Generative Adversarial Networks
Lecture 2

Introduction To Deep Learning (11-785 / 685 / 485) Spring 2022

Fuyu Tang, Roshan Ram, Ameya Mahabaleshwarkar
Overview

- GANs Quick Recap
- GAN Training
- Issues with GAN Training
- Remedies for GAN Training Issues
- GAN Architectures & Recent Progress
● Discriminative models learn the conditional probability $P(Y \mid X)$
● Generative models understand the joint probability $P(X, Y) = P(X \mid Y) \cdot P(Y)$
   ○ GANs model $P(X)$, the distribution of training data
   ○ Once sufficiently trained, GANs can generate new data from $P(X)$
● Generative networks are used to generate samples from an unlabeled distribution $P(X)$ given samples $X_1, \ldots, X_n$. For example:
   ○ Generate images given sample images
   ○ Generate music given sample music
   ○ Generate realistic text given sample corpus
GAN Framework

## GAN Training

<table>
<thead>
<tr>
<th>Description</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training samples</td>
<td>$x$</td>
</tr>
<tr>
<td>Latent noise vector</td>
<td>$z$</td>
</tr>
<tr>
<td>Discriminator</td>
<td>$D(x; \theta_d)$</td>
</tr>
<tr>
<td>Generator</td>
<td>$G(z; \theta_g)$</td>
</tr>
<tr>
<td>Probability distribution of real data</td>
<td>$P_{data}$</td>
</tr>
<tr>
<td>Probability distribution of fake data</td>
<td>$P_q$</td>
</tr>
<tr>
<td>Probability distribution of latent noise vector</td>
<td>$P_z$</td>
</tr>
<tr>
<td>Generator output - generated <em>fake</em> data</td>
<td>$G(z)$</td>
</tr>
<tr>
<td>Discriminator output for <em>fake</em> data</td>
<td>$D(G(z))$</td>
</tr>
<tr>
<td>Discriminator output for <em>real</em> data</td>
<td>$D(x)$</td>
</tr>
</tbody>
</table>
GAN Training

<table>
<thead>
<tr>
<th>Description</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training samples</td>
<td>$x$</td>
</tr>
<tr>
<td>Latent noise vector</td>
<td>$z$</td>
</tr>
<tr>
<td>Discriminator</td>
<td>$D(x; \theta_d)$</td>
</tr>
<tr>
<td>Generator</td>
<td>$G(z; \theta_g)$</td>
</tr>
<tr>
<td>Probability distribution of real data</td>
<td>$P_{\text{data}}$</td>
</tr>
<tr>
<td>Probability distribution of fake data</td>
<td>$P_{\hat{z}}$</td>
</tr>
<tr>
<td>Probability distribution of latent noise vector</td>
<td>$P_z$</td>
</tr>
<tr>
<td>Generator output - generated fake data</td>
<td>$G(z)$</td>
</tr>
<tr>
<td>Discriminator output for fake data</td>
<td>$D(G(z))$</td>
</tr>
<tr>
<td>Discriminator output for real data</td>
<td>$D(x)$</td>
</tr>
</tbody>
</table>

**Discriminator objective:**
- maximize output for real data $D(x)$
- minimize output for fake data $D(G(x))$

**OR**
- maximize output for real data $D(x)$
- maximize output for $1 - D(G(z))$

**Generator objective:**
- minimize output for fake data $1 - D(G(z))$

Since log is a monotonically increasing function:

