P_{data} \parallel P_G) = \max_D \left\{ E_{x \sim P_{data}}[D(x)] + E_{x \sim P_G}[f^*(D(x))] \right\} \]

<table>
<thead>
<tr>
<th>Name</th>
<th>( D_f(P \parallel Q) )</th>
<th>Generator ( f(u) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total variation</td>
<td>( \frac{1}{2} \int</td>
<td>p(x) - q(x)</td>
</tr>
<tr>
<td>Kullback-Leibler</td>
<td>( \int p(x) \log \frac{p(x)}{q(x)} , dx )</td>
<td>( u \log u )</td>
</tr>
<tr>
<td>Reverse Kullback-Leibler</td>
<td>( \int q(x) \log \frac{q(x)}{p(x)} , dx )</td>
<td>(- \log u )</td>
</tr>
<tr>
<td>Pearson ( \chi^2 )</td>
<td>( \int \frac{(q(x) - p(x))^2}{p(x)} , dx )</td>
<td>( (u - 1)^2 )</td>
</tr>
<tr>
<td>Neyman ( \chi^2 )</td>
<td>( \int \frac{(p(x) - q(x))^2}{q(x)} , dx )</td>
<td>( \frac{(1-u)^2}{u} )</td>
</tr>
<tr>
<td>Squared Hellinger</td>
<td>( \int \left( \sqrt{p(x)} - \sqrt{q(x)} \right)^2 , dx )</td>
<td>( (\sqrt{u} - 1)^2 )</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>( \int (p(x) - q(x)) \log \left( \frac{p(x)}{q(x)} \right) , dx )</td>
<td>((u - 1) \log u)</td>
</tr>
<tr>
<td>Jensen-Shannon</td>
<td>( \frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x) + q(x)} + q(x) \log \frac{2q(x)}{p(x) + q(x)} , dx )</td>
<td>( -(u + 1) \log \frac{1+u}{2} + u \log u )</td>
</tr>
<tr>
<td>Jensen-Shannon-weighted</td>
<td>( \int p(x) \log \frac{\pi p(x) + (1-\pi)q(x)}{p(x) + q(x)} + (1-\pi)q(x) \log \frac{q(x)}{\pi p(x) + (1-\pi)q(x)} , dx )</td>
<td>( \pi u \log u - (1 - \pi + \pi u) \log(1 - \pi + \pi u) )</td>
</tr>
<tr>
<td>GAN</td>
<td>( \int p(x) \log \frac{2p(x)}{p(x) + q(x)} + q(x) \log \frac{2q(x)}{p(x) + q(x)} , dx )</td>
<td>( u \log u - (u + 1) \log(u + 1) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Conjugate ( f^*(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total variation</td>
<td>( t )</td>
</tr>
<tr>
<td>Kullback-Leibler (KL)</td>
<td>( \exp(t - 1) )</td>
</tr>
<tr>
<td>Reverse KL</td>
<td>( -1 - \log(-t) )</td>
</tr>
<tr>
<td>Pearson ( \chi^2 )</td>
<td>( \frac{1}{4} t^2 + t )</td>
</tr>
<tr>
<td>Neyman ( \chi^2 )</td>
<td>( 2 - 2\sqrt{1-t} )</td>
</tr>
<tr>
<td>Squared Hellinger</td>
<td>( \frac{t}{1-t} ) W(e^{1-t}) + \frac{1}{W(e^{1-t})} + t - 2</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>( -\log(2 - \exp(t)) )</td>
</tr>
<tr>
<td>Jensen-Shannon</td>
<td>( (1-\pi) \log \frac{1-\pi}{1-\pi e^{t/\pi}} )</td>
</tr>
<tr>
<td>Jensen-Shannon-weighted</td>
<td>( -\log(1 - \exp(t)) )</td>
</tr>
</tbody>
</table>

Using the f-divergence you like 😊

Double-loop v.s. Single-step

\[ G^* = \arg \min_G \max_D V(G, D) \]

- Original paper of GAN: double-loop algorithm
  - In each iteration
    - Given a generator \( G^t, D^t = \arg \max_D V(G^t, D) \)
    - Update the parameters \( \theta_D \) many times to find \( D^t \)
    - Update generator once:
      - \( \theta_{G}^{t+1} \leftarrow \theta_{G}^{t} + \eta \nabla_{\theta_G} V(\theta_{G}^{t}, \theta_{D}^{t}) \)

