Generative Adversarial Networks (GANs)

From Ian Goodfellow et al.

A short tutorial by :-

Binglin, Shashank & Bhargav
Outline

• **Part 1:** Introduction to GANs

• **Part 2:** Some challenges with GANs

• **Part 3:** Applications of GANs
Part 1

• Motivation for Generative Models
• From Adversarial Training to GANs
• GAN’s Architecture
• GAN’s objective
• DCGANs
GANs

• **Generative**
  • Learn a generative model

• **Adversarial**
  • Trained in an adversarial setting

• **Networks**
  • Use Deep Neural Networks
Why Generative Models?

• We’ve only seen discriminative models so far
  • Given an image $X$, predict a label $Y$
  • Estimates $P(Y|X)$

• Discriminative models have several key limitations
  • Can’t model $P(X)$, i.e. the probability of seeing a certain image
  • Thus, can’t sample from $P(X)$, i.e. can’t generate new images

• Generative models (in general) cope with all of above
  • Can model $P(X)$
  • Can generate new images
Magic of GANs...

Which one is Computer generated?

Magic of GANs...

http://people.eecs.berkeley.edu/~junyanz/projects/gvm/
Adversarial Training

• **In the last lecture, we saw:**
  • We can generate adversarial samples to fool a discriminative model
  • We can use those adversarial samples to make models robust
  • We then require more effort to generate adversarial samples
  • Repeat this and we get better discriminative model

• **GANs extend that idea to generative models:**
  • Generator: generate fake samples, tries to fool the Discriminator
  • Discriminator: tries to distinguish between real and fake samples
  • Train them against each other
  • Repeat this and we get better Generator and Discriminator
GAN’s Architecture

- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

Training Discriminator

Training Generator

Generator in action
GAN’s formulation

$$\min_G \max_D V(D, G)$$

• It is formulated as a **minimax game**, where:
  • The Discriminator is trying to maximize its reward $V(D, G)$
  • The Generator is trying to minimize Discriminator’s reward (or maximize its loss)

$$V(D, G) = \mathbb{E}_{x \sim p(x)}[\log D(x)] + \mathbb{E}_{z \sim q(z)}[\log(1 - D(G(z)))]$$

• The Nash equilibrium of this particular game is achieved at:
  • $p_{data}(x) = p_{gen}(x) \ \forall x$
  • $D(x) = \frac{1}{2} \ \forall x$
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 $p_g(z)$.
• 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:

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

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.
Vanishing gradient strikes back again...

$$V(D, G) = \mathbb{E}_{x \sim p(x)} \left[ \log D(x) \right] + \mathbb{E}_{z \sim q(z)} \left[ \log \left( 1 - D(G(z)) \right) \right]$$

$$\nabla_{\theta_G} V(D, G) = \nabla_{\theta_G} \mathbb{E}_{z \sim q(z)} \left[ \log \left( 1 - D(G(z)) \right) \right]$$

- \( \nabla_a \log(1 - \sigma(a)) = \frac{-\nabla_a \sigma(a)}{1 - \sigma(a)} = \frac{-\sigma(a)(1 - \sigma(a))}{1 - \sigma(a)} = -\sigma(a) = -D(G(z)) \)

- Gradient goes to 0 if \( D \) is confident, i.e. \( D(G(z)) \to 0 \)

- Minimize \(-\mathbb{E}_{z \sim q(z)} \left[ \log D(G(z)) \right]\) for Generator instead (keep Discriminator as it is)
Faces

CIFAR

DCGAN: Bedroom images

Deep Convolutional GANs (DCGANs)

Generator Architecture

Key ideas:

- Replace FC hidden layers with Convolutions
  - **Generator**: Fractional-Strided convolutions
- Use Batch Normalization after each layer
- **Inside Generator**
  - Use ReLU for hidden layers
  - Use Tanh for the output layer

Latent vectors capture interesting patterns...

Part 2

• Advantages of GANs
• Training Challenges
  • Non-Convergence
  • Mode-Collapse
• Proposed Solutions
  • Supervision with Labels
  • Mini-Batch GANs
• Modification of GAN’s losses
  • Discriminator (EB-GAN)
  • Generator (InfoGAN)
Advantages of GANs

• Plenty of existing work on Deep Generative Models
  • Boltzmann Machine
  • Deep Belief Nets
  • Variational AutoEncoders (VAE)

• Why GANs?
  • Sampling (or generation) is straightforward.
  • Training doesn't involve Maximum Likelihood estimation.
  • Robust to Overfitting since Generator never sees the training data.
  • Empirically, GANs are good at capturing the modes of the distribution.

