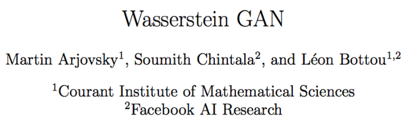
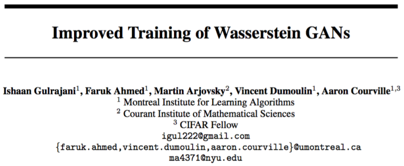


Wasserstein GAN

Martin Arjovsky[1], Soumith Chintala[2], and Léon Bottou[1,2]

[1]Courant Institute of Mathematical Sciences
[2]Facebook AI Research



Improved Training of Wasserstein GANs

Ishaan Gulrajani[1], Faruk Ahmed[1], Martin Arjovsky[2], Vincent Dumoulin[1], Aaron Courville[1,3]
[1] Montreal Institute for Learning Algorithms
[2] Courant Institute of Mathematical Sciences
[3] CIFAR Fellow
igul222@gmail.com
{faruk.ahmed,vincent.dumoulin,aaron.courville}@umontreal.ca
ma4371@nyu.edu

# Nishant Gurnani

GAN Reading Group

April 14th, 2017

## Why are these Papers Important?

## Why are these Papers Important?

- Recently a large number of GAN frameworks have been proposed - BGAN, LSGAN, DCGAN, DiscoGAN ...

## Why are these Papers Important?

- Recently a large number of GAN frameworks have been proposed - BGAN, LSGAN, DCGAN, DiscoGAN . . .
- Wasserstein GAN is yet another GAN training algorithm, however it is backed up by rigorous theory in addition to good performance

## Why are these Papers Important?

- Recently a large number of GAN frameworks have been proposed - BGAN, LSGAN, DCGAN, DiscoGAN ...
- Wasserstein GAN is yet another GAN training algorithm, however it is backed up by rigorous theory in addition to good performance
- WGAN removes the need to balance generator updates with discriminator updates, thus removing a key source of training instability in the original GAN

## Why are these Papers Important?

- Recently a large number of GAN frameworks have been proposed - BGAN, LSGAN, DCGAN, DiscoGAN . . .
- Wasserstein GAN is yet another GAN training algorithm, however it is backed up by rigorous theory in addition to good performance
- WGAN removes the need to balance generator updates with discriminator updates, thus removing a key source of training instability in the original GAN
- Empirical results show correlation between discriminator loss and perceptual quality thus providing a rough measure of training progress

## Why are these Papers Important?

- Recently a large number of GAN frameworks have been proposed - BGAN, LSGAN, DCGAN, DiscoGAN . . .
- Wasserstein GAN is yet another GAN training algorithm, however it is backed up by rigorous theory in addition to good performance
- WGAN removes the need to balance generator updates with discriminator updates, thus removing a key source of training instability in the original GAN
- Empirical results show correlation between discriminator loss and perceptual quality thus providing a rough measure of training progress

**Goal** - **Convince you that WGAN is the “best” GAN**

## Outline

## Outline

## What does it mean to learn a probability distribution?

When learning generative models, we assume the data we have comes from some unknown distribution $\mathbb{P}_r$.

Want to learn a distribution $\mathbb{P}_\theta$ that approximates $\mathbb{P}_r$, where $\theta$ are the parameters of the distribution.

There are two approaches for doing this:

## What does it mean to learn a probability distribution?

When learning generative models, we assume the data we have comes from some unknown distribution $\mathbb{P}_r$.

Want to learn a distribution $\mathbb{P}_\theta$ that approximates $\mathbb{P}_r$, where $\theta$ are the parameters of the distribution.

There are two approaches for doing this:

1. Directly learn the probability density function $\mathbb{P}_\theta$ and then optimize through maximum likelihood estimation

## What does it mean to learn a probability distribution?

When learning generative models, we assume the data we have comes from some unknown distribution $\mathbb{P}_r$.

Want to learn a distribution $\mathbb{P}_\theta$ that approximates $\mathbb{P}_r$, where $\theta$ are the parameters of the distribution.

There are two approaches for doing this:

1. Directly learn the probability density function $\mathbb{P}_\theta$ and then optimize through maximum likelihood estimation
2. Learn a function that transforms an existing distribution $Z$ into $\mathbb{P}_\theta$. Here, $g_\theta$ is some differentiable function, $Z$ is a common distribution (usually uniform or Gaussian), and $\mathbb{P}_\theta = g_\theta(Z)$

## Maximum Likelihood approach is problematic

Recall that for continuous distributions $\mathbb{P}$ and $\mathbb{Q}$ the *KL* divergence is:

$$KL(\mathbb{P}||\mathbb{Q}) = \int_x \mathbb{P}(x) \log \frac{\mathbb{P}(x)}{\mathbb{Q}(x)} dx$$

and given function $\mathbb{P}_\theta$, the MLE objective is

$$\max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log \mathbb{P}_\theta(x^{(i)})$$

In the limit (as $m \to \infty$), samples will appear based on the data distribution $\mathbb{P}_r$, so

$$\lim_{m \to \infty} \max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log \mathbb{P}_\theta(x^{(i)}) = \max_{\theta \in \mathbb{R}^d} \int_x \mathbb{P}_r(x) \log \mathbb{P}_\theta(x) dx$$