- Paper of f-GAN: Single-step algorithm
  - In each iteration
    - \( \theta_{D}^{t+1} \leftarrow \theta_{D}^{t} - \eta \nabla_{\theta_D} V(\theta_{G}^{t}, \theta_{D}^{t}) \)
    - \( \theta_{G}^{t+1} \leftarrow \theta_{G}^{t} + \eta \nabla_{\theta_G} V(\theta_{G}^{t}, \theta_{D}^{t}) \)

Inner Loop

Outer Loop

One Backpropogation
### Experimental Results

- Approximate a mixture of Gaussians

<table>
<thead>
<tr>
<th>train \ test</th>
<th>KL</th>
<th>KL-rev</th>
<th>JS</th>
<th>Jeffrey</th>
<th>Pearson</th>
</tr>
</thead>
<tbody>
<tr>
<td>KL</td>
<td>0.2808</td>
<td>0.3423</td>
<td>0.1314</td>
<td>0.5447</td>
<td>0.7345</td>
</tr>
<tr>
<td>KL-rev</td>
<td>0.3518</td>
<td>0.2414</td>
<td>0.1228</td>
<td>0.5794</td>
<td>1.3974</td>
</tr>
<tr>
<td>JS</td>
<td>0.2871</td>
<td>0.2760</td>
<td>0.1210</td>
<td>0.5260</td>
<td>0.92160</td>
</tr>
<tr>
<td>Jeffrey</td>
<td>0.2869</td>
<td>0.2975</td>
<td>0.1247</td>
<td>0.5236</td>
<td>0.8849</td>
</tr>
<tr>
<td>Pearson</td>
<td>0.2970</td>
<td>0.5466</td>
<td>0.1665</td>
<td>0.7085</td>
<td>0.648</td>
</tr>
</tbody>
</table>
W-GAN

Using Earth Mover’s Distance
(Wasserstein Distance)

Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein GAN, arXiv prepring, 2017
Earth Mover’s Distance

• Considering one distribution $P$ as a pile of earth, and another distribution $Q$ as the target
• The average distance the earth mover has to move the earth?

$W(P, Q) = d$
Earth Mover’s Distance

If there are many possible “moving plan”, using the one with the smallest average distance to define the earth mover’s distance.

Source of image: https://vincentherrmann.github.io/blog/wasserstein/
A “moving plan” is a matrix. The value of the element is the amount of earth from one position to another.

Average distance of a plan $\gamma$:

$$B(\gamma) = \sum_{x_g, x_d} \gamma(x_g, x_d) \| x_g - x_d \|$$

Earth Mover’s Distance:

$$W(P_G, P_{data}) = \min_{\gamma \in \Pi} B(\gamma)$$

The best plan
Earth Mover’s Distance

\[ JS(P_{G_0}, P_{data}) = \log 2 \]
\[ W(P_{G_0}, P_{data}) = d_0 \]

\[ JS(P_{G_50}, P_{data}) = \log 2 \]
\[ W(P_{G_50}, P_{data}) = d_{50} \]

\[ JS(P_{G_{100}}, P_{data}) = 0 \]
\[ W(P_{G_{100}}, P_{data}) = 0 \]
Back to the GAN framework

\[ W(P_{data}, P_G) \]

\[ = \max_{D \in 1-Lipschitz} \left\{ E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)] \right\} \]

**Lipschitz Function**

\[ \|f(x_1) - f(x_2)\| \leq K\|x_1 - x_2\| \]

\( K=1 \) for "1 – Lipschitz"

\[ D_f(P_{data} \| P_G) \]

\[ = \max_D \left\{ E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[f^*(D(x))] \right\} \]
Back to the GAN framework

\[ W(P_{\text{data}}, P_G) \]

\[ = \max_{D \in 1-Lipschitz} \left\{ E_{x \sim P_{\text{data}}} [D(x)] - E_{x \sim P_G} [D(x)] \right\} \]

\[ D(x_1) = +\infty \quad \text{and} \quad D(x_1) = -\infty \]

\[ k + d \]

\[ k \]

Blue: D(x) for original GAN
Green: D(x) for WGAN

\[ \|f(x_1) - f(x_2)\| \leq \|x_1 - x_2\| \]

WGAN will provide gradient to push \( P_G \) towards \( P_{\text{data}} \)
Back to the GAN framework

\[
K \quad W(P_{data}, P_G) = \max_{D \in 1-\text{Lipschitz}} \left\{ E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)] \right\}
\]

How to use gradient descent to optimize this equation?

