Generative Adversarial Networks, and Applications

Outline:

• Generative Models vs Discriminative Models (Background)
• Generative Adversarial Networks (GANs)
• Mathematical Proofs
• GAN Results
• GAN Failures
• Deep Convolutional GAN Architecture (Related Work)
• GAN-based Applications (Related Work)
Generative Models vs Discriminative Models:

- Data: Pixels
- Features: X
- Labels: Y (Zebra, No zebra)
- Discriminative model: model $p(y|x)$.
- Generative model: model $p(x,y)$.
  → Learn $p(y|x)$ indirectly using Bayes rule.

(Bayes rule: $p(y|x) \propto p(x|y)p(y)$)

Discriminative Model:

- Map each sample to feature space.
- Learns decision boundary.
- For each point ($x'$) in feature space:
  - $P(\text{Zebra}|x')+P(\text{No Zebra}|x')=1$
Generative Model:

Learns probability distribution of each class of data:

\[ \int P(x|\text{Zebra}) dx = 1 \]
\[ \int P(x|\text{No Zebra}) dx = 1 \]

Generative Adversarial Networks (GANs):

There is a game between two networks:
- Generative network.
- Discriminative network.

Example: Counterfeit Money.
- G is a crook and is trying to generate fake money.
- D is a teller at the Bank.

G’s goal is to generate samples that D classifies as real.
\[ \rightarrow \] G should learn the underlying distribution of real money.
Input: A real image (x), or a fake image (x' = G(z)).
Output: Probability that input is real.

D’s goal:
• D(x) = 1
• D(G(z)) = 0

G’s goal:
• D(G(z)) = 1

Objective Function:

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

Remember that:
- x: Real image
- G(z): Fake image
- D: Discriminator network
- G: Generative network

 discriminator’s ability to recognize generator samples as being fake
Desired Convergence:

We will prove that the models will approach equilibrium:

- \( D(x) = 0.5 \)
- \( D(G(z)) = 0.5 \)

\( D \) cannot discriminate.

\( G \) will learn underlying distribution of real data. (\( P_{data} \))
Training Algorithm:

From: Generative Adversarial Nets, Goodfellow et al, 2014

Algorithm 1: Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, $k$, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```
for number of training iterations do
    for $k$ steps do
        • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_{\text{noise}}(z)$.
        • 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:
          $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D \left(x^{(i)}\right) + \log \left(1 - D \left(G \left(z^{(i)}\right)\right)\right) \right].$$
    end for
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_{\text{noise}}(z)$.
    • 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.

Training Algorithm:

- The Algorithm takes $k$ steps to optimize for the Discriminative net, $D$, and then one step of optimizing $G$, the generative net based on the outcome from $D$ after above $k$ steps.
- While optimizing for $D$, we work on the entire value function. For $G$, we only use the second term of the value function, as the gradient w.r.t. for the first term is zero.
Training Algorithm:

- It is important to note that we ascend the gradient when optimizing for D, as it is a maximization problem, and descend the gradient when optimizing for G as it is a minimization problem.

- Initially when G is far from optimal, the gradients in second optimization might be very small. Instead, we can ascend the gradient for \( \log(D(G(z))) \) initially.

From: Generative Adversarial Nets, Goodfellow et al, 2014
Global Optimality of $p_g = p_{\text{data}}$

This is done using

- A proposition for finding an optimal discriminator $D$ for a fixed $G$ (Proposition 1)
- A theorem for finding the global minimum for $G$ (Theorem 1)

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

$$D \ast G(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}$$

**Proof for Proposition 1:**

- **Expected Value:**
  $$E[g(x)] = \sum x p(x) g(x) = \int x p(x) g(x) \, dx$$

- **Change of Variable:**
  - If $Y = u(X)$ and $u$ is invertible, then $X = v(Y)$ where $v$ is the inverse of $u$.
  - By using the change of variable rule on the density functions for the distribution of $Y$ to the distribution of $X$, we get:
    $$f_Y(y) = f_X(v(y)) \times |v'(y)|$$

- **Now:**
  $$V(G, D) = E_x - p_{\text{data}}(x) [\log(D(x))] + E_z - p_z(z) [\log(1 - D(G(z)))]$$
  $$= \int_x p_{\text{data}}(x) \log(D(x)) \, dx + \int_z p_z(z) \log(1 - D(G(z))) \, dz$$
  $$= \int_x p_{\text{data}}(x) \log(D(x)) \, dx + p_g(x) \log(1 - D(x)) \, dx$$
Proof for Proposition 1 (Contd’.):

• From the last equation on previous slide:

\[ V(G, D) = \int \left( p_{\text{data}}(x) \log(D(x)) + p_g(x) \log(1 - D(G(x))) \right) dx \]

• For a function \( f: y \rightarrow a \log(y) + b \log(1-y) \), the maximum occurs at \( y = \frac{a}{a+b} \).