## Maximum Likelihood approach is problematic

$$\begin{aligned}
\lim_{m\to\infty} \max_{\theta\in\mathbb{R}^d} \frac{1}{m}\sum_{i=1}^{m} \log \mathbb{P}_\theta(x^{(i)}) &= \max_{\theta\in\mathbb{R}^d} \int_x \mathbb{P}_r(x) \log \mathbb{P}_\theta(x) dx \\
&= \min_{\theta\in\mathbb{R}^d} -\int_x \mathbb{P}_r(x) \log \mathbb{P}_\theta(x) dx \\
&= \min_{\theta\in\mathbb{R}^d} \int_x \mathbb{P}_r(x) \log \mathbb{P}_r(x) dx - \int_x \mathbb{P}_r(x) \log \mathbb{P} \\
&= \min_{\theta\in\mathbb{R}^d} KL(\mathbb{P}_r || \mathbb{P}_\theta)
\end{aligned}$$

## Maximum Likelihood approach is problematic

$$\begin{aligned}
\lim_{m\to\infty} \max_{\theta\in\mathbb{R}^d} \frac{1}{m}\sum_{i=1}^{m} \log \mathbb{P}_\theta(x^{(i)}) &= \max_{\theta\in\mathbb{R}^d} \int_x \mathbb{P}_r(x)\log \mathbb{P}_\theta(x)dx \\
&= \min_{\theta\in\mathbb{R}^d} -\int_x \mathbb{P}_r(x)\log \mathbb{P}_\theta(x)dx \\
&= \min_{\theta\in\mathbb{R}^d} \int_x \mathbb{P}_r(x)\log \mathbb{P}_r(x)dx - \int_x \mathbb{P}_r(x)\log \mathbb{P} \\
&= \min_{\theta\in\mathbb{R}^d} KL(\mathbb{P}_r||\mathbb{P}_\theta)
\end{aligned}$$

- Note if $\mathbb{P}_\theta = 0$ at an $x$ where $\mathbb{P}_r > 0$, the $KL$ divergence goes to $+\infty$ (bad for the MLE if $\mathbb{P}_\theta$ has low dimensional support)

## Maximum Likelihood approach is problematic

$$\begin{aligned}
\lim_{m\to\infty} \max_{\theta\in\mathbb{R}^d} \frac{1}{m}\sum_{i=1}^{m} \log \mathbb{P}_\theta(x^{(i)}) &= \max_{\theta\in\mathbb{R}^d} \int_x \mathbb{P}_r(x)\log\mathbb{P}_\theta(x)dx \\
&= \min_{\theta\in\mathbb{R}^d} -\int_x \mathbb{P}_r(x)\log\mathbb{P}_\theta(x)dx \\
&= \min_{\theta\in\mathbb{R}^d} \int_x \mathbb{P}_r(x)\log\mathbb{P}_r(x)dx - \int_x \mathbb{P}_r(x)\log\mathbb{P} \\
&= \min_{\theta\in\mathbb{R}^d} KL(\mathbb{P}_r||\mathbb{P}_\theta)
\end{aligned}$$

- Note if $\mathbb{P}_\theta = 0$ at an $x$ where $\mathbb{P}_r > 0$, the $KL$ divergence goes to $+\infty$ (bad for the MLE if $\mathbb{P}_\theta$ has low dimensional support)
- Typical remedy is to add a noise term to the model distribution to ensure distribution is defined everywhere

## Maximum Likelihood approach is problematic

$$
\begin{aligned}
\lim_{m\to\infty} \max_{\theta\in\mathbb{R}^d} \frac{1}{m}\sum_{i=1}^{m} \log \mathbb{P}_\theta(x^{(i)}) &= \max_{\theta\in\mathbb{R}^d} \int_x \mathbb{P}_r(x)\log \mathbb{P}_\theta(x)dx \\
&= \min_{\theta\in\mathbb{R}^d} -\int_x \mathbb{P}_r(x)\log \mathbb{P}_\theta(x)dx \\
&= \min_{\theta\in\mathbb{R}^d} \int_x \mathbb{P}_r(x)\log \mathbb{P}_r(x)dx - \int_x \mathbb{P}_r(x)\log \mathbb{P} \\
&= \min_{\theta\in\mathbb{R}^d} KL(\mathbb{P}_r||\mathbb{P}_\theta)
\end{aligned}
$$

- Note if $\mathbb{P}_\theta = 0$ at an $x$ where $\mathbb{P}_r > 0$, the $KL$ divergence goes to $+\infty$ (bad for the MLE if $\mathbb{P}_\theta$ has low dimensional support)
- Typical remedy is to add a noise term to the model distribution to ensure distribution is defined everywhere
- This unfortunately introduces some error, and empirically people have needed to add a lot of random noise to make models train

## Transforming an existing distribution

Shortcomings of the maximum likelihood approach motivate the second approach of learning a $g_\theta$ (a generator) to transform a known distribution Z.

Advantages of this approach:

## Transforming an existing distribution

Shortcomings of the maximum likelihood approach motivate the second approach of learning a $g_\theta$ (a generator) to transform a known distribution Z.

Advantages of this approach:

- Unlike densities, this approach can represent distributions confined to a low dimensional manifold

## Transforming an existing distribution

Shortcomings of the maximum likelihood approach motivate the second approach of learning a $g_\theta$ (a generator) to transform a known distribution Z.