Clip all the weights \( w \) between \( c \) and \(-c\)

After parameter update, if \( w > c \), then \( w = c \); if \( w < -c \), then \( w = -c \)

We only ensure that

\[
\|f(x_1) - f(x_2)\| \leq K \|x_1 - x_2\|
\]

For some \( K \)
Algorithm of WGAN

Repeat k times

Learning D

- In each training iteration:
  - Sample m examples \( \{x^1, x^2, \ldots, x^m\} \) from data distribution \( P_{data}(x) \)
  - Sample m noise samples \( \{z^1, z^2, \ldots, z^m\} \) from the prior \( P_{prior}(z) \)
  - Obtaining generated data \( \{\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^m\}, \tilde{x}^i = G(z^i) \)
  - Update discriminator parameters \( \theta_d \) to maximize
    \[
    \tilde{W} = \frac{1}{m} \sum_{i=1}^{m} D(x^i) - \frac{1}{m} \sum_{i=1}^{m} D(\tilde{x}^i)
    \]
    \[
    \theta_d \leftarrow \theta_d + \eta \nabla \tilde{W}(\theta_d)
    \]
    Weight clipping

- Sample another m noise samples \( \{z^1, z^2, \ldots, z^m\} \) from the prior \( P_{prior}(z) \)

Learning G

- Update generator parameters \( \theta_g \) to minimize
  \[
  \tilde{W} = \frac{1}{m} \sum_{i=1}^{m} D(x^i) - \frac{1}{m} \sum_{i=1}^{m} D(\tilde{x}^i)
  \]
  \[
  \theta_g \leftarrow \theta_g - \eta \nabla \tilde{W}(\theta_g)
  \]

Initialize \( \theta_d \) for D and \( \theta_g \) for G

Using RMSProp instead of Adam

No sigmoid and log

Weight clipping
DCGAN generator:

W-GAN

GAN

DCGAN generator (no batch normalization, bad structure):

W-GAN

GAN

MLP generator:

W-GAN

GAN

https://arxiv.org/abs/1701.07875
Improved W-GAN

Improved WGAN

$$W(P_{data}, P_G)$$

$$= \max_{D \in \text{1-Lipschitz}} \left\{ E_{x \sim P_{data}} [D(x)] - E_{x \sim P_G} [D(x)] \right\}$$

Gradient Penalty

$$-\lambda E_{x \sim P_{inter}} \left[ (\|\nabla_x D(x)\| - 1)^2 \right]$$
Improved WGAN

https://arxiv.org/abs/1704.00028
Improved WGAN

\[ \|f(x_1) - f(x_2)\| \leq \|x_1 - x_2\| \]

1-lipschitz function \iff \text{The norm of the gradients smaller than 1}

\[ W(P_{data}, P_G) = \max_{D \in 1-\text{Lipschitz}} \left\{ E_{x \sim P_{data}} [D(x)] - E_{x \sim P_G} [D(x)] \right\} \]

\[ \approx \max_D \left\{ E_{x \sim P_{data}} [D(x)] - E_{x \sim P_G} [D(x)] \right\} \]

\[ -\lambda \int_x \max(0, \|\nabla_x D(x)\| - 1) dx \]

\[ -\lambda E_{x \sim P_{inter}} \left[ \max(0, \|\nabla_x D(x)\| - 1) \right] \]

Force the norm of the gradient smaller than 1

Force the norm of the gradient close to 1
Improved WGAN

\[-\lambda \int_x \max(0, \|\nabla_x D(x)\| - 1) dx\] 
\[-\lambda E_{x \sim P_{\text{inter}}} \left[ \max(0, \|\nabla_x D(x)\| - 1) \right] \}