**Discriminator objective:**
- maximize output for real data $\log(D(x))$
- maximize output for $\log(1 - D(G(z)))$

**Generator objective:**
- minimize output for fake data $\log(1 - D(G(z)))$
GAN Training

<table>
<thead>
<tr>
<th>Description</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training samples</td>
<td>$x$</td>
</tr>
<tr>
<td>Latent noise vector</td>
<td>$z$</td>
</tr>
<tr>
<td>Discriminator</td>
<td>$D(x; \theta_d)$</td>
</tr>
<tr>
<td>Generator</td>
<td>$G(z; \theta_g)$</td>
</tr>
<tr>
<td>Probability distribution of real data</td>
<td>$P_{data}$</td>
</tr>
<tr>
<td>Probability distribution of fake data</td>
<td>$P_{g}$</td>
</tr>
<tr>
<td>Probability distribution of latent noise vector</td>
<td>$P_{z}$</td>
</tr>
<tr>
<td>Generator output - generated fake data</td>
<td>$G(z)$</td>
</tr>
<tr>
<td>Discriminator output for fake data</td>
<td>$D(G(z))$</td>
</tr>
<tr>
<td>Discriminator output for real data</td>
<td>$D(x)$</td>
</tr>
</tbody>
</table>

For one data point:

**Discriminator objective:**
- $\max_D (\log(D(x)) + \log(1 - D(G(z))))$

**Generator objective:**
- $\min_G (\log(1 - D(G(z))))$

For the entire distribution:

**Discriminator objective:**
- $\max_D \mathbb{E}_{x \sim P_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim P_z} [1 - \log D(G(z))]$

**Generator objective:**
- $\min_G \mathbb{E}_{z \sim P_z} [1 - \log D(G(z))]$

Which stays same after adding a constant,
- $\min_G \mathbb{E}_{x \sim P_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim P_z} [1 - \log D(G(z))]$
GAN Training

<table>
<thead>
<tr>
<th>Description</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training samples</td>
<td>$x$</td>
</tr>
<tr>
<td>Latent noise vector</td>
<td>$z$</td>
</tr>
<tr>
<td>Discriminator</td>
<td>$D(x; \theta_d)$</td>
</tr>
<tr>
<td>Generator</td>
<td>$G(z; \theta_g)$</td>
</tr>
<tr>
<td>Probability distribution of real data</td>
<td>$P_{data}$</td>
</tr>
<tr>
<td>Probability distribution of fake data</td>
<td>$P_g$</td>
</tr>
<tr>
<td>Probability distribution of latent noise vector</td>
<td>$P_z$</td>
</tr>
<tr>
<td>Generator output - generated fake data</td>
<td>$G(z)$</td>
</tr>
<tr>
<td>Discriminator output for fake data</td>
<td>$D(G(z))$</td>
</tr>
<tr>
<td>Discriminator output for real data</td>
<td>$D(x)$</td>
</tr>
</tbody>
</table>

GAN objective:

- $\min_{G} \max_{D} E_{x \sim P_{data}(x)}[\log D(x)] + E_{z \sim P_z(z)}[1 - \log D(G(z))]$
- $\min_{G} \max_{D} P_{data}(x)(\log D(x)) + P_g(x)(1 - \log D(x))$

Jointly optimizing min-max is complicated, so we first find the current best $D$ by taking the derivative of GAN objective:

- $-\frac{P_{data}(x)}{D(x)} + \frac{P_g(x)}{1-D(x)} = 0$

$D(x) = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)}$

Substituting this value in the GAN objective for Generator loss:

- $E_{x \sim P_{data}(x)}[\log \frac{P_{data}(x)}{\frac{1}{2}(P_{data}(x) + P_g(x))}] + E_{z \sim P_z(z)}[\log \frac{P_{data}(x)}{\frac{1}{2}(P_{data}(x) + P_g(x))}] - 2 \log 2$

Kullback-Leibler(KL) and Jensen-Shannon(JS) divergences are given by:

- $KL(P_1 || P_2) = E_{x \sim P_1(x)}[\log \frac{P_1}{P_2}]$
- $JSD(P_1 || P_2) = \frac{1}{2} KL(P_1 || \frac{P_1 + P_2}{2}) + \frac{1}{2} KL(P_2 || \frac{P_1 + P_2}{2})$

This makes the Generator loss:

- $2JSD(P_{data} || P_g) - 2 \log 2$
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 $p_{data}(x)$.
        • Update the discriminator by ascending its stochastic gradient:
          \[ \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D(x^{(i)}) + \log \left( 1 - D(G(z^{(i)})) \right) \right]. \]
    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:
      \[ \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left( 1 - D(G(z^{(i)})) \right). \]
end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.
Qualitative Effects of JS-Divergence

