Semi-supervised GAN

HAMIDREZA MOHEBBI
UMASS BOSTON
10/13/17
Outline

• Unsupervised Learning Problem of GAN
  • Explanation of The Problem by DCGAN

• CatGAN
  • Objective Function
  • Semi-supervised Learning
  • Experimental Results

• Improved GAN
  • Feature Matching
  • Minibatch Discrimination
  • Virtual Batch Normalization
  • Semi-supervised Learning
  • Experimental Results

• Conclusion
Unsupervised Learning Problem of GAN

Main Topics of This Presentation

CatGAN [Springenberg, 2016]

Improv ed GAN [Salimans, 2016]

Lack of Trading Information

Lack of Trading Information Between $G$ and $D$:
CatGAN’s objective function trade-off mutual information between observed examples and their predicted categorical class distribution, against robustness of the classifier to an adversarial generative model.

The Problem Training GAN: Failed to coverage when used to seek for a Nash equilibrium using gradient descent techniques [Goodfellow, 2014].
Explanation of The Problem by DCGAN

Can be thought of as two separate networks
Explanation of The Problem by DCGAN
Explanation of The Problem by DCGAN

Generator $G(.)$
input = random numbers, output = generated image

Uniform noise vector (random numbers)
Explanation of The Problem by DCGAN

**Generator G(.)**
- Input: random numbers
- Output: generated image

**Discriminator D(.)**
- Input: generated/real image
- Output: prediction of real image

Uniform noise vector (random numbers)
Explanation of The Problem by DCGAN

Generator $G(.)$
- input = random numbers
- output = generated image

Discriminator $D(.)$
- input = generated/real image
- output = prediction of real image

Real image, so goal is $D(x) = 1$

Generated image, so goal is $D(G(z)) = 0$

Discriminator Goal: discriminate between real and generated images i.e., $D(x) = 1$, where $x$ is a real image $D(G(z)) = 0$, where $G(z)$ is a generated image
Explanation of The Problem by DCGAN

**Generator G(.)**
- input = random numbers,
- output = generated image

**Discriminator D(.)**
- input = generated/real image,
- output = prediction of real image

**Generator Goal:** Fool $D(G(z))$
- i.e., generate an image $G(z)$ such that $D(G(z))$ is wrong.
- i.e., $D(G(z)) = 1$

**Discriminator Goal:** discriminate between real and generated images
- i.e., $D(x)=1$, where $x$ is a real image
- $D(G(z))=0$, where $G(z)$ is a generated image

Real image, so goal is $D(x)=1$

Generated image, so goal is $D(G(z))=0$
**Generator G(.)**
- input = random numbers
- output = generated image

**Discriminator D(.)**
- input = generated/real image
- output = prediction of real image

**Generator Goal:** Fool $D(G(z))$
- i.e., generate an image $G(z)$ such that $D(G(z))$ is wrong.
- i.e., $D(G(z)) = 1$

**Discriminator Goal:** discriminate between real and generated images
- i.e., $D(x) = 1$, where $x$ is a real image
- $D(G(z)) = 0$, where $G(z)$ is a generated image

**Explanation of The Problem by DCGAN**

1. Both goals are unsupervised
2. Optimal when $D(.) = 0.5$ (i.e., cannot tell the difference between real and generated images) and $G(z)$ learns the training images distribution

**Notes**
- Conflicting goals
- Both goals are unsupervised
- Optimal when $D(.) = 0.5$ (i.e., cannot tell the difference between real and generated images) and $G(z)$ learns the training images distribution
CatGAN learns a discriminative neural network classifier $D$ that **maximize mutual information** between the inputs $x$ and the labels $y$ (as predicted through the conditional distribution $p(y | x, D)$ for the number of $K$ unknown categories).

**Problem?**

This goal of **perfect reconstruction** is often directly opposed to the goal of learning a classifier.

**Solution**

Usually GAN trained to model the data distribution through **reconstruction of input examples**.

A Mechanism for Trading Information Between $G$ and $D$
CatGAN

Be certain of class assignment for samples from $D$.

Be uncertain of assignment for generated samples.

Use all classes equally.

(i) minimize $H[p(y|x, D)]$

(ii) maximize $H[p(y|G(z), D)]$

(iii) maximize $H[p(y|D)]$
Objective Function

\[ \mathbb{E}_{x \sim \mathcal{X}} \left[ H \left[ p(y \mid x, D) \right] \right] = \frac{1}{N} \sum_{i=1}^{N} H \left[ p(y \mid x^i, D) \right] \]

\[ = \frac{1}{N} \sum_{i=1}^{N} - \sum_{k=1}^{K} p(y = k \mid x^i, D) \log p(y = k \mid x^i, D). \]

\[ \mathbb{E}_{z \sim P(z)} \left[ H \left[ p(y \mid D(z), D) \right] \right] \approx \frac{1}{M} \sum_{i=1}^{M} H \left[ p(y \mid G(z^i), D) \right], \text{ with } z^i \sim P(z), \]