Advantages of this approach:

- Unlike densities, this approach can represent distributions confined to a low dimensional manifold
- It's very easy to generate samples - given a trained $g_\theta$, simply sample random noise $z \sim Z$, and evaluate $g_\theta(z)$

## Transforming an existing distribution

Shortcomings of the maximum likelihood approach motivate the second approach of learning a $g_\theta$ (a generator) to transform a known distribution Z.

Advantages of this approach:

- Unlike densities, this approach can represent distributions confined to a low dimensional manifold
- It's very easy to generate samples - given a trained $g_\theta$, simply sample random noise $z \sim Z$, and evaluate $g_\theta(z)$

### VAEs and GANs are well known examples of this approach

VAEs focus on the approximate likelihood of the examples and so share the limitation that you need to fiddle with additional noise terms.

## Transforming an existing distribution

Shortcomings of the maximum likelihood approach motivate the second approach of learning a $g_\theta$ (a generator) to transform a known distribution Z.

Advantages of this approach:

- Unlike densities, this approach can represent distributions confined to a low dimensional manifold
- It's very easy to generate samples - given a trained $g_\theta$, simply sample random noise $z \sim Z$, and evaluate $g_\theta(z)$

### VAEs and GANs are well known examples of this approach

VAEs focus on the approximate likelihood of the examples and so share the limitation that you need to fiddle with additional noise terms.

GANs offer much more flexibility but their training is unstable.

## Transforming an existing distribution

To train $g_\theta$ (and by extension $\mathbb{P}_\theta$), we need a measure of distance between distributions i.e. $d(\mathbb{P}_r, \mathbb{P}_\theta)$.

## Transforming an existing distribution

To train $g_\theta$ (and by extension $\mathbb{P}_\theta$), we need a measure of distance between distributions i.e. $d(\mathbb{P}_r, \mathbb{P}_\theta)$.

### Distance Properties

## Transforming an existing distribution

To train $g_\theta$ (and by extension $\mathbb{P}_\theta$), we need a measure of distance between distributions i.e. $d(\mathbb{P}_r, \mathbb{P}_\theta)$.

### Distance Properties

- Distance $d$ is weaker than distance $d'$ if every sequence that converges under $d'$ converges under $d$

## Transforming an existing distribution

To train $g_\theta$ (and by extension $\mathbb{P}_\theta$), we need a measure of distance between distributions i.e. $d(\mathbb{P}_r, \mathbb{P}_\theta)$.

### Distance Properties

- Distance $d$ is weaker than distance $d'$ if every sequence that converges under $d'$ converges under $d$
- Given a distance $d$, we can treat $d(\mathbb{P}_r, \mathbb{P}_\theta)$ as a loss function

## Transforming an existing distribution

To train $g_\theta$ (and by extension $\mathbb{P}_\theta$), we need a measure of distance between distributions i.e. $d(\mathbb{P}_r, \mathbb{P}_\theta)$.

### Distance Properties

- Distance $d$ is weaker than distance $d'$ if every sequence that converges under $d'$ converges under $d$
- Given a distance $d$, we can treat $d(\mathbb{P}_r, \mathbb{P}_\theta)$ as a loss function
- We can minimize $d(\mathbb{P}_r, \mathbb{P}_\theta)$ with respect to $\theta$ as long as the mapping $\theta \mapsto \mathbb{P}_\theta$ is continuous (true if $g_\theta$ is a neural network)

## Transforming an existing distribution

To train $g_\theta$ (and by extension $\mathbb{P}_\theta$), we need a measure of distance between distributions i.e. $d(\mathbb{P}_r, \mathbb{P}_\theta)$.

### Distance Properties

- Distance $d$ is weaker than distance $d'$ if every sequence that converges under $d'$ converges under $d$
- Given a distance $d$, we can treat $d(\mathbb{P}_r, \mathbb{P}_\theta)$ as a loss function
- We can minimize $d(\mathbb{P}_r, \mathbb{P}_\theta)$ with respect to $\theta$ as long as the mapping $\theta \mapsto \mathbb{P}_\theta$ is continuous (true if $g_\theta$ is a neural network)

**How do we define $d$?**

## Distance definitions

## Distance definitions

### Notation

$\chi$ - compact metric set (such as the space of images $[0,1]^d$)
$\Sigma$ - set of all Borel subsets of $\chi$
Prob($\chi$) - space of probability measures defined on $\chi$

## Distance definitions

### Notation

$\chi$ - compact metric set (such as the space of images $[0,1]^d$)
$\Sigma$ - set of all Borel subsets of $\chi$
Prob($\chi$) - space of probability measures defined on $\chi$

### Total Variation (TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_\theta(A)|$$

## Distance definitions

### Notation

$\chi$ - compact metric set (such as the space of images $[0,1]^d$)
$\Sigma$ - set of all Borel subsets of $\chi$
Prob($\chi$) - space of probability measures defined on $\chi$

### Total Variation (TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_\theta(A)|$$

### Kullback-Leibler (KL) divergence

$$KL(\mathbb{P}_r || \mathbb{P}_\theta) = \int_x \mathbb{P}_r(x) \log \frac{\mathbb{P}_r(x)}{\mathbb{P}_\theta(x)} d\mu(x)$$

where both $\mathbb{P}_r$ and $\mathbb{P}_\theta$ are assumed to be absolutely continuous with respect to a same measure $\mu$ defined on $\chi$.