- The KL-divergence $D_{KL}(p, q)$ penalizes the generator if it misses some modes of images: the penalty is high where $p(x) > 0$ but $q(x) \to 0$. Nevertheless, it is acceptable that some images do not look real. The penalty is low when $p(x) \to 0$ but $q(x) > 0$. (Poorer quality but more diverse samples)

- JS-divergence is symmetrical. Unlike KL-divergence, it will penalize poor images badly, but can allow less diversity

GANs Training Issues

1. Mode Collapse
2. Vanishing Gradients
3. Convergence and Oscillation
GANs Training Issues - Mode Collapse

- If a generator produces an especially plausible output, the generator may learn to produce only that output.
- The **Generator gets stuck** at a point where it only produces a limited variety of samples or one sample repeatedly during or after training.
- Each iteration of Generator over-optimizes for a particular Discriminator, and the Discriminator never manages to learn its way out of the trap.

GANs Training Issues - Vanishing Gradients

- If the Discriminator is too good, then the Generator training can fail due to vanishing gradients. An optimal Discriminator doesn't provide enough information for the Generator to make progress.
- Below image shows how gradients will vanish if the distribution of generated images ($p$) is too different than the distribution of real images ($q_1, q_2, q_3$)

GANs Training Issues - Convergence and Oscillation

● GAN training is based on a zero-sum, non-cooperative, minmax game. In short, if one wins the other loses
● In game theory, the GAN model converges when the discriminator and the generator reach a Nash equilibrium. This is the optimal point for the GAN objective
● Simplified example: Consider two player A and B which control the value of x and y respectively. Player A wants to maximize the value $xy$ while B wants to minimize it
GANs Training Issues - Convergence and Oscillation

- We update the parameter $x$ and $y$ based on the gradient of the functions -
  
  $f(x) = xy$, $f(y) = -xy$
  
  $\frac{\partial f}{\partial x} = y$ and $\frac{\partial f}{\partial y} = -x$
  
  $x \rightarrow x-\alpha \cdot y$ and $y \rightarrow y+\alpha \cdot x$ ( $\alpha$ is learning rate )
GANs Training Issues - Convergence and Oscillation

- We update the parameter $x$ and $y$ based on the gradient of the functions $f(x) = xy$, $f(y) = -xy$

  \[
  \frac{\partial f}{\partial x} = y \quad \text{and} \quad \frac{\partial f}{\partial y} = -x
  \]

  
  \[x \rightarrow x - \alpha \cdot y \quad \text{and} \quad y \rightarrow y + \alpha \cdot x \quad (\alpha \text{ is learning rate})\]

- The Nash equilibrium is $x = y = 0$. This is the only state where the action of your opponent does not matter. It is the only state that any opponents’ actions will not change the game outcome

Feature Matching

Statistics of generated images should match statistics of real images.

Discriminator produces multidimensional output, a “statistic” of data.

Generator trained to minimize L2 between real and generated data.

Discriminator trained to maximize L2 between real and generated data.

Goal: matching features in real images

\[ \| E_X D(X) - E_Z D(G(Z)) \|^2_2 \]

\[ \| E_X f(X) - E_Z f(G(Z)) \|^2_2 \]

Minibatch Discrimination

Discriminator can look at multiple inputs at once and decide if those inputs come from the real or generated distribution.

- GANs frequently collapse to a single point
- Discriminator needs to differentiate between two distributions
- Easier task if looking at multiple samples

Append the similarity between the image and other images in the same batch in one of the dense layers in the discriminator to classify whether this image is real or generated.

Historical averaging

Dampen oscillations by encouraging updates to converge to a mean.

