Generative Adversarial Network: a Brief Introduction

Lili Mou
doublepower.mou@gmail.com
Outline

- Generative adversarial net
- Conditional generative adversarial net
- Deep generative image models using Laplacian pyramid of adversarial networks
Deep generative models are less impactful than deep discriminative models, because...

- Of the difficulty of approximating many intractable probabilistic computations that arise in maximum likelihood estimation and related strategies.
- Of the difficulty of leveraging the benefits of piecewise linear units in the generative context.
- (My point of view) Generative problems are much more difficult than discriminative ones.
A Game Theory Perspective

• Two agents:
  – Generative model: Generate new samples that are as similar as the data
  – Discriminative model: Distinguish samples in disguise

• Each agent takes a step in turn
Objective of GAN

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

- **G(z):** A generated sample from distribution z
- **D(x) =** Estimated (by D) prob. that x is a real data sample
  - D(x)=1: D regards x as a training sample w.p.1
  - D(x)=0: D regards x as a generative sample w.p.1
- The relationship with traditional minimax problem
  - In a two-agent WuZi chess, V evaluates how bad the current position is to me.
Objective of GAN

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

• G(z): A generated sample from distribution z
• D(x) = Estimated (by D) prob. that x is a real data sample
  - D(x)=1: D regards x as a training sample w.p.1
  - D(x)=0: D regards x as a generative sample w.p.1

• The relationship with traditional minimax problem
  - In a two-agent WuZi chess, V evaluates how bad the current position is to me.

Mathematicians seem to be pessimistic creatures who think in terms of losses. Decision theorists in economics and business talk instead in terms of gains (utility).
Objective of GAN

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

- \(G(z)\): A generated sample from distribution \(z\)
- \(D(x) = \text{Estimated (by } D\text{) prob. that } x \text{ is a real data sample}\)
  - \(D(x)=1\): \(D\) regards \(x\) as a training sample w.p.1
  - \(D(x)=0\): \(D\) regards \(x\) as a generative sample w.p.1
- The relationship with traditional minimax problem
  - In a two-agent WuZi chess, \(V\) evaluates how bad the current position is to me.
  - Adversary's goal: maximize \(V\)
  - My Goal: minimize \(\max \ V\)
Objective of GAN

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

Algorithm

\[ \max_D V(D, G) \]

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

- Choose $k = 1$. Recall CD-k
- \[ \text{minimize } \log(1 - D(G(z))) \iff \text{maximize } \log D(G(z)) \]
- The latter yields a larger gradient especially at beginning steps

\[ \text{for number of training iterations do} \]
\[ \text{for } k \text{ steps do} \]
  - 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 $p_{\text{data}}(x)$.
  - Update the discriminator by ascending its stochastic gradient:
    \[ \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D(x^{(i)}) + \log \left( 1 - D\left(G\left(z^{(i)}\right)\right) \right) \right]. \]
\[ \text{end for} \]
- 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:
  \[ \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left( 1 - D\left(G\left(z^{(i)}\right)\right) \right). \]
\[ \text{end for} \]
Intuitive Explanation

Figure 1: Generative adversarial nets are trained by simultaneously updating the discriminative distribution \( D \), blue, dashed line so that it discriminates between samples from the data generating distribution (black, dotted line) \( p_x \) from those of the generative distribution \( p_g \) (G) (green, solid line). The lower horizontal line is the domain from which \( z \) is sampled, in this case uniformly. The horizontal line above is part of the domain of \( x \). The upward arrows show how the mapping \( x = G(z) \) imposes the non-uniform distribution \( p_g \) on transformed samples. \( G \) contracts in regions of high density and expands in regions of low density of \( p_g \). (a) Consider an adversarial pair near convergence: \( p_g \) is similar to \( p_{\text{data}} \) and \( D \) is a partially accurate classifier. (b) In the inner loop of the algorithm \( D \) is trained to discriminate samples from data, converging to \( D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \). (c) After an update to \( G \), gradient of \( D \) has guided \( G(z) \) to flow to regions that are more likely to be classified as data. (d) After several steps of training, if \( G \) and \( D \) have enough capacity, they will reach a point at which both cannot improve because \( p_g = p_{\text{data}} \). The discriminator is unable to differentiate between the two distributions, i.e. \( D(x) = \frac{1}{2} \).
Intuitive Explanation

Discriminative distribution, i.e., how likely a sample is in training set.