## Distance Definitions

## Distance Definitions

### Jensen-Shannon (JS) Divergence

$$JS(\mathbb{P}_r, \mathbb{P}_\theta) = KL(\mathbb{P}_r || \mathbb{P}_m) + KL(\mathbb{P}_\theta || \mathbb{P}_m)$$

where $\mathbb{P}_m$ is the mixture $(\mathbb{P}_r + \mathbb{P}_\theta)/2$

## Distance Definitions

### Jensen-Shannon (JS) Divergence

$$JS(\mathbb{P}_r, \mathbb{P}_\theta) = KL(\mathbb{P}_r || \mathbb{P}_m) + KL(\mathbb{P}_\theta || \mathbb{P}_m)$$

where $\mathbb{P}_m$ is the mixture $(\mathbb{P}_r + \mathbb{P}_\theta)/2$

### Earth-Mover (EM) distance or Wasserstein-1

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

where $\Pi(\mathbb{P}_r, \mathbb{P}_\theta)$ denotes the set of all join distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_\theta$

## Understanding the EM distance

# Understanding the EM distance

## Main Idea

Probability distributions are defined by how much mass they put on each point.

## Understanding the EM distance

### Main Idea

Probability distributions are defined by how much mass they put on each point.

Imagine we started with distribution $\mathbb{P}_r$, and wanted to move mass around to change the distribution into $\mathbb{P}_\theta$.

## Understanding the EM distance

### Main Idea

Probability distributions are defined by how much mass they put on each point.

Imagine we started with distribution $\mathbb{P}_r$, and wanted to move mass around to change the distribution into $\mathbb{P}_\theta$.

Moving mass $m$ by distance $d$ costs $m \cdot d$ effort.

## Understanding the EM distance

### Main Idea

Probability distributions are defined by how much mass they put on each point.

Imagine we started with distribution $\mathbb{P}_r$, and wanted to move mass around to change the distribution into $\mathbb{P}_\theta$.

Moving mass $m$ by distance $d$ costs $m \cdot d$ effort.

The earth mover distance is the minimal effort we need to spend.

## Understanding the EM distance

### Main Idea

Probability distributions are defined by how much mass they put on each point.

Imagine we started with distribution $\mathbb{P}_r$, and wanted to move mass around to change the distribution into $\mathbb{P}_\theta$.

Moving mass $m$ by distance $d$ costs $m \cdot d$ effort.

The earth mover distance is the minimal effort we need to spend.

### Transport Plan

Each $\gamma \in \Pi$ is a transport plan and to execute the plan, for all $x, y$ move $\gamma(x, y)$ mass from $x$ to $y$.

## Understanding the EM distance

## Understanding the EM distance

What properties does the plan need to satisfy to transform $\mathbb{P}_r$ into $\mathbb{P}_\theta$?

## Understanding the EM distance

What properties does the plan need to satisfy to transform $\mathbb{P}_r$ into $\mathbb{P}_\theta$?

The amount of mass that leaves $x$ is $\int_y \gamma(x, y)dy$. This must equal $\mathbb{P}_r(x)$, the amount of mass originally at $x$.

## Understanding the EM distance

What properties does the plan need to satisfy to transform $\mathbb{P}_r$ into $\mathbb{P}_\theta$?

The amount of mass that leaves $x$ is $\int_y \gamma(x, y)dy$. This must equal $\mathbb{P}_r(x)$, the amount of mass originally at $x$.

The amount of mass that enters $y$ is $\int_x \gamma(x, y)dx$. This must equal $\mathbb{P}_\theta(y)$, the amount of mass that ends up at y.

## Understanding the EM distance

What properties does the plan need to satisfy to transform $\mathbb{P}_r$ into $\mathbb{P}_\theta$?

The amount of mass that leaves $x$ is $\int_y \gamma(x, y)dy$. This must equal $\mathbb{P}_r(x)$, the amount of mass originally at $x$.

The amount of mass that enters $y$ is $\int_x \gamma(x, y)dx$. This must equal $\mathbb{P}_\theta(y)$, the amount of mass that ends up at y.

This shows why the marginals of $\gamma \in \Pi$ must be $\mathbb{P}_r$ and $\mathbb{P}_\theta$. For scoring, the effort spent is

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

## Understanding the EM distance

What properties does the plan need to satisfy to transform $\mathbb{P}_r$ into $\mathbb{P}_\theta$?

The amount of mass that leaves $x$ is $\int_y \gamma(x, y) dy$. This must equal $\mathbb{P}_r(x)$, the amount of mass originally at $x$.

The amount of mass that enters $y$ is $\int_x \gamma(x, y) dx$. This must equal $\mathbb{P}_\theta(y)$, the amount of mass that ends up at y.