- GANs frequently create a cycle or experience oscillations
- **Add a term** to reduce oscillations that encourage the current parameters to be near a **moving average of the parameters**

\[ \left\| \theta - \frac{1}{t} \sum_{i}^{t} \theta_i \right\|_2^2 \]

One-sided Label Smoothing

Don’t over-penalize generated images

- Label smoothing is a common and easy technique that improves performance across many domains

- Sigmoid tries hard to saturate to 0 or 1 but can never quite reach that goal

- Provide targets that are epsilon or 1- epsilon so the sigmoid doesn’t saturate and overtrain

- Experimentally, **smooth the real targets** but **do not smooth the generated targets** when training the discriminator.

Virtual Batch Normalization

Use batch normalization to accelerate convergence

- Batch normalization accelerates convergence
- However, hard to apply in adversarial setting
- Collect statistics on **fixed batch** of real data and use to normalize other data.

Recap

VAE

KL Divergence

GANs

Jenson-Shanon Divergence
Special Case:

\[ P(X) \]

\[ X = 0 \]

\[ Q(X) \]

\[ X = 1 \]
KL-Divergence:

Let $\theta$ be the distance between the two peaks of the distribution.

If $\theta \neq 0$, 
$$KL(P\|Q) = 1 \log \left(\frac{1}{0}\right) = \infty$$

If $\theta = 0$, 
$$KL(P\|Q) = 1 \log \left(\frac{1}{1}\right) = 0$$

Not differentiable w.r.t $\theta$

![Diagram showing two distributions P(X) and Q(X) with X = 0 and X = 1](image-url)
Jenson-Shanon-Divergence:

\[ m(X) = \frac{P_D + P_G}{2} \]

\[ JS(P_D || P_G) = \frac{1}{2} KL(P_D || m) + \frac{1}{2} KL(P_G || m) \]

Let \( \theta \) be the distance between the two peaks of the distribution.

If \( \theta \neq 0 \),

\[ JSD(P || Q) = 0.5 \times (1 \log \left( \frac{1}{0.5} \right) + 1 \log \left( \frac{1}{0.5} \right)) = \log 2 \]

If \( \theta = 0 \),

\[ JSD(P || Q) = 0.5 \times (1 \log \left( \frac{1}{1} \right) + 1 \log \left( \frac{1}{1} \right)) = 0 \]

**Constant to \( \theta \)**

\[ P(X) \]

---

\[ X = 0 \]

---

\[ Q(X) \]

---

\[ X = 1 \]
Wasserstein Distance: \[ W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} [ \|x - y\| ] \]

- Minimum cost of turning one pile of dirt into another pile of dirt, when both distributions are treated as pile of dirt.
- The total $\Sigma$ mass $\times$ mean distance required to transform one distribution to another

Red points, Blue points represent two different distributions.
Wasserstein Distance:

Let $\theta$ be the distance between the two peaks of the distribution.

$$W(P,Q) = |\theta|$$

Differentiable w.r.t $\theta$
WGAN

\[
W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} \left[ \|x - y\| \right]
\]

Kantorovitch-Rubinstein duality

\[
\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{x \sim \mathbb{P}_r} [D(x)] - \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g} [D(\tilde{x})]
\]

D should be a 1-Lipschitz function:

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

A function is K-Lipschitz if its gradients are at most K everywhere.
\[\rightarrow W(\mathbb{P}_r, \mathbb{P}_g) \text{ is continuous everywhere, and differentiable almost everywhere}\]

Weight Clipping

Restrict weights between [-c, c]

Arjovsky, Martin, Soumith Chintala, and Léon Bottou.
“Wasserstein generative adversarial networks.”
Gradient Penalty

\[ L = \mathbb{E}_{\hat{x} \sim P_g} [D(\hat{x})] - \mathbb{E}_{x \sim P_r} [D(x)] + \lambda \mathbb{E}_{\hat{x} \sim P_\alpha} [(\| \nabla_{\hat{x}} D(\hat{x}) \|_2 - 1)^2] . \]

Gradient penalty introduces a softer constraint on gradients

- more stable training
- requires very little hyper-parameter tuning

Differentiable  Not Differentiable  Not Differentiable

Meaningful Loss ($\theta \neq 0$)  Constant ($\theta \neq 0$)  Not symmetric

Figure 1: These plots show $\rho(p, p_0)$ as a function of $\theta$ when $\rho$ is the EM distance (left plot) or the JS divergence (right plot). The EM plot is continuous and provides a usable gradient everywhere. The JS plot is not continuous and does not provide a usable gradient.
GAN Architectures

There are many variations of GANs for modeling different tasks. This is not meant to be exhaustive but a sample of the possibilities.