Only give gradient constraint to the area between $P_{\text{data}}$ and $P_{G}$

Because they influence how $P_{G}$ moves to $P_{\text{data}}$

“Given that enforcing the Lipschitz constraint everywhere is intractable, enforcing it only along these straight lines seems sufficient and experimentally results in good performance.”
\[
\text{Improved WGAN} \quad -\lambda E_{x \sim P_{\text{inter}}} \left[ \max (0, \| \nabla_x D(x) \| - 1) \right] \\
-\lambda E_{x \sim P_{\text{inter}}} \left[ (\| \nabla_x D(x) \| - 1)^2 \right]
\]

One may wonder why we penalize the norm of the gradient for differing from 1, instead of just penalizing large gradients. The reason is that the optimal critic ... actually has gradients with norm 1 almost everywhere under Pr and Pg

(check the proof in the appendix)

Simply penalizing overly large gradients also works in theory, but experimentally we found that this approach converged faster and to better optima.
Algorithm of Improved WGAN

- In each training iteration:
  - Sample m examples \( \{x^1, x^2, \ldots, x^m\} \) from data distribution \( P_{data}(x) \)
  - Sample m noise samples \( \{z^1, z^2, \ldots, z^m\} \) from the prior \( P_{prior}(z) \)
  - Obtaining generated data \( \{\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^m\} \), \( \tilde{x}^i = G(z^i) \)
  - Sampling \( \{\varepsilon^1, \varepsilon^2, \ldots, \varepsilon^m\} \) from U[0-1]

\[ \hat{x}^i \leftarrow \varepsilon^i x^i + (1 - \varepsilon^i) \tilde{x}^i \]

- Update discriminator parameters \( \theta_d \) to maximize
  - \( W' = \frac{1}{m} \sum_{i=1}^{m} D(x^i) - \frac{1}{m} \sum_{i=1}^{m} D(\tilde{x}^i) \)
  - \( \theta_d \leftarrow \theta_d + \eta \nabla W'(\theta_d) \)
  - \( + \lambda \sum_{i=1}^{m} (\|\nabla_{\hat{x}} D(\hat{x})\| - 1)^2 \)

Only modifying the discriminator part

Learning D
Repeat k times
Back to Adam
DCGAN
G: CNN, D: CNN

LSGAN
G: CNN (no normalization), D: CNN (no normalization)

WGAN
Original
G: CNN (tanh), D: CNN (tanh)

Improved

DCGAN
G: MLP, D: CNN

LSGAN
G: CNN (bad structure), D: CNN

Original WGAN
G: 101 layer, D: 101 layer

Improved WGAN
Sentence Generation

• good bye.

Consider this matrix as an “image”
Sentence Generation

- Real sentence
  - A binary classifier can immediately find the difference.

- Generated
  - JS divergence always 0
  - WGAN is helpful

Can never be 1-of-N
Improved WGAN successfully generating sentences

WGAN with gradient penalty

Busino game camerate spent odea In the bankaway of smarling the Singers May, who kill that imvic Keray Pents of the same Reagan D Manging include a tudancs shat "His Zuith Dudget, the Denmben In during the Utational questio Divos from The ’ noth ronkies of She like Monday, of macunser S The investor used ty the present A papees are cointry congress oo A few year inom the group that s He said this syenn said they wan As a world 1 88 ,for Autouries Foand, th Word people car, Il High of the upseader homing pull The guipe is worly move dogsfor The 1874 incidested he could be The allo tooks to security and c Solice Norkedin pring in since ThiS record ( 31.) UBS ) and Ch It was not the annuas were plogr This will be us, the ect of DAN These leaded as most-worsd p2 a0 The time I paid0a South Cubry i Dour Fraps higs it was these del This year out howneed allowed lo Kaulna Seto consficates to repor A can teal, he was schoon news In th 200. Pesish picriers rega Konney Panice rimiber the teami The new centuct cut Denester of The near, had been one injustie The incescion to week to shorte The company the high product of 20 - The time of accomplishment, wh John WVuderenson seqiivic spends A ceetens in indestredly the Wat
感謝李仲翊同學提供實驗結果
輸出32個字 (包含標點)