\[ H_X \left[ p(y \mid D) \right] = H \left[ \frac{1}{N} \sum_{i=1}^{N} p(y \mid x^i, D) \right], \]

\[ H_G \left[ p(y \mid D) \right] \approx H \left[ \frac{1}{M} \sum_{i=1}^{M} p(y \mid G(z^i), D) \right], \text{ with } z^i \sim P(z). \]
Semi-supervised Learning

\[ CE[y, p(y \mid x, D)] = - \sum_{i=1}^{K} y_i \log p(y = y_i \mid x, D). \]

\[ \mathcal{L}_D^L = \max_D H_X[p(y \mid D)] - \mathbb{E}_{x \sim X} \left[ H[p(y \mid x, D)] \right] + \mathbb{E}_{z \sim P(z)} \left[ H[p(y \mid G(z), D)] \right] \\
+ \lambda \mathbb{E}_{(x, y) \sim X^L} \left[ CE[y, p(y \mid x, D)] \right], \]

\( \lambda \) is a cost weighting term
Experimental Results

The Circle Dataset with $K = 2$:

The RIM the objective function only specifies that the deep network has to separate the data into two equal classes, without any geometric constraints.
The Blob Dataset with $K = 3$:

The class identity is now known a priori as all models are trained unsupervisedly. Hence the different color/class assignment for different models.
Experimental Results

**MNIST and CIFAR-10 Datasets:**

Exemplary images produced by a generator trained using the semi-supervised CatGAN objective. We show samples for a generator trained on MNIST (left) CIFAR-10 (right). In this experiment the number of “pseudo” categories is 20.
# Experimental Results

## MNIST test error (%) with \( n \) labeled examples

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( n = 100 )</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmbedCNN (Weston et al., 2012)</td>
<td>7.75</td>
<td>-</td>
</tr>
<tr>
<td>SWWAE (Zhao et al., 2015)</td>
<td>8.71 ±0.34</td>
<td>0.71</td>
</tr>
<tr>
<td>Small-CNN (Rasmus et al., 2015)</td>
<td>6.43 (± 0.84)</td>
<td>0.36</td>
</tr>
<tr>
<td>Conv-Ladder ( \Gamma )-model (Rasmus et al., 2015)</td>
<td><strong>0.86 (± 0.41)</strong></td>
<td>-</td>
</tr>
<tr>
<td>RIM + CNN</td>
<td>10.75 (± 2.25)</td>
<td>0.53</td>
</tr>
<tr>
<td>Conv-GAN + SVM</td>
<td>15.43 (± 1.72)</td>
<td>9.64</td>
</tr>
<tr>
<td>Conv-CatGAN (unsupervised)</td>
<td>4.27</td>
<td></td>
</tr>
<tr>
<td>Conv-CatGAN (semi-supervised)</td>
<td>1.39 (± 0.28)</td>
<td>0.48</td>
</tr>
</tbody>
</table>

## CIFAR-10 test error (%) with \( n \) labeled examples

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( n = 4000 )</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>View-Invariant k-means Hui (2013)</td>
<td>27.4 (± 0.7)</td>
<td>18.1</td>
</tr>
<tr>
<td>Exemplar-CNN (Dosovitskiy et al., 2014)</td>
<td>23.4 (± 0.2)</td>
<td>15.7</td>
</tr>
<tr>
<td>Conv-Ladder ( \Gamma )-model (Rasmus et al., 2015)</td>
<td><strong>20.09 (± 0.46)</strong></td>
<td><strong>9.27</strong></td>
</tr>
<tr>
<td>Conv-CatGAN (semi-supervised)</td>
<td>19.58 (± 0.58)</td>
<td>9.38</td>
</tr>
</tbody>
</table>
Improved GAN

**Techniques to encourage convergence of the GAN’s game**

- Feature Matching
- Minibatch Discrimination
- Semi-supervised Learning

**Addresses the instability of GANs**: The new objective requires $G$ to generate data that matches the expected value of the features on an intermediate layer of $D$.

**Collapse of generator to a parameter setting where it always emits the same point**: Allow the discriminator to look at multiple data examples and perform minibatch discrimination.

**Minimizing both $L_{supervised}$, $L_{unsupervised}$**: Train $G$ to minimize the GAN game-value using $D$. 
Feature Matching

$f(x)$ : activations on an intermediate layer of the discriminator.

**New Generator Objective:**

$$\| \mathbb{E}_{x \sim P_{data}} f(x) - \mathbb{E}_{z \sim P_z(z)} f(G(z)) \|^2_2$$

The objective has a **fixed point** where exactly matches the distribution of training data. There is **no guarantee** of reaching this fixed point in practice.
Minibatch Discrimination

Identify generator samples that are close together: Let $f(x_i) \in \mathbb{R}^A$ denote a vector of features for input $x_i$ produced by some intermediate layer in $D$.