- GAN
- Conditional GAN
- LapGAN
- Recurrent Adversarial Network
- Categorical GAN
- InfoGAN
- AAE
- BiGAN
- CycleGAN
GAN

Unqualified, “GAN” typically refers to a simple model of $P(X)$ [GPM$^{+}14$]. This is a vanilla GAN. Think unsupervised generation of unlabeled images, video, etc.
Conditional GANs

A conditional GAN models $P(X \mid Y)$. For example, generate samples of MNIST conditioned on the digit you are generating [MO14]. The model is constructed by adding the labels $Y$ as an input to both generator and discriminator.

$$\min_G \max_D V(D, G) = \mathbb{E}_X \log D(X, Y) + \mathbb{E}_Z \log D(G(Z, Y), Y)$$
Conditional GAN Architecture
Conditional GAN Results

Figure 2: Generated MNIST digits, each row conditioned on one label
LapGAN

A Laplacian GAN is constructed of a chain of conditional GANs, to generate progressively larger images. A GAN generates small, blurry images. A conditional GAN generates larger images conditioned on the smaller image, repeated until you reach the desired size. [DCSF15]
LapGAN Architecture
BiGANs

A Bi-Directional Generative Adversarial Network trains an encoder/decoder pair in an elegant fashion. The discriminator tries to tell the difference between pairs of real data and encoded real data from data generated from prior samples and prior samples. [DKD16]

\[ V(D, E, G) = \mathbb{E}_X \log D(X, E(X)) + \mathbb{E}_Z \log (1 - D(G(Z), Z)) \]

This method simultaneously trains the pair and does not require any assumptions about the distance metric in either the hidden or output spaces.
BiGAN Architecture

Figure 1: The structure of Bidirectional Generative Adversarial Networks (BiGAN).
CycleGAN

CycleGAN trains a pair of conditional GANs to perform image-to-image translation [ZPIE17].

- GAN $A$ trained to convert from $X$ to $Y$
- GAN $B$ trained to convert from $Y$ to $X$
- Additional “cycle-consistency” losses $\| Y - A(B(Y)) \|_1$ and $\| X - B(A(X)) \|$
CycleGAN Results
CycleGAN Architecture

Start

Input_A

Discriminator A

Decision [0,1]

Generator A2B

Decision [0,1]

Discriminator B

Generated_B

Cyclic_A

Generator B2A
CycleGAN Results
CycleGAN Results
CycleGAN Lesson

- There is no paired dataset of zebras and horses
- So no easy discriminative method to train zebras from horses
- But using GANs, can train distributions to match
Neural Style Transfer Results
Neural Style Transfer Results
Style Transfer with CycleGAN

Input     Monet     Van Gogh     Cezanne     Ukiyo-e

Image 1. Original image (Input).
Image 2. Image stylized in Monet's style.
Image 4. Image stylized in Cezanne's style.
Image 5. Image stylized in Ukiyo-e style.
Neural Style Transfer vs CycleGAN

- **Neural Style Transfer**
  - Need a *content* and *style* image
  - Specific & small number of images
  - More control
  - [https://reiinakano.com/arbitrary-image-stylization-tfjs/](https://reiinakano.com/arbitrary-image-stylization-tfjs/)

- **CycleGAN**
  - Just need 2 domains of images. No need for specific *content* or *style* images
  - Many similar pictures
  - Specificity of images doesn’t really matter
[ADDITIONAL] Kaggle: CycleGAN for Monet Paintings

https://www.kaggle.com/code/dimitreoliveira/introduction-to-cyclegan-monet-paintings

I’m Something of a Painter Myself

Introduction to CycleGAN - Monet paintings
Just Think an Extra couple of Seconds before Assuming Something is Real