This shows why the marginals of $\gamma \in \Pi$ must be $\mathbb{P}_r$ and $\mathbb{P}_\theta$. For scoring, the effort spent is

$$\int_x \int_y \gamma(x, y) ||x - y|| dy dx = \mathbb{E}_{(x,y)\sim\gamma}[||x - y||]$$

Computing the infimum of this over all valid $\gamma$ gives the earth mover distance

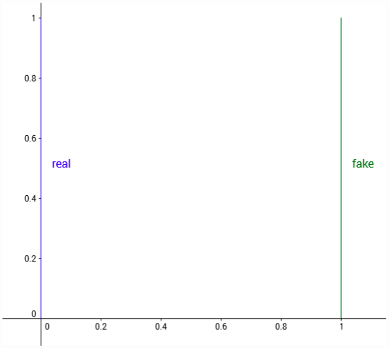
## Learning Parallel Lines Example

## Learning Parallel Lines Example

Let $Z \sim U[0,1]$ and let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$, uniform on a straight vertical line passing through the origin. Now let $g_\theta(z) = (\theta, z)$ with $\theta$ a single real parameter.

## Learning Parallel Lines Example

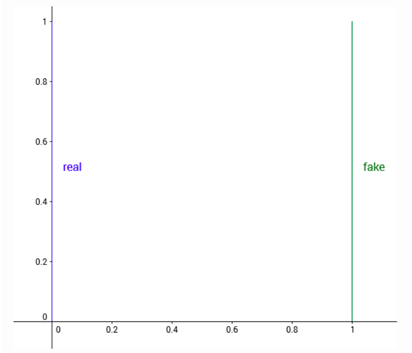
Let $Z \sim U[0,1]$ and let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$, uniform on a straight vertical line passing through the origin. Now let $g_\theta(z) = (\theta, z)$ with $\theta$ a single real parameter.



Real and fake distribution when $\theta = 1$

## Learning Parallel Lines Example

Let $Z \sim U[0,1]$ and let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$, uniform on a straight vertical line passing through the origin. Now let $g_\theta(z) = (\theta, z)$ with $\theta$ a single real parameter.



Real and fake distribution when $\theta = 1$

We'd like our optimization algorithm to learn to move $\theta$ to 0. As $\theta \to 0$, the distance $d(\mathbb{P}_0, \mathbb{P}_\theta)$ should decrease.

## Learning Parallel Lines Example

For many common distance functions, this doesn't happen.

## Learning Parallel Lines Example

For many common distance functions, this doesn't happen.

$$\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$

$$JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$

$$KL(\mathbb{P}_\theta, \mathbb{P}_0) = KL(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0 \\ 0 & \text{if } \theta = 0 \end{cases}$$

$$W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|$$

## Theoretical Justification

### Theorem (1)

*Let $\mathbb{P}_r$ be a fixed distribution over $\chi$. Let $Z$ be a random variable over another space $\mathcal{Z}$. Let $g : \mathcal{Z} \times \mathbb{R}^d \to \chi$ be a function, that will be denote $g_\theta(z)$ with $z$ the first coordinate and $\theta$ the second. Let $\mathbb{P}_\theta$ denote the distribution of $g_\theta()$. Then,*

1. *If $g$ is continuous in $\theta$, so is $W(\mathbb{P}_r, \mathbb{P}_\theta)$.*
2. *If $g$ is locally Lipschitz and satisfies regularity assumption 1, then $W(\mathbb{P}_r, \mathbb{P}_\theta)$ is continuous everywhere, and differentiable almost everywhere*
3. *Statements 1-2 are false for the Jensen-Shannon divergence $JS(\mathbb{P}_r, \mathbb{P}_\theta)$ and all the KLs.*

## Theoretical Justification

### Theorem (2)

*Let $\mathbb{P}$ be a distribution on a compact space $\chi$ and $(\mathbb{P}_n)_{n\in\mathbb{N}}$ be a sequence of distributions on $\chi$. Then, considering all limits as $n \to \infty$,*

1. *The following statements are equivalent*
   - $\delta(\mathbb{P}_n, \mathbb{P}) \to 0$
   - $JS(\mathbb{P}_n, \mathbb{P}) \to 0$
2. *The following statements are equivalent*
   - $W(\mathbb{P}_n, \mathbb{P}) \to 0$
   - $\mathbb{P}_n \xrightarrow{D} \mathbb{P}$ *where* $\xrightarrow{D}$ *represents convergence in distribution for random variables*
3. $KL(\mathbb{P}_n||\mathbb{P}) \to 0$ *or* $KL(\mathbb{P}||\mathbb{P}_n) \to 0$ *imply the statements in (1).*
4. *The statements in (1) imply the statements in (2).*

## Generative Adversarial Networks

Recall that the GAN training strategy is to define a game between two competing networks.

## Generative Adversarial Networks

Recall that the GAN training strategy is to define a game between two competing networks.

The *generator* network $G$ maps a source of noise to the input space.

## Generative Adversarial Networks

Recall that the GAN training strategy is to define a game between two competing networks.

The *generator* network $G$ maps a source of noise to the input space.

The *discriminator* network $D$ receives either a generated sample or a true data sample and must distinguish between the two.

## Generative Adversarial Networks

Recall that the GAN training strategy is to define a game between two competing networks.

The *generator* network $G$ maps a source of noise to the input space.

The *discriminator* network $D$ receives either a generated sample or a true data sample and must distinguish between the two.

The generator is trained to fool the discriminator.