Data distribution

Generative distribution

Two-point distribution

Figure 1: Generative adversarial nets are trained by simultaneously updating the discriminative distribution ($D$, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_x$ from those of the generative distribution $p_g$ ($G$) (green, solid line). The lower horizontal line is the domain from which $z$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $x$. The upward arrows show how the mapping $x = G(z)$ imposes the non-uniform distribution $p_g$ on transformed samples. $G$ contracts in regions of high density and expands in regions of low density of $p_g$. (a) Consider an adversarial pair near convergence: $p_g$ is similar to $p_{\text{data}}$ and $D$ is a partially accurate classifier. (b) In the inner loop of the algorithm $D$ is trained to discriminate samples from data, converging to $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}$. (c) After an update to $G$, gradient of $D$ has guided $G(z)$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if $G$ and $D$ have reached a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminative distribution, i.e. $D(x) = \frac{1}{2}$.
Intuitive Explanation

Discriminative distribution, i.e., how likely a sample is in training set

Data distribution

Generative distribution

Figure 1: Generative adversarial nets are trained by simultaneously updating the discriminative distribution ($D$, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_x$ from those of the generative distribution $p_g$ ($G$) (green, solid line). The lower horizontal line is the domain from which $z$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $x$. The upward arrows show how the mapping $x = G(z)$ imposes the non-uniform distribution $p_g$ on transformed samples. $G$ contracts in regions of high density and expands in regions of low density of $p_g$. (a) Consider an adversarial pair near convergence: $p_g$ is similar to $p_{data}$ and $D$ is a partially accurate classifier. (b) In the inner loop of the algorithm $D$ is trained to discriminate samples from data, converging to $D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$. (c) After an update to $G$, gradient of $D$ has guided $G(z)$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if $G$ and $D$ have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{data}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(x) = \frac{1}{2}$. 
Figure 1: Generative adversarial nets are trained by simultaneously updating the discriminative distribution ($D$, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_x$ from those of the generative distribution $p_g$ (G) (green, solid line). The lower horizontal line is the domain from which $z$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $x$. The upward arrows show how the mapping $x = G(z)$ imposes the non-uniform distribution $p_g$ on transformed samples. $G$ contracts in regions of high density and expands in regions of low density of $p_g$. (a) Consider an adversarial pair near convergence: $p_g$ is similar to $p_{data}$ and $D$ is a partially accurate classifier. (b) In the inner loop of the algorithm $D$ is trained to discriminate samples from data, converging to $D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$. (c) After an update to $G$, gradient of $D$ has guided $G(z)$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if $G$ and $D$ have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{data}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(x) = \frac{1}{2}$.
Theoretical Analysis

4.1 Global Optimality of $p_g = p_{\text{data}}$

We first consider the optimal discriminator $D$ for any given generator $G$.

**Proposition 1.** For $G$ fixed, the optimal discriminator $D$ is

$$D^*_G(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}$$

**Proof.** The training criterion for the discriminator $D$, given any generator $G$, is to maximize the quantity $V(G, D)$

$$V(G, D) = \int_x p_{\text{data}}(x) \log(D(x)) dx + \int_z p_z(z) \log(1 - D(g(z))) dz$$

$$= \int_x p_{\text{data}}(x) \log(D(x)) + p_g(x) \log(1 - D(x)) dx$$

(3)

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \rightarrow a \log(y) + b \log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $\text{Supp}(p_{\text{data}}) \cup \text{Supp}(p_g)$, concluding the proof. \qed
Global minimum: \( p_g = p_{data} \)

**Theorem 1.** The global minimum of the virtual training criterion \( C(G) \) is achieved if and only if \( p_g = p_{data} \). At that point, \( C(G) \) achieves the value \(-\log 4\).

**Proof.** For \( p_g = p_{data} \), \( D_G^*(x) = \frac{1}{2} \), (consider Eq. 2). Hence, by inspecting Eq. 4 at \( D_G^*(x) = \frac{1}{2} \), we find \( C(G) = \log \frac{1}{2} + \log \frac{1}{2} = -\log 4 \). To see that this is the best possible value of \( C(G) \), reached only for \( p_g = p_{data} \), observe that

\[
\mathbb{E}_{x \sim p_{data}} [-\log 2] + \mathbb{E}_{x \sim p_g} [-\log 2] = -\log 4
\]

and that by subtracting this expression from \( C(G) = V(D_G^*, G) \), we obtain:

\[
C(G) = -\log(4) + KL \left( p_{data} \left\| \frac{p_{data} + p_g}{2} \right\| \right) + KL \left( p_g \left\| \frac{p_{data} + p_g}{2} \right\| \right)
\]

