CS6101
Deep Unsupervised Learning
Recess Week
WGAN, WGAN-GP, Progressive GAN
Ang Yi Zhe
Wasserstein GAN
Problems with Vanilla GANs

- Unstable training - hard to achieve Nash Equilibrium
- Low dimensional supports
- Vanishing gradient
- Mode Collapse
- Lack of a proper evaluation metric
- Not robust to architectures and hyperparameter choices
Problems with GANs - Low Dimensional Supports

Problem arises when supports of $Pr$ and $Pg$ lie on low dimensional manifolds

→ Disjoint Supports

→ Easily find perfect discriminator

→ No gradient signal during training
Wasserstein GAN

A new GAN training algorithm

- Good empirical results backed up by theory
- Able to train the discriminator to convergence
  - Removing the need to balance discriminator/generator updates.
- Correlation between discriminator loss and perceptual quality
  - Easier to gauge training progress and determine stopping criteria.
Wasserstein Distance - An Alternative Divergence Measure

\[ W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}(x, y) \sim \gamma \left[ \| x - y \| \right] \]

Minimum energy cost of moving and transforming a pile of dirt in the shape of one probability distribution, to the shape of the other distribution, where

Energy cost = Amount of Dirt * Moving Distance
Wasserstein Distance - Explained

Adapted from: Jonathan Hui, Medium
Wasserstein Distance - Explained

Transport Plan

<table>
<thead>
<tr>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Adapted from: Jonathan Hui, Medium
Wasserstein Distance - Explained

Energy Cost = Amount of Dirt × Moving Distance

\[ = \gamma(x, y) \|x - y\| \]

\[ = 2 \times 6 \]

\[ \gamma(3, 9) = 2 \]

Adapted from: Jonathan Hui, Medium
**Wasserstein Distance - Explained**

\[ \text{Total Energy Cost} = \sum_{x,y} \gamma(x, y) \| x - y \| \]

\[ = \mathbb{E}_{(x,y) \sim \gamma} \| x - y \| \]

**Transport Plan**

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Adapted from: [Jonathan Hui, Medium](https://jondez.net/medium)
Wasserstein Distance - Explained

\[ \gamma_i \in \prod(\mathcal{P}_r, \mathcal{P}_g) \]

Set of all possible Joint Probability Distributions between \( \mathcal{P}_r \) and \( \mathcal{P}_g \)
Wasserstein Distance - Explained

\[ W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|] \]

Find the smallest value among all valid transport plans

Sum of distance moved, weighted by the amount of mass moved

**Minimum energy cost** of *moving and transforming* a pile of dirt in the shape of one probability distribution, to the shape of the other distribution, where

**Energy cost** = Amount of Dirt \(*\) Moving Distance
Comparison of Distance Measures

**KL-Divergence**

\[ D_{KL}(P\|Q) = \int_x P(x) \log \frac{P(x)}{Q(x)} \, dx \]

**JS-Divergence**

\[ D_{JS}(P\|Q) = \frac{1}{2} D_{KL}(P\|\frac{P+Q}{2}) + \frac{1}{2} D_{KL}(Q\|\frac{P+Q}{2}) \]

**Wasserstein Distance**

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

How do these various measures perform when both the real and generator’s data lie on low dimensional manifolds?
Comparison of Distance Measures

\( \forall (x, y) \in P, x = 0 \text{ and } y \sim U(0, 1) \)

\( \forall (x, y) \in Q, x = \theta, 0 \leq \theta \leq 1 \text{ and } y \sim U(0, 1) \)

What is the distance between these two disjoint distributions?
Comparison of Distance Measures

\[ D_{KL}(P \parallel Q) = D_{KL}(Q \parallel P) = \begin{cases} +\infty & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0, \end{cases} \]

\[ D_{JS}(P \parallel Q) = \begin{cases} \log 2 & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0, \end{cases} \]

\[ W(P, Q) = |\theta| \]

There exist cases for KL and JS where,
- The distributions don’t converge
- The gradient is always 0

→ WD is best; Provides a smooth measure
Kantorovich-Rubinstein Duality

\[ W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}(x,y) \sim \gamma [\|x - y\|] \]

is intractable, so the paper shows how we can compute an approximation:

\[
\left[ \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_g}[f(x)] \right]
\]