Problems with GANs

- **Probability Distribution is Implicit**
  - Not straightforward to compute $P(X)$.
  - Thus *Vanilla GANs* are only good for Sampling/Generation.

- **Training is Hard**
  - Non-Convergence
  - Mode-Collapse

---

Training Problems

• **Non-Convergence**

• **Mode-Collapse**
• **Deep Learning models (in general) involve a single player**
  • The player tries to maximize its reward (minimize its loss).
  • Use SGD (with Backpropagation) to find the optimal parameters.
  • SGD has convergence guarantees (under certain conditions).
  • **Problem:** With non-convexity, we might converge to local optima.
    \[
    \min_G L(G)
    \]

• **GANs instead involve two (or more) players**
  • Discriminator is trying to maximize its reward.
  • Generator is trying to minimize Discriminator’s reward.
    \[
    \min_G \max_D V(D, G)
    \]
  • SGD was not designed to find the Nash equilibrium of a game.
  • **Problem:** We might not converge to the Nash equilibrium at all.

Non-Convergence

\[
\min_{x} \max_{y} V(x, y)
\]

Let \( V(x, y) = xy \)

- **State 1:**
  - \( x > 0 \)
  - \( y > 0 \)
  - \( V > 0 \)
  - Increase \( y \), Decrease \( x \)

- **State 2:**
  - \( x < 0 \)
  - \( y > 0 \)
  - \( V < 0 \)
  - Decrease \( y \), Decrease \( x \)

- **State 3:**
  - \( x < 0 \)
  - \( y < 0 \)
  - \( V > 0 \)
  - Decrease \( y \), Increase \( x \)

- **State 4:**
  - \( x > 0 \)
  - \( y < 0 \)
  - \( V < 0 \)
  - Increase \( y \), Increase \( x \)

- **State 5:**
  - \( x > 0 \)
  - \( y > 0 \)
  - \( V > 0 \)
  - \( \text{== State 1} \)
  - Increase \( y \), Decrease \( x \)
Non-Convergence

\[
\min_x \max_y xy
\]

- \( \frac{\partial}{\partial x} = -y \quad \text{...} \quad \frac{\partial}{\partial y} = x \)

- \( \frac{\partial^2}{\partial y^2} = \frac{\partial}{\partial x} = -y \)

- Differential equation’s solution has sinusoidal terms

- Even with a small learning rate, it will not converge

Problems with GANs

• Non-Convergence

• Mode-Collapse
Mode-Collapse

- Generator fails to output diverse samples

Expected Output

<table>
<thead>
<tr>
<th>Step</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image" alt="Expected Step 0" /></td>
</tr>
<tr>
<td>5k</td>
<td><img src="image" alt="Expected Step 5k" /></td>
</tr>
<tr>
<td>10k</td>
<td><img src="image" alt="Expected Step 10k" /></td>
</tr>
<tr>
<td>15k</td>
<td><img src="image" alt="Expected Step 15k" /></td>
</tr>
<tr>
<td>20k</td>
<td><img src="image" alt="Expected Step 20k" /></td>
</tr>
<tr>
<td>25k</td>
<td><img src="image" alt="Expected Step 25k" /></td>
</tr>
</tbody>
</table>

Target

Some real examples

Some Solutions

• Mini-Batch GANs
• Supervision with labels

• Some recent attempts :-
  • Unrolled GANs
  • W-GANs
Basic (Heuristic) Solutions

• Mini-Batch GANs
• Supervision with labels
How to reward sample diversity?

• **At Mode Collapse,**
  • Generator produces good samples, but a very few of them.
  • Thus, Discriminator can’t tag them as fake.

• **To address this problem,**
  • Let the Discriminator know about this edge-case.

• **More formally,**
  • Let the Discriminator look at the entire batch instead of single examples
  • If there is lack of diversity, it will mark the examples as fake

• **Thus,**
  • Generator will be forced to produce diverse samples.

Mini-Batch GANs

• Extract features that capture diversity in the mini-batch
  • For e.g. L2 norm of the difference between all pairs from the batch

• Feed those features to the discriminator along with the image

• Feature values will differ b/w diverse and non-diverse batches
  • Thus, Discriminator will rely on those features for classification

• This in turn,  
  • Will force the Generator to match those feature values with the real data  
  • Will generate diverse batches

Basic (Heuristic) Solutions

• Mini-Batch GANs

• Supervision with labels
Supervision with Labels

• Label information of the real data might help

• Empirically generates much better samples

Alternate view of GANs

\[
V(D, G) = \mathbb{E}_{x \sim p(x)}[\log D(x)] + \mathbb{E}_{z \sim q(z)}[\log(1 - D(G(z)))]
\]

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

- In this formulation, Discriminator’s strategy was \( D(x) \to 1, \quad D(G(z)) \to 0 \)

- Alternatively, we can flip the binary classification labels i.e. \( \text{Fake} = 1, \quad \text{Real} = 0 \)

\[
V(D, G) = \mathbb{E}_{x \sim p(x)}[\log(1 - D(x))] + \mathbb{E}_{z \sim q(z)}[\log(D(G(z)))]
\]

- In this new formulation, Discriminator’s strategy will be \( D(x) \to 0, \quad D(G(z)) \to 1 \)

Alternate view of GANs (Contd.)