(5)

where \( KL \) is the Kullback–Leibler divergence. We recognize in the previous expression the Jensen–Shannon divergence between the model's distribution and the data generating process:

\[
C(G) = -\log(4) + 2 \cdot JSD \left( p_{data} \left\| p_g \right\| \right)
\]

(6)

Since the Jensen–Shannon divergence between two distributions is always non-negative, and zero iff they are equal, we have shown that \( C^* = -\log(4) \) is the global minimum of \( C(G) \) and that the only solution is \( p_g = p_{data} \), i.e., the generative model perfectly replicating the data distribution.

- The cost is convex in \( p_g \).
- But in practice, we always train a parametric model \( G(z; \theta_g) \)
Outline

• Generative adversarial net
• Conditional generative adversarial net
• Deep generative image models using Laplacian pyramid of adversarial networks
Conditional Generative Adversarial Nets
## Multi-Modal Generation

<table>
<thead>
<tr>
<th>User tags + annotations</th>
<th>Generated tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>montanha, trem, inverno, frio, people, male, plant life, tree, structures, transport, car</td>
<td>taxi, passenger, line, transportation, railway station, passengers, railways, signals, rail, rails</td>
</tr>
<tr>
<td>food, raspberry, delicious, homemade</td>
<td>chicken, fattening, cooked, peanut, cream, cookie, house made, bread, biscuit, bakes</td>
</tr>
<tr>
<td>water, river</td>
<td>creek, lake, along, near, river, rocky, treeline, valley, woods, waters</td>
</tr>
<tr>
<td>people, portrait, female, baby, indoor</td>
<td>love, people, posing, girl, young, strangers, pretty, women, happy, life</td>
</tr>
</tbody>
</table>

Table 2: Samples of generated tags

*Image convnet, word embeddings pretrained*

- **G:**
- **D:**
## Multi-Modal Generation

<table>
<thead>
<tr>
<th>User tags + annotations</th>
<th>Generated tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>montanha, trem, inverno, frio, people, male, plant life, tree, structures, transport, car</td>
<td>taxi, passenger, line, transportation, railway station, passengers, railways, signals, rail, rails</td>
</tr>
<tr>
<td>food, raspberry, delicious, homemade</td>
<td>chicken, fattening, cooked, peanut, cream, cookie, house made, bread, biscuit, bakes</td>
</tr>
<tr>
<td>water, river</td>
<td>creek, lake, along, near, river, rocky, treeline, valley, woods, waters</td>
</tr>
<tr>
<td>people, portrait, female, baby, indoor</td>
<td>love, people, posing, girl, young, strangers, pretty, women, happy, life</td>
</tr>
</tbody>
</table>

Table 2: Samples of generated tags

### Image convnet, word embeddings pretrained

- **G**: Gaussian noise + image features → regression over word embedding
- **D**: image + embedding → sigmoid
Outline

- Generative adversarial net
- Conditional generative adversarial net
- Deep generative image models using Laplacian pyramid of adversarial networks
Laplacian Pyramid