## Generative Adversarial Networks

Formally we can express the game between the generator $G$ and the discriminator $D$ with the minimax objective:

$$\min_G \max_D \mathbb{E}_{x \sim \mathbb{P}_r}[\log(D(x))] + \mathbb{E}_{\widetilde{x} \sim \mathbb{P}_g}[\log(1 - D(\widetilde{x}))]$$

where $\mathbb{P}_r$ is the data distribution and $\mathbb{P}_g$ is the model distribution implicitly defined by $\widetilde{x} = G(z)$, $z \sim p(z)$

## Generative Adversarial Networks

### Remarks

## Generative Adversarial Networks

### Remarks

- If the discriminator is trained to optimality before each generator parameter update, minimizing the value function amounts to minimizing the Jensen-Shannon divergence between the data and model distributions on $x$

# Generative Adversarial Networks

## Remarks

- If the discriminator is trained to optimality before each generator parameter update, minimizing the value function amounts to minimizing the Jensen-Shannon divergence between the data and model distributions on $x$
- This is expensive and often leads to vanishing gradients as the discriminator saturates

# Generative Adversarial Networks

## Remarks

- If the discriminator is trained to optimality before each generator parameter update, minimizing the value function amounts to minimizing the Jensen-Shannon divergence between the data and model distributions on $x$
- This is expensive and often leads to vanishing gradients as the discriminator saturates
- In practice, this requirement is relaxed, and the generator and discriminator are update simultaneously

## Kantorivich-Rubinstein Duality

Unfortunately, computing the Wasserstein distance exactly is intractable.

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

## Kantorivich-Rubinstein Duality

Unfortunately, computing the Wasserstein distance exactly is intractable.

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

However, a result from the Kantorivich-Rubinstein Duality (Villani 2008) shows $W$ is equivalent to

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

where the supremum is taken over all 1-Lipschitz functions

## Kantorivich-Rubinstein Duality

Unfortunately, computing the Wasserstein distance exactly is intractable.

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

However, a result from the Kantorivich-Rubinstein Duality (Villani 2008) shows $W$ is equivalent to

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

where the supremum is taken over all 1-Lipschitz functions

**Calculating this is still intractable, but now it's easier to approximate.**

## Wasserstein GAN Approximation

## Wasserstein GAN Approximation

Note that if we replace the supremum over 1-Lipschitz functions with the supremum over $K$-Lipschitz functions, then the supremum is $K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta)$ instead.

## Wasserstein GAN Approximation

Note that if we replace the supremum over 1-Lipschitz functions with the supremum over $K$-Lipschitz functions, then the supremum is $K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta)$ instead.

Suppose we have a parametrized function family $\{f_w\}_{w \in W}$, where $w$ are the weights and $W$ is the set of all possible weights

Furthermore suppose these functions are all $K$-Lipschitz for some $K$.

## Wasserstein GAN Approximation

Note that if we replace the supremum over 1-Lipschitz functions with the supremum over $K$-Lipschitz functions, then the supremum is $K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta)$ instead.

Suppose we have a parametrized function family $\{f_w\}_{w \in W}$, where $w$ are the weights and $W$ is the set of all possible weights

Furthermore suppose these functions are all $K$-Lipschitz for some $K$. Then we have

$$\begin{aligned} \max_{w \in W} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f_w(x)] &\leq \sup_{||f||_L \leq K} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)] \\ &= K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta) \end{aligned}$$

## Wasserstein GAN Approximation

Note that if we replace the supremum over 1-Lipschitz functions with the supremum over $K$-Lipschitz functions, then the supremum is $K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta)$ instead.

Suppose we have a parametrized function family $\{f_w\}_{w \in W}$, where $w$ are the weights and $W$ is the set of all possible weights

Furthermore suppose these functions are all $K$-Lipschitz for some $K$. Then we have

$$\begin{aligned}\max_{w \in W} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f_w(x)] &\leq \sup_{||f||_L \leq K} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)] \\ &= K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta)\end{aligned}$$

If $\{f_w\}_{w \in W}$ contains the true supremum among $K$-Lipschitz functions, this gives the distance exactly.

## Wasserstein GAN Approximation

Note that if we replace the supremum over 1-Lipschitz functions with the supremum over $K$-Lipschitz functions, then the supremum is $K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta)$ instead.

Suppose we have a parametrized function family $\{f_w\}_{w \in W}$, where $w$ are the weights and $W$ is the set of all possible weights

Furthermore suppose these functions are all $K$-Lipschitz for some $K$. Then we have

$$\begin{aligned} \max_{w \in W} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f_w(x)] &\leq \sup_{||f||_L \leq K} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)] \\ &= K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta) \end{aligned}$$

If $\{f_w\}_{w \in W}$ contains the true supremum among $K$-Lipschitz functions, this gives the distance exactly.

**In practice this won't be true!**

## Wasserstein GAN Algorithm

Looping all this back to generative models, we'd like to train $\mathbb{P}_\theta = g_\theta(z)$ to match $\mathbb{P}_r$.

## Wasserstein GAN Algorithm

Looping all this back to generative models, we'd like to train $\mathbb{P}_\theta = g_\theta(z)$ to match $\mathbb{P}_r$.

Intuitively, given a fixed $g_\theta$, we can compute the optimal $f_w$ for the Wasserstein distance.

## Wasserstein GAN Algorithm

Looping all this back to generative models, we'd like to train $\mathbb{P}_\theta = g_\theta(z)$ to match $\mathbb{P}_r$.