• Therefore:

\[ D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \]

Global Optimality of \( p_g=p_{\text{data}} \)

• For a given \( G \), the minmax value function can be defined as:

\[ C(G) = \max D V(G, D) \]

\[ = E_x \sim p_{\text{data}} [\log(D_G^*(x))] + E_x \sim p_g [\log(1 - D_G^*(x))] \]

\[ = E_x \sim p_{\text{data}} [\log\left( \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right)] + E_x \sim p_g [\log\left( \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} \right)] \]

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

- For $p_g = p_{\text{data}}$, $D_G^* = 1/2$; So,

- Also,

$$C(G) = E_x - p_{\text{data}}[-\log 2 + \log \left( \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right)] + E_x - p_g[-\log 2 + \log \left( \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} \right)]$$

$$= -\log(4) + \int_{x \sim p_{\text{data}}(x)} p_{\text{data}}(x) \log \left( \frac{p_{\text{data}}(x)}{(p_{\text{data}}(x) + p_g(x)) / 2} \right) + \int_{x \sim p_g(x)} p_g(x) \log \left( \frac{p_g(x)}{(p_{\text{data}}(x) + p_g(x)) / 2} \right)$$

Proof of Theorem 1 (Contd."):

- KL-Divergence between two probability distributions $p$ and $q$:

$$KL(p \parallel q) = \sum_i p_i \log \left( \frac{p_i}{q_i} \right) = \int p(x) \log \left( \frac{p(x)}{q(x)} \right) dx$$

- From the last equation on previous slide:

$$C(G) = -\log(4) + \int_{x \sim p_{\text{data}}(x)} p_{\text{data}}(x) \log \left( \frac{p_{\text{data}}(x)}{(p_{\text{data}}(x) + p_g(x)) / 2} \right) + \int_{x \sim p_g(x)} p_g(x) \log \left( \frac{p_g(x)}{(p_{\text{data}}(x) + p_g(x)) / 2} \right)$$

$$KL(p_{\text{data}} \parallel \frac{p_{\text{data}} + p_g}{2}) + KL(p_g \parallel \frac{p_{\text{data}} + p_g}{2})$$

- $KL(p \parallel q) \geq 0$ for all $p, q$; equality holds when $p=q$.
- Therefore the minimum of $C(G) = -\log(4)$ obtained when $p_g = p_{\text{data}}$. 
Convergence of Training Algorithm:

• For this, we use the following proposition.

• **Proposition 2**: If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and \( p_g \) is updated to improve the criterion:

\[
E_x - p_{\text{true}}[\log D_g(x)] + E_x - p_g[\log(1 - D_g(x))]
\]

then \( p_g \) converges to \( p_{\text{data}} \).

GAN Failures
Deep Convolutional GAN Architecture (related work)
GAN Results

Failure 1: Mode Collapse

Problem
- Goal of GAN: To generate fake examples imitating real samples
- Easy way of achieving goal: Just generate easy modes (classes).

Failure 1: Mode Collapse (demo)

Image Source: CSE 253 (Winter 2017) Project of Kwonjoon Lee, Andrew Leverentz, Ali Mirzaei, and Andrew Durnford.

Failure 2: Vanishing Gradient

• Problem
  ▪ The representational power (or capacity) between discriminator and generator is not balanced
DCGAN: The First Stabilized GAN (01/07/2016)

Discriminator

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Output Size</th>
<th>Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution (4, 4)</td>
<td>32 x 32 x 64</td>
<td>64</td>
</tr>
<tr>
<td>w/ stride (2, 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LeakyReLU (0.2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subsampling (2, 2)</td>
<td>16 x 16 x 128</td>
<td></td>
</tr>
<tr>
<td>Convolution (4, 4)</td>
<td>8 x 8 x 256</td>
<td></td>
</tr>
<tr>
<td>LeakyReLU (0.2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fully Connected (4<em>4</em>512, 1)</td>
<td>512</td>
<td></td>
</tr>
<tr>
<td>sigmoid</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Generator