- Upsample: an $i \times i$ image $\rightarrow$ an $2i \times 2i$ image
- Downsampling: an $i \times i$ image $\rightarrow$ an $i/2 \times i/2$ image
Figure 2: The training procedure for our LAPGAN model. Starting with a 64x64 input image \( I \) from our training set (top left): (i) we take \( I_0 = I \) and blur and downsample it by a factor of two (red arrow) to produce \( I_1 \); (ii) we upsample \( I_1 \) by a factor of two (green arrow), giving a low-pass version \( l_0 \) of \( I_0 \); (iii) with equal probability we use \( l_0 \) to create either a real or a generated example for the discriminative model \( D_0 \). In the real case (blue arrows), we compute high-pass \( h_0 = I_0 - l_0 \) which is input to \( D_0 \) that computes the probability of it being real vs generated. In the generated case (magenta arrows), the generative network \( G_0 \) receives as input a random noise vector \( z_0 \) and \( l_0 \). It outputs a generated high-pass image \( \tilde{h}_0 = G_0(z_0, l_0) \), which is input to \( D_0 \). In both the real/generated cases, \( D_0 \) also receives \( l_0 \) (orange arrow). Optimizing Eqn. 2, \( G_0 \) thus learns to generate realistic high-frequency structure \( \tilde{h}_0 \) consistent with the low-pass image \( l_0 \). The same procedure is repeated at scales 1 and 2, using \( I_1 \) and \( I_2 \). Note that the models at each level are trained independently. At level 3, \( I_3 \) is an 8x8 image, simple enough to be modeled directly with a standard GANs \( G_3 \) & \( D_3 \).
Figure 2: The training procedure for our LAPGAN model. Starting with a 64×64 input image $I$ from our training set (top left): (i) we take $I_0 = I$ and blur and downsample it by a factor of two (red arrow) to produce $I_1$; (ii) we upsample $I_1$ by a factor of two (green arrow), giving a low-pass version $l_0$ of $I_0$; (iii) with equal probability we use $l_0$ to create either a real or a generated example for the discriminative model $D_0$. In the real case (blue arrows), we compute high-pass $h_0 = I_0 - l_0$ which is input to $D_0$ that computes the probability of it being real vs generated. In the generated case (magenta arrows), the generative network $G_0$ receives as input a random noise vector $z_0$ and $l_0$. It outputs a generated high-pass image $\tilde{h}_0 = G_0(z_0, l_0)$, which is input to $D_0$. In both the real/generated cases, $D_0$ also receives $l_0$ (orange arrow). Optimizing Eqn. 2, $G_0$ thus learns to generate realistic high-frequency structure $\tilde{h}_0$ consistent with the low-pass image $l_0$. The same procedure is repeated at scales 1 and 2, using $I_1$ and $I_2$. Note that the models at each level are trained independently. At level 3, $I_3$ is an 8×8 image, simple enough to be modeled directly with a standard GANs $G_3$ & $D_3$. 
Figure 2: The training procedure for our LAPGAN model. Starting with a 64×64 input image $I$ from our training set (top left): (i) we take $I_0 = I$ and blur and downsample it by a factor of two (red arrow) to produce $I_1$; (ii) we upsample $I_1$ by a factor of two (green arrow), giving a low-pass version $l_0$ of $I_0$; (iii) with equal probability we use $l_0$ to create either a real or a generated example for the discriminative model $D_0$. In the real case (blue arrows), we compute high-pass $h_0 = I_0 - l_0$ which is input to $D_0$ that computes the probability of it being real vs generated. In the generated case (magenta arrows), the generative network $G_0$ receives as input a random noise vector $z_0$ and $l_0$. It outputs a generated high-pass image $\tilde{h}_0 = G_0(z_0, l_0)$, which is input to $D_0$. In both the real/generated cases, $D_0$ also receives $l_0$ (orange arrow). Optimizing Eqn. 2, $G_0$ thus learns to generate realistic high-frequency structure $\tilde{h}_0$ consistent with the low-pass image $l_0$. The same procedure is repeated at scales 1 and 2, using $I_1$ and $I_2$. Note that the models at each level are trained independently. At level 3, $I_3$ is an 8×8 image, simple enough to be modeled directly with a standard GANs $G_3$ & $D_3$. 
Figure 2: The training procedure for our LAPGAN model. Starting with a 64x64 input image $I$ from our training set (top left): (i) we take $I_0 = I$ and blur and downsample it by a factor of two (red arrow) to produce $I_1$; (ii) we upsample $I_1$ by a factor of two (green arrow), giving a low-pass version $l_0$ of $I_0$; (iii) with equal probability we use $l_0$ to create either a real or a generated example for the discriminative model $D_0$. In the real case (blue arrows), we compute high-pass $h_0 = I_0 - l_0$ which is input to $D_0$ that computes the probability of it being real vs generated. In the generated case (magenta arrows), the generative network $G_0$ receives as input a random noise vector $z_0$ and $l_0$. It outputs a generated high-pass image $\tilde{h}_0 = G_0(z_0, l_0)$, which is input to $D_0$. In both the real/generated cases, $D_0$ also receives $l_0$ (orange arrow). Optimizing Eqn. 2, $G_0$ thus learns to generate realistic high-frequency structure $\tilde{h}_0$ consistent with the low-pass image $l_0$. The same procedure is repeated at scales 1 and 2, using $I_1$ and $I_2$. Note that the models at each level are trained independently. At level 3, $I_3$ is an 8x8 image, simple enough to be modeled directly with a standard GANs $G_3$ & $D_3$. 

Training Each step trained separately
Generation
Results

CC-LAPGAN: Ship

CC-LAPGAN: Truck