Suppose \( f \) comes from a family of K-Lipschitz continuous functions, \( \{f_w\}_{w \in W} \), parameterized by \( w \),

\[
\geq \max_{w \in W} \mathbb{E}_{x \sim P_r}[f_w(x)] - \mathbb{E}_{x \sim P_g}[f_w(x)]
\]

To learn \( w \) to find a good \( f_w \) to approximate the Wasserstein Distance between \( P_r \) and \( P_g \to \) use a neural network!
To train \( P_g = g_\theta(Z) \) to match \( P_r \) using the Wasserstein Distance,

\[
W(P_r, P_g) = \max_{f_w \in W} \mathbb{E}_{x \sim P_r} [f_w(x)] - \mathbb{E}_{z \sim Z} [f_w(g_\theta(z))]
\]

01 For a fixed generator, sample from real data and generator to train \( f_w \) to convergence using gradient ascent, in order to approximate the Wasserstein Distance.

\[
\nabla W(P_r, P_g) = -\mathbb{E}_{z \sim Z} [\nabla_\theta f_w(g_\theta(z))]
\]

02 Sample from the generator, and use the approximate Wasserstein Distance to train the generator using gradient descent.

03 Repeat.

Similar to original minimax GAN setup!
**WGAN vs GAN**

**Vanilla GAN**
\[
\min_G \max_D \mathbb{E}_{x \sim P_r}[ \log D(x) ] + \mathbb{E}_{z \sim Z}[ \log(1 - D(G(z))) ]
\]

**WGAN**
\[
\min_G \max_D \mathbb{E}_{x \sim P_r}[ D(x) ] - \mathbb{E}_{z \sim Z}[ D(G(z)) ]
\]

- Uses Wasserstein Loss
- Predictions no longer constrained to [0, 1], but can be any real number
- Critic must be K-Lipschitz continuous (by clipping the weights)
- Train the critic multiple times for each update of generator
K-Lipschitz Continuity

There exists a real constant $K \geq 0$ s.t. for all $x_1, x_2 \in \mathbb{R}$,

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|, \quad \left| \frac{f(x_1) - f(x_2)}{x_1 - x_2} \right| \leq K$$

We require a limit on the rate at which the predictions can change between any two images.

Enforce the Lipschitz constraint by clipping the weights of the critic to lie within a small range, after each training batch.
Results - Nice Gradients

Taken from: Wasserstein GAN
Results - Correlates with Image Quality

Taken from: Wasserstein GAN
Results - Robust to Architectural Changes

Taken from: Wasserstein GAN
Wasserstein GAN - GP
Problems with WGAN

“Weight clipping is clearly a terrible way to enforce a Lipschitz constraint” - Authors
Weight Clipping - Instability in Training

Weights concentrated at lower/upper bounds of clipping interval

Vanishing / Exploding Gradients

Taken from: Improved Training of Wasserstein GANs
Weight Clipping - Reduced Capacity of Model

8 Gaussians  25 Gaussians  Swiss Roll

Taken from: Improved Training of Wasserstein GANs
A differentiable function $f$ is $1$-Lipschitz if and only if it has gradients with norm at most $1$ everywhere.

$$\max_D \mathbb{E}_{x \sim P_r} [D(x)] - \mathbb{E}_{\tilde{x} \sim P_g} [D(\tilde{x})] + \lambda \mathbb{E}_{\hat{x} \sim P_{\hat{x}}} [(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]$$

- Sample from a linear interpolation between real and fake samples
- Norm of gradient of critic w.r.t. input
- Penalise when value is away from $1$
WGAN-GP vs WGAN

- Include a gradient penalty term in the critic loss function
- Don’t clip the weights of the critic
- Don’t use batch normalization layers in the critic
- More computationally intensive
### Results - Enhanced Training Stability