• If all we want to encode is $D(x) \to 0$, $D(G(z)) \to 1$

\[
D^* = \argmax_D \mathbb{E}_{x \sim p(x)} [\log(1 - D(x))] + \mathbb{E}_{z \sim q(z)} [\log(D(G(z)))]
\]

We can use this

\[
D^* = \argmin_D \mathbb{E}_{x \sim p(x)} \log(D(x)) + \mathbb{E}_{z \sim q(z)} [\log(1 - D(G(z)))]
\]

• Now, we can replace cross-entropy with any loss function (Hinge Loss)

\[
D^* = \argmin_D \mathbb{E}_{x \sim p(x)} D(x) + \mathbb{E}_{z \sim q(z)} \max(0, m - D(G(z)))
\]

• And thus, instead of outputting probabilities, Discriminator just has to output:-
  • High values for fake samples
  • Low values for real samples

Energy-Based GANs

• Modified game plans
  • **Generator** will try to generate samples with low values
  • **Discriminator** will try to assign high scores to fake values

• Use AutoEncoder inside the Discriminator

• Use **Mean-Squared Reconstruction error as** $D(x)$
  • High Reconstruction Error for Fake samples
  • Low Reconstruction Error for Real samples

\[
D(x) = ||Dec(Enc(x)) - x||_{MSE}
\]

The Cool Stuff...

3D Faces

(a) Azimuth (pose)
(b) Elevation
(c) Lighting
(d) Wide or Narrow

Cool Stuff (contd.)

3D Chairs

(a) Rotation

(b) Width

How to reward Disentanglement?

- Disentanglement means individual dimensions independently capturing key attributes of the image

- Let’s partition the noise vector into 2 parts :-
  - $z$ vector will capture slight variations in the image
  - $c$ vector will capture the main attributes of the image
    - For e.g. Digit, Angle and Thickness of images in MNIST

- If $c$ vector captures the key variations in the image,

**Will $c$ and $x_{fake}$ be highly correlated or weakly correlated?**
Recap: Mutual Information

- Mutual Information captures the mutual dependence between two variables.

- Mutual information between two variables $X, Y$ is defined as:

$$I(X; Y) = \sum_{x,y} p(x, y) \log \left( \frac{p(x, y)}{p(x)p(y)} \right)$$

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$
InfoGAN

• We want to maximize the mutual information $I$ between $c$ and $x = G(z, c)$

• Incorporate in the value function of the minimax game.

$$\min_{G} \max_{D} V_I(D, G) = V(D, G) - \lambda I(c; G(z, c))$$
Mutual Information’s Variational Lower bound

\[ I(c; G(z, c)) = H(c) - H(c|G(z, c)) \]

\[ = \mathbb{E}_{x \sim G(z,c)} \left[ \mathbb{E}_{c' \sim P(c|x)}[\log P(c'|x)] \right] + H(c) \]

\[ = \mathbb{E}_{x \sim G(z,c)} \left[ D_{KL}(P||Q) + \mathbb{E}_{c' \sim P(c|x)}[\log Q(c'|x)] \right] + H(c) \]

\[ \geq \mathbb{E}_{x \sim G(z,c)} \left[ \mathbb{E}_{c' \sim P(c|x)}[\log Q(c'|x)] \right] + H(c) \]

\[ \geq \mathbb{E}_{c \sim P(c), x \sim G(z,c)}[\log Q(c|x)] + H(c) \]
Part 3

- **Conditional GANs**
- **Applications**
  - Image-to-Image Translation
  - Text-to-Image Synthesis
  - Face Aging
- **Advanced GAN Extensions**
  - Coupled GAN
  - LAPGAN – Laplacian Pyramid of Adversarial Networks
  - Adversarially Learned Inference
- **Summary**
Conditional GANs

MNIST digits generated conditioned on their class label.

Figure 2 in the original paper.
Conditional GANs

• Simple modification to the original GAN framework that conditions the model on additional information for better multi-modal learning.

• Lends to many practical applications of GANs when we have explicit supervision available.