Intuitively, given a fixed $g_\theta$, we can compute the optimal $f_w$ for the Wasserstein distance.

We can then backpropagate through $W(\mathbb{P}_r, g_\theta(Z))$ to get the gradient for $\theta$.

$$\begin{aligned} \nabla_\theta W(\mathbb{P}_r, \mathbb{P}_\theta) &= \nabla_\theta(\mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim Z}[f_w(g_\theta(z))]) \\ &= -\mathbb{E}_{z\sim Z}[\nabla_\theta f_w(g_\theta(z))] \end{aligned}$$

## Wasserstein GAN Algorithm

The training process now has three steps:

## Wasserstein GAN Algorithm

The training process now has three steps:

1. For a fixed $\theta$, compute an approximation of $W(\mathbb{P}_r, \mathbb{P}_\theta)$ by training $f_w$ to convergence

## Wasserstein GAN Algorithm

The training process now has three steps:

1. For a fixed $\theta$, compute an approximation of $W(\mathbb{P}_r, \mathbb{P}_\theta)$ by training $f_w$ to convergence
2. Once we have the optimal $f_w$, compute the $\theta$ gradient $-\mathbb{E}_{z \sim Z}[\nabla_\theta f_w(g_\theta(z))]$ by sampling several $z \sim Z$

## Wasserstein GAN Algorithm

The training process now has three steps:

1. For a fixed $\theta$, compute an approximation of $W(\mathbb{P}_r, \mathbb{P}_\theta)$ by training $f_w$ to convergence
2. Once we have the optimal $f_w$, compute the $\theta$ gradient $-\mathbb{E}_{z\sim Z}[\nabla_\theta f_w(g_\theta(z))]$ by sampling several $z \sim Z$
3. Update $\theta$, and repeat the process

## Wasserstein GAN Algorithm

The training process now has three steps:

1. For a fixed $\theta$, compute an approximation of $W(\mathbb{P}_r, \mathbb{P}_\theta)$ by training $f_w$ to convergence
2. Once we have the optimal $f_w$, compute the $\theta$ gradient $-\mathbb{E}_{z\sim Z}[\nabla_\theta f_w(g_\theta(z))]$ by sampling several $z \sim Z$
3. Update $\theta$, and repeat the process

### Important Detail

The entire derivation only works when the function family $\{f_w\}_{w\in W}$ is $K$-Lipschitz.

## Wasserstein GAN Algorithm

The training process now has three steps:

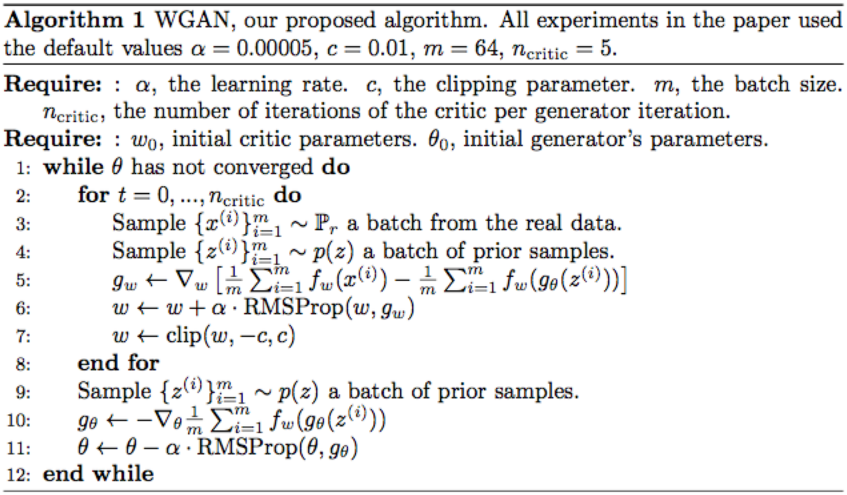
1. For a fixed $\theta$, compute an approximation of $W(\mathbb{P}_r, \mathbb{P}_\theta)$ by training $f_w$ to convergence
2. Once we have the optimal $f_w$, compute the $\theta$ gradient $-\mathbb{E}_{z\sim Z}[\nabla_\theta f_w(g_\theta(z))]$ by sampling several $z \sim Z$
3. Update $\theta$, and repeat the process

### Important Detail

The entire derivation only works when the function family $\{f_w\}_{w\in W}$ is $K$-Lipschitz.

To guarantee this is true, the authors use weight clamping. The weights $w$ are constrained to lie within $[-c, c]$, by clipping $w$ after every update to $w$.