<table>
<thead>
<tr>
<th>DCGAN</th>
<th>LSGAN</th>
<th>WGAN (clipping)</th>
<th>WGAN-GP (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline</strong> ($G$: DCGAN, $D$: DCGAN)</td>
<td><img src="image1" alt="Baseline" /></td>
<td><img src="image2" alt="Baseline" /></td>
<td><img src="image3" alt="Baseline" /></td>
</tr>
<tr>
<td>$G$: No BN and a constant number of filters, $D$: DCGAN</td>
<td><img src="image4" alt="Baseline" /></td>
<td><img src="image5" alt="Baseline" /></td>
<td><img src="image6" alt="Baseline" /></td>
</tr>
<tr>
<td>$G$: 4-layer 512-dim ReLU MLP, $D$: DCGAN</td>
<td><img src="image7" alt="Baseline" /></td>
<td><img src="image8" alt="Baseline" /></td>
<td><img src="image9" alt="Baseline" /></td>
</tr>
<tr>
<td>No normalization in either $G$ or $D$</td>
<td><img src="image10" alt="Baseline" /></td>
<td><img src="image11" alt="Baseline" /></td>
<td><img src="image12" alt="Baseline" /></td>
</tr>
<tr>
<td>Gated multiplicative nonlinearities everywhere in $G$ and $D$</td>
<td><img src="image13" alt="Baseline" /></td>
<td><img src="image14" alt="Baseline" /></td>
<td><img src="image15" alt="Baseline" /></td>
</tr>
<tr>
<td>tanh nonlinearities everywhere in $G$ and $D$</td>
<td><img src="image16" alt="Baseline" /></td>
<td><img src="image17" alt="Baseline" /></td>
<td><img src="image18" alt="Baseline" /></td>
</tr>
<tr>
<td>101-layer ResNet $G$ and $D$</td>
<td><img src="image19" alt="Baseline" /></td>
<td><img src="image20" alt="Baseline" /></td>
<td><img src="image21" alt="Baseline" /></td>
</tr>
</tbody>
</table>

*Taken from: [Improved Training of Wasserstein GANs](https://arxiv.org/abs/1701.07875)*
Results - Enhanced Training Stability

Unsupervised

<table>
<thead>
<tr>
<th>Method</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALI [8] (in [27])</td>
<td>5.34 ± .05</td>
</tr>
<tr>
<td>BEGAN [4]</td>
<td>5.62</td>
</tr>
<tr>
<td>DCGAN [22] (in [11])</td>
<td>6.16 ± .07</td>
</tr>
<tr>
<td>Improved GAN (-L+HA) [23]</td>
<td>6.86 ± .06</td>
</tr>
<tr>
<td>EGAN-Ent-VI [7]</td>
<td>7.07 ± .10</td>
</tr>
<tr>
<td>DFM [27]</td>
<td>7.72 ± .13</td>
</tr>
<tr>
<td><strong>WGAN-GP ResNet (ours)</strong></td>
<td><strong>7.86 ± .07</strong></td>
</tr>
</tbody>
</table>

Supervised

<table>
<thead>
<tr>
<th>Method</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>SteinGAN [26]</td>
<td>6.35</td>
</tr>
<tr>
<td>DCGAN (with labels, in [26])</td>
<td>6.58</td>
</tr>
<tr>
<td>Improved GAN [23]</td>
<td>8.09 ± .07</td>
</tr>
<tr>
<td>AC-GAN [20]</td>
<td>8.25 ± .07</td>
</tr>
<tr>
<td>WGAN-GP ResNet (ours)</td>
<td>8.42 ± .10</td>
</tr>
<tr>
<td><strong>SGAN [11]</strong></td>
<td>8.59 ± .12</td>
</tr>
</tbody>
</table>

Taken from: Improved Training of Wasserstein GANs
Progressive GAN
Key Innovations of ProGAN

- Progressively growing and smoothly fading in higher-resolution layers
- Mini-batch standard deviation
- Equalized learning rate
- Pixel-wise feature normalization
- 1024x1024 images (or even more)!

Progressive Growing of GANs for Improved Quality, Stability, and Variation
Progressive Growing

Instead of training all the layers of the generator and discriminator at once, we gradually grow the GAN, one layer at a time, to handle progressively higher resolution images.

Start off with an easy to traverse loss landscape, and then gradually increase the complexity as we get closer to the objective.

Taken from: Sarah Wolf, Medium
Introduce 2 new pathways:

1. Nearest Neighbour Upscaling
   a. Upscale old output

2. NNU + Convolution Layer
   a. Learned upscaling

Introduce new layer but also retain some of the previous output, **smoothly** (linearly) fading in the new layer
Mini-batch Standard Deviation

Introduce a “minibatch standard deviation” layer near the end of the discriminator - computes the standard deviations of the feature map pixels across the batch, and appends as an extra channel.

- A way for the Discriminator to tell whether the samples it is getting are varied enough
- If SD is low → fake
- Encourage Generator to increase variance of generated samples
Pixel-wise Feature Normalization

Normalize the feature vector in each pixel to have unit norm in the generator after each convolutional layer.

- Stability of training
- Less memory intensive than batch-norm
- Prevent feature map magnitudes from getting too large