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Output Size</th>
<th>Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latent Random Noise</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Fully Connected (100, 4<em>4</em>512)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deconvolution (4, 4)</td>
<td>8 x 8 x 256</td>
<td></td>
</tr>
<tr>
<td>w/ stride (2, 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReLU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upsampling (2, 2)</td>
<td>16 x 16 x 128</td>
<td></td>
</tr>
<tr>
<td>Convolution (4, 4)</td>
<td>32 x 32 x 64</td>
<td></td>
</tr>
<tr>
<td>LeakyReLU (0.2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upsampling (2, 2)</td>
<td>64 x 64 x 3</td>
<td></td>
</tr>
<tr>
<td>Convolution (4, 4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tanh</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

P (input is real)

DCGAN: Results

Bedroom images

Face images

GAN-Based Applications (related work)

- **Inspiration**
  - Vector("King") – Vector("Man") + Vector("Woman") \(\cong\) Vector("Queen")
  - Simple arithmetic operations reveal rich linear structure in representation space (Mikolov et al., 2013)

- **Question & Method**
  - Whether similar structure exists in Z representation (Input) of our Generator
  - e.g. “Smiling Woman” – “Neutral Woman” + “Neutral Man” \(\cong\) “Smiling Man”

- **Difficulties**
  - Results are unstable if working on only single sample, but averaging the Z vector for three exemplars yields consistent and stable generations.

DCGAN: Vector Arithmetic on Face Samples (01/07/2016)
DCGAN: Vector Arithmetic on Face Samples

• Result

![Result Diagram]

DCGAN: Vector Arithmetic on Face Samples

• **Result**
  - “Turn Vector”
  - By adding interpolations to random samples ($Z$) we were able to transform their pose.

![Turn Vector](image)


---

DCGAN: Vector Arithmetic on Face Samples

• **Future**
  - Conditional generative models can learn to convincingly model object attributes like scale, rotation, and position (Dosovitskiy et al., 2014)
  - Further exploring the mentioned vector arithmetic could dramatically reduce the amount of data needed for conditional generative modeling of complex image distributions.
  - *e.g.* Gender + Expression + Race + Age + Pose... = Individual with many details.
Video GAN: (Conditional) Video Generation (09/2016)

- **Video GAN Basic**
  - Generator tries to generate a synthetic video, while Discriminator tries to discriminate synthetic from real videos
  - Structure (Two-stream):


- **Video Generation:**
  - 5000-hour videos from Flickr for training
  - Experiment with four scene categories: golf course, hospital rooms, beaches, and train stations.

- **Result:**
Video GAN: (Conditional) Video Generation

- **Conditional Video GAN Structure**
  - Given a static image $x_0$, extrapolate a video of possible consequent frames.

- **Result**

CycleGAN: Collection Style Transfer (03/30/2017)

- **Image to Image Translation** (11/21/2016)
  - Paired Data
  - Conditional GAN

CycleGAN: Collection Style Transfer (03/30/2017)

**CycleGAN Basics**
- Discovering **special characteristics of each image collection** and how these characteristics could be **translated into the other image collection**, all in the absence of any paired training examples.
- Generator 1: Input $x$ -> Output $G(x)$
- Discriminator: Tell $G(x)$ from $x$
- Generator 2: Input $y$ -> Output $F(y)$
- Discriminator 2: Tell $F(y)$ from $y$
- **Original Two GANs Loss**
- Add a **Cycle Consistency Loss** that encourages $F(G(x)) = x$ and $G(F(y)) = y$
- L1 norm in this case

---

**Objective**
- Minic the style of a whole set of art work, instead of a single piece of art work.
- e.g. Transfer photos(X) to Van Gogh style(Y), rather than just in the style of Starry Night.

**Method**
- Train the model on **landscape photographs downloaded from Flickr(X) and art work from WikiArt(Y)**

---

Some art works of Vincent Van Gogh, from WikiArt

Starry Night, 1889, from WikiArt
• Result


CycleGAN: Other Applications

• Photo generation from paintings
  ▫ Also include an identity loss that encourages identity mapping (photo and painting should not vary much)

CycleGAN: Other Applications

- **Season Transfer**


Thanks!