$C_b(x_i, x_j) = \exp(-\|M_{i,b} - M_{j,b}\|_{L_1})$

$C_b$ and $O(x_i)_b$ computation is similar to reduction operation and can be parallelized.
Virtual Batch Normalization

- **Minibatch:**
  - Showed its effectiveness on DCGAN
  - Makes the output be dependent on several other inputs in the same minibatch.

- **VBN:**
  - Each example is normalized based on the statistics collected on a reference batch of examples that are chosen once and fixed at the start of training, and on itself. The reference batch is normalized using only its own statistics.
  - VBN is computationally expensive because it requires running forward propagation on two minibatches of data, so it has been used only in the generator network.
Semi-supervised Learning

**Semi-supervised GAN objective function:** We can do semi-supervised learning by simply adding samples from the GAN generator $G$ to data set.

\[
L = -\mathbb{E}_{x,y \sim p_{data}(x,y)} \left[ \log p_{model}(y|x) \right] - \mathbb{E}_{x \sim G} \left[ \log p_{model}(y = K + 1|x) \right] \\
= L_{\text{supervised}} + L_{\text{unsupervised}}, \text{ where} \\
L_{\text{supervised}} = -\mathbb{E}_{x,y \sim p_{data}(x,y)} \log p_{model}(y|x, y < K + 1) \\
L_{\text{unsupervised}} = -\{\mathbb{E}_{x \sim p_{data}(x)} \log[1 - p_{model}(y = K + 1|x)] + \mathbb{E}_{x \sim G} \log[p_{model}(y = K + 1|x)]\}.
\]

\[
L_{\text{unsupervised}} = -\{\mathbb{E}_{x \sim p_{data}(x)} \log D(x) + \mathbb{E}_{z \sim \text{noise}} \log(1 - D(G(z)))\}.
\]
Semi-supervised Learning

Optimal solution for minimizing both $L_{supervised}$ and $L_{unsupervised}$:

- $\exp[l_j(x)] = c(x)p(y = j, x) \quad \forall \ j < K + 1$
- $\exp[l_{K+1}(x)] = c(x)P_G(x)$

In practice, $L_{unsupervised}$ will only help if it is not trivial to minimize for our classifier and we thus need to train $G$ to approximate the data distribution.

“One way to do this is by training $G$ to minimize the GAN game-value, using the discriminator $D$ defined by our classifier. This approach introduces an interaction between $G$ and our classifier that we do not fully understand yet, but empirically we find that optimizing $G$ using feature matching GAN works very well for semi-supervised learning, while training $G$ using GAN with minibatch discrimination does not work at all.”
Experimental Results

**MNIST Datasets:**

Samples generated by model during semi-supervised training with feature matching.

Right Samples generated with minibatch discrimination.
## Experimental Results

### MNIST Datasets:

<table>
<thead>
<tr>
<th>Model</th>
<th>20</th>
<th>50</th>
<th>100</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGN</td>
<td>333 ± 14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virtual Adversarial</td>
<td>212</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CatGAN</td>
<td>191 ± 10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Skip Deep Generative Model</td>
<td>132 ± 7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ladder network</td>
<td>106 ± 37</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auxiliary Deep Generative Model</td>
<td>96 ± 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our model</td>
<td>1677 ± 452</td>
<td>221 ± 136</td>
<td>93 ± 6.5</td>
<td>90 ± 4.2</td>
</tr>
<tr>
<td>Ensemble of 10 of our models</td>
<td>1134 ± 445</td>
<td>142 ± 96</td>
<td>86 ± 5.6</td>
<td>81 ± 4.3</td>
</tr>
</tbody>
</table>
Experimental Results

CIFAR-10 Datasets:
samples generated by model during semi-supervised training with feature matching.

Right Samples generated with minibatch discrimination.
## Experimental Results

**CIFAR-10 Datasets:**

<table>
<thead>
<tr>
<th>Model</th>
<th>Test error rate for a given number of labeled samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1000</td>
</tr>
<tr>
<td>Ladder network</td>
<td></td>
</tr>
<tr>
<td>CatGAN</td>
<td></td>
</tr>
<tr>
<td>Our model</td>
<td>21.83±2.01</td>
</tr>
<tr>
<td>Ensemble of 10 of our models</td>
<td>19.22±0.54</td>
</tr>
</tbody>
</table>
Experimental Results

ImageNet Datasets:

Samples generated by DCGAN.

Samples generated by Improved GAN.
Conclusion

CatGAN

- CatGAN is a framework for robust unsupervised and semi-supervised learning.
- This method combines neural network classifiers with an adversarial generative model that regularizes a discriminatively trained classifier.
- The generator is capable of generating images of high visual fidelity.

Improved GAN

- This research propose several techniques to stabilize training of GAN.
- The state-of-the-art results on a number of different data sets in computer vision achieved with applying proposed techniques to the problem of semi-supervised learning.
References


Useful Links

• CatGAN GitHub: https://github.com/smayru/catgan
• Improved GAN GitHub: https://github.com/openai/improved-gan
• DCGAN-tensorflow GitHub: https://github.com/carpedm20/DCGAN-tensorflow
• OpenAI Blog: https://blog.openai.com/generative-models/