Part 3

• Conditional GANs

• **Applications**
  • Image-to-Image Translation
  • Text-to-Image Synthesis
  • Face Aging

• **Advanced GAN Extensions**
  • Coupled GAN
  • LAPGAN – Laplacian Pyramid of Adversarial Networks
  • Adversarially Learned Inference

• **Summary**
Image-to-Image Translation

![Figure 1 in the original paper.](image)

Link to an interactive demo of this paper

Image-to-Image Translation

• Architecture: DCGAN-based architecture

• Training is conditioned on the images from the source domain.

• Conditional GANs provide an effective way to handle many complex domains without worrying about designing structured loss functions explicitly.

Text-to-Image Synthesis

Motivation

Given a text description, generate images closely associated.

Uses a conditional GAN with the generator and discriminator being condition on “dense” text embedding.

Figure 1 in the original paper.

Text-to-Image Synthesis

Positive Example:
Real Image, Right Text

Negative Examples:
Real Image, Wrong Text
Fake Image, Right Text

Face Aging with Conditional GANs

- Differentiating Feature: Uses an *Identity Preservation Optimization* using an auxiliary network to get a better approximation of the latent code \((z^*)\) for an input image.
- Latent code is then conditioned on a discrete (one-hot) embedding of age categories.

![Diagram](image)

Figure 1 in the original paper.

Face Aging with Conditional GANs

Figure 3 in the original paper.

Conditional GANs

Conditional Model Collapse

• Scenario observed when the Conditional GAN starts *ignoring* either the code (c) or the noise variables (z).

• This limits the diversity of images generated.

Part 3

• Conditional GANs

• Applications
  • Image-to-Image Translation
  • Text-to-Image Synthesis
  • Face Aging

• Advanced GAN Extensions
  • Coupled GAN
  • LAPGAN – Laplacian Pyramid of Adversarial Networks
  • Adversarially Learned Inference

• Summary
Coupled GAN

- Learning a joint distribution of multi-domain images.
- Using GANs to learn the joint distribution with samples drawn from the marginal distributions.
- Direct applications in domain adaptation and image translation.

Figure 2 in the original paper.

Coupled GANs

- Architecture

Weight-sharing constraints the network to learn a joint distribution without corresponding supervision.

Coupled GANs

- Some examples of generating facial images across different feature domains.

- Corresponding images in a column are generated from the same latent code $z$.

Figure 4 in the original paper.

Laplacian Pyramid of Adversarial Networks

- Based on the Laplacian Pyramid representation of images. (1983)
- Generate high resolution (dimension) images by using a hierarchical system of GANs
- Iteratively increase image resolution and quality.

Laplacian Pyramid of Adversarial Networks

Image Generation using a LAPGAN
- Generator $G_3$ generates the base image $\tilde{I}_3$ from random noise input $z_3$.
- Generators $(G_2, G_1, G_0)$ iteratively generate the difference image ($\tilde{h}$) conditioned on previous small image ($l$).
- This difference image is added to an up-scaled version of previous smaller image.

Laplacian Pyramid of Adversarial Networks

Training Procedure:
Models at each level are trained independently to learn the required representation.
Adversarially Learned Inference

• Basic idea is to learn an encoder/inference network along with the generator network.

• Consider the following joint distributions over $x$ (image) and $z$ (latent variables):

$$q(x, z) = q(x) \cdot q(z|x) \quad \text{encoder distribution}$$

$$p(x, z) = p(z) \cdot p(x|z) \quad \text{generator distribution}$$

Adversarially Learned Inference

**Figure 1 in the original paper.**

\[
\min_G \max_D \mathbb{E}_{q(x)}[\log(D(x, G_z(x)))] + \mathbb{E}_{p(x)}[\log(1 - D(G_x(z), z))]
\]

Adversarially Learned Inference

- Nash equilibrium yields
  - Joint: \( p(x, z) \sim q(x, z) \)
  - Marginals: \( p(x) \sim q(x) \) and \( p(z) \sim q(z) \)
  - Conditionals: \( p(x|z) \sim q(x|z) \) and \( p(z|x) \sim q(z|x) \)

- Inferred latent representation successfully reconstructed the original image.
- Representation was useful in the downstream semi-supervised task.

Summary

• GANs are generative models that are implemented using two stochastic neural network modules: **Generator** and **Discriminator**.
• **Generator** tries to generate samples from random noise as input
• **Discriminator** tries to distinguish the samples from Generator and samples from the real data distribution.
• Both networks are trained adversarially (in tandem) to fool the other component. In this process, both models become better at their respective tasks.
Why use GANs for Generation?

• Can be trained using back-propagation for Neural Network based Generator/Discriminator functions.
• Sharper images can be generated.
• Faster to sample from the model distribution: *single* forward pass generates a *single* sample.
Reading List


Applications:

Questions?