• 升雲白遲丹齋取，此酒新巷市入頭。黃道故海歸中後，不驚入得韻子門。
• 據口容章蕃翎翎，邦貸無遊隔將毬。外蕭曾臺遶出畧，此計推上呂天夢。
• 新來賓伎泉，手雪泓臺蓑。曾子花路魏，不謀散薦船。
• 功持牧度機邀爭，不躚官嬉牧涼散。不迎白旅今掩冬，盡蘸金祗可停。
• 玉十洪沄爭春風，溪子風佛挺橫鞋。盤盤稅焰先花齋，誰過飄鶴一丞幢。
• 海人依野庇，為阻例沉迥。座花不佐樹，弟闌十名儂。
• 入維當興日世瀕，不評皺。頭醉空其杯，駸園凋送頭。
• 鉢笙動春枝，寶叅潔長知。官為宻爛去，絆粒薛一靜。
• 吾涼腕不楚，縱先待旅知。楚人縱酒待，一蔓飄聖猜。
• 折幕故瘻應韻子，徑頭霜瓊老徑徑。尚錯春鐫熊悽梅，去吹依能九將香。
• 通可矯目鸚須浄，丹逅挐花一抵嫖。外子當日中前醒，迎日幽筆鉤弧前。
• 庭愛四樹人庭好，無衣服仍繡秋州。更怯風流欲鴆雲，帛陽舊據靑亭儷。
More about Discrete Output

• SeqGAN
  • Lantao Yu, Weinan Zhang, Jun Wang, Yong Yu, SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient, AAAI, 2017

• Boundary seeking GAN

• Gumbel-Softmax
  • Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, Yoshua Bengio, Maximum-Likelihood Augmented Discrete Generative Adversarial Networks, arXiv preprint, 2017
Conditional GAN
Conditional GAN

- Text to image by traditional supervised learning

\[ c^1: \text{a dog is running} \quad \hat{x}^1: \]
\[ c^2: \text{a bird is flying} \quad \hat{x}^2: \]

Text: “train”

A blurry image!
Conditional GAN

Prior distribution $z \rightarrow G \rightarrow \text{train} \rightarrow x = G(c,z)$

It is a distribution.
Approximate the distribution below

Text: “train”

Target of NN output

A blurry image!
Conditional GAN

Prior distribution \( z \)

\[ x = G(c, z) \]

May generate \( x \) not related to \( c \)

Positive example: \((\text{train}, \text{train})\)

Negative example: \((\text{train}, \text{Image})\) (\(\text{cat}, \text{Image}\))
Image-to-image Translation

\[ x = G(c, z) \]
Image-to-image Translation

- Experimental results

Testing:

It is blurry because it is the average of several images.
Image-to-image

- Experimental results

Testing:

input  close  GAN  GAN + close
Unpaired Transformation - Cycle GAN, Disco GAN

Transform an object from one domain to another **without pair data**

- **photo** → **van Gogh**

**Domain X**

**Domain Y**

- **Monet** → **photo**
- **Zebras** → **Horses**
- **Summer** → **Winter**

- **photo** → **Monet**
- **horse** → **zebra**
- **summer** → **winter**
**Cycle GAN**

https://arxiv.org/abs/1703.10593
https://junyanz.github.io/CycleGAN/

Domain X → \( G_{X \rightarrow Y} \) → Domain Y

Become similar to domain Y

Not what we want

\( D_Y \) → 1/0

Input image belongs to domain Y or not
Cycle GAN

Domain X

Domain Y

\[ G_{X \rightarrow Y} \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow G_{Y \rightarrow X} \]

as close as possible

Input image belongs to domain Y or not

\[ D_Y \rightarrow 1/0 \]

Lack of information for reconstruction

1. Input image belongs to domain Y.
2. Processed image through \( G_{X \rightarrow Y} \) to \( G_{Y \rightarrow X} \).
3. \( D_Y \) determines if the input image belongs to domain Y.
Cycle GAN

As close as possible

\[ G_{X \rightarrow Y} \]

\[ D_X \]

\[ 1/0 \]

\[ G_{Y \rightarrow X} \]

\[ D_Y \]

\[ 1/0 \]

\[ \text{belongs to domain } X \text{ or not} \]

\[ \text{belongs to domain } Y \text{ or not} \]

As close as possible
Unpaired Transformation – 真人動畫化

• http://qiita.com/Hi-king/items/8d36d9029ad1203aac55

http://www.iis.ee.ic.ac.uk/cxiong/database.html

把真人頭像變成動漫人物
Unpaired Transformation – 真人動畫化

• Using the code: https://github.com/Hi-king/kawaii_creator
• It is not cycle GAN, Disco GAN
Evaluation

Likelihood

Prior Distribution

Generator $P_G$

$x_i$

Log Likelihood: $L = \frac{1}{N} \sum_i \log P_G(x^i)$

We cannot compute $P_G(x^i)$. We can only sample from $P_G$. 

\* real data (not observed during training)

\* generated data
Likelihood
- Kernel Density Estimation

• Estimate the distribution of $P_G(x)$ from sampling

Now we have an approximation of $P_G$, so we can compute $P_G(x^i)$ for each real data $x^i$

Then we can compute the likelihood.
Likelihood - Kernel Density Estimation

• How many samples?

• Weird results?

<table>
<thead>
<tr>
<th>Model</th>
<th>Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBN</td>
<td>138</td>
</tr>
<tr>
<td>GAN</td>
<td>225</td>
</tr>
<tr>
<td>True Distribution</td>
<td>243</td>
</tr>
<tr>
<td>K-means</td>
<td>313</td>
</tr>
</tbody>
</table>
Likelihood v.s. Quality

• Low likelihood, high quality?

Considering a model generating images from training set ……

• High likelihood, low quality?

\[
L = \frac{1}{N} \sum_{i} \log P_G(x^i) = -\log 100 + \frac{1}{N} \sum_{i} \log P_G(x^i)
\]

\[
P_G(x^i) = 0
\]
Evaluate by Other Networks

Well-trained CNN

\[ x \rightarrow P(y|x) \]

Lower entropy means higher visual quality

\[ \frac{1}{N} \sum_n P(y^n|x^n) \]

High entropy means high variety
In order to estimate both the missing modes and the sample qualities in our experiments, we used several different metrics for different experiments instead of human annotators.

The inception score (Salimans et al., 2016) was considered as a good assessment for sample quality from a labelled dataset:

\[
\exp (\mathbb{E}_x KL(p(y|x)||p^*(y)))
\]

Where \(x\) denotes one sample, \(p(y|x)\) is the softmax output of a trained classifier of the labels, and \(p^*(y)\) is the overall label distribution of generated samples. The intuition behind this score is that a strong classifier usually has a high confidence for good samples. However, the inception score is sometimes not a good metric for our purpose. Assume a generative model that collapse to a very bad image. Although the model is very bad, it can have a perfect inception score, because \(p(y|x)\) can have a high entropy and \(p^*(y)\) can have a low entropy. So instead, for labelled datasets, we propose another assessment for both visual quality and variety of samples, the MODE score:

\[
\exp (\mathbb{E}_x KL(p(y|x)||p(y)) - KL(p^*(y)||p(y)))
\]
Evaluate by Other Networks - Inception Score

• Improved W-GAN
K-Nearest Neighbor

- Using k-nearest neighbor to check whether the generator generates new objects
Missing Mode

Discriminator always knows they are real with high confidence

Missing anything?

Energy-based GAN

Another point of view
Original GAN

- Role of discriminator: lead the generator

When the data distribution and generated distribution is the same, the discriminator will be useless (flat).
Original GAN

The discriminator is flat in the end.

Source: https://www.youtube.com/watch?v=ebMei6bYeWw (credit: Benjamin Striner)
Energy-based Model

• We want to find an evaluation function $F(x)$
  • Input: object $x$ (e.g. images), output: scalar (how good $x$ is)
    • Real $x$ has high $F(x)$
    • $F(x)$ can be a network
• We can find good $x$ by $F(x)$:
  • Generate $x$ with large $F(x)$

• How to find $F(x)$?

![Diagram](image-url)
Energy-based GAN

- We want to find an evaluation function \( F(x) \)
- How to find \( F(x) \)?

In the end ......
Energy-based GAN

• Preview: Framework of structured learning (Energy-based Model)
  • ML Lecture 21: Structured Learning - Introduction
    • https://www.youtube.com/watch?v=5OYu0vxXEv8
  • ML Lecture 22: Structured Learning - Linear Model
    • https://www.youtube.com/watch?v=HfPw40JPays
  • ML Lecture 23: Structured Learning - Structured SVM
    • https://www.youtube.com/watch?v=YjvGVVrCrhQ
  • ML Lecture 24: Structured Learning - Sequence Labeling
    • https://www.youtube.com/watch?v=o9FPSqobMys