## Wasserstein GAN Algorithm



**Algorithm 1** WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

```
Require: : $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size.
    $n_{\text{critic}}$, the number of iterations of the critic per generator iteration.
Require: : $w_0$, initial critic parameters. $\theta_0$, initial generator's parameters.
 1: while $\theta$ has not converged do
 2:     for $t = 0, ..., n_{\text{critic}}$ do
 3:         Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
 4:         Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
 5:         $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
 6:         $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
 7:         $w \leftarrow \text{clip}(w, -c, c)$
 8:     end for
 9:     Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
10:     $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
11:     $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
12: end while
```

## Comparison with Standard GANs

## Comparison with Standard GANs

- In GANs, the discriminator maximizes

$$\frac{1}{m}\sum_{i=1}^{m}\log D(x^{(i)}) + \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(g_\theta(z^{(i)})))$$

where we constrain $D(x)$ to always be a probability $p \in (0, 1)$

## Comparison with Standard GANs

- In GANs, the discriminator maximizes

$$\frac{1}{m}\sum_{i=1}^{m}\log D(x^{(i)}) + \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(g_\theta(z^{(i)})))$$

  where we constrain $D(x)$ to always be a probability $p \in (0,1)$
- In WGANs, nothing requires the $f_w$ to output a probability and hence it is referred to as a critic instead of a discriminator

## Comparison with Standard GANs

- In GANs, the discriminator maximizes

$$\frac{1}{m}\sum_{i=1}^{m}\log D(x^{(i)}) + \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(g_\theta(z^{(i)})))$$

  where we constrain $D(x)$ to always be a probability $p \in (0, 1)$
- In WGANs, nothing requires the $f_w$ to output a probability and hence it is referred to as a critic instead of a discriminator
- Although GANs are formulated as a min max problem, in practice we never train $D$ to convergence

## Comparison with Standard GANs

- In GANs, the discriminator maximizes

$$\frac{1}{m}\sum_{i=1}^{m}\log D(x^{(i)}) + \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(g_\theta(z^{(i)})))$$

  where we constrain $D(x)$ to always be a probability $p \in (0, 1)$
- In WGANs, nothing requires the $f_w$ to output a probability and hence it is referred to as a critic instead of a discriminator
- Although GANs are formulated as a min max problem, in practice we never train $D$ to convergence
- Consequently, we're updating $G$ against an objective that kind of aims towards the JS divergence, but doesn't go all the way

## Comparison with Standard GANs

- In GANs, the discriminator maximizes

$$\frac{1}{m}\sum_{i=1}^{m}\log D(x^{(i)}) + \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(g_\theta(z^{(i)})))$$

  where we constrain $D(x)$ to always be a probability $p \in (0, 1)$
- In WGANs, nothing requires the $f_w$ to output a probability and hence it is referred to as a critic instead of a discriminator
- Although GANs are formulated as a min max problem, in practice we never train $D$ to convergence
- Consequently, we're updating $G$ against an objective that kind of aims towards the JS divergence, but doesn't go all the way
- In constrast, because the Wasserstein distance is differentiable nearly everywhere, we can (and should) train $f_w$ to convergence before each generator update, to get as accurate an estimate of $W(\mathbb{P}_r, \mathbb{P}_\theta)$ as possible

## Properties of optimal WGAN critic

## Properties of optimal WGAN critic

An open question is how to effectively enforce the Lipschitz constraint on the critic?

## Properties of optimal WGAN critic

An open question is how to effectively enforce the Lipschitz constraint on the critic?

Previously (Arjovksy et. al 2017) we've seen that you can clip the weights of the critic to lie within the compact space $[-c, c]$

## Properties of optimal WGAN critic

An open question is how to effectively enforce the Lipschitz constraint on the critic?

Previously (Arjovksy et. al 2017) we've seen that you can clip the weights of the critic to lie within the compact space $[-c, c]$

To understand why weight clipping is problematic in WGAN critic we need to understand what are the properties of the optimal WGAN critic?

## Properties of optimal WGAN critic

## Properties of optimal WGAN critic

If the optimal critic under the Kantorovich-Rubinstein dual $D^*$ is differentiable, and $x$ is a point from our generator distribution $\mathbb{P}_\theta$, then there is a point $y$ sampled from the true distribution $\mathbb{P}_r$ such that the gradient of $D^*$ at all points $x_t = (1-t)x + ty$ lie on a straight line between x and y.

## Properties of optimal WGAN critic

If the optimal critic under the Kantorovich-Rubinstein dual $D^*$ is differentiable, and $x$ is a point from our generator distribution $\mathbb{P}_\theta$, then there is a point $y$ sampled from the true distribution $\mathbb{P}_r$ such that the gradient of $D^*$ at all points $x_t = (1-t)x + ty$ lie on a straight line between x and y.

In other words, $\nabla D^*(x_t) = \frac{y - x_t}{||y - x_t||}$.

## Properties of optimal WGAN critic

If the optimal critic under the Kantorovich-Rubinstein dual $D^*$ is differentiable, and $x$ is a point from our generator distribution $\mathbb{P}_\theta$, then there is a point $y$ sampled from the true distribution $\mathbb{P}_r$ such that the gradient of $D^*$ at all points $x_t = (1-t)x + ty$ lie on a straight line between x and y.

In other words, $\nabla D^*(x_t) = \frac{y - x_t}{||y - x_t||}$.

**This implies that the optimal WGAN critic has gradients with norm 1 almost everywhere under $\mathbb{P}_r$ and $\mathbb{P}_\theta$**

## Gradient penalty

## Gradient penalty

We consider an alternative method to enforce the Lipschitz constraint on the training objective.

## Gradient penalty

We consider an alternative method to enforce the Lipschitz constraint on the training objective.

A differentiable function is 1-Lipschitz if and only if it has gradients with norm less than or equal to 1 everywhere.

## Gradient penalty

We consider an alternative method to enforce the Lipschitz constraint on the training objective.

A differentiable function is 1-Lipschitz if and only if it has gradients with norm less than or equal to 1 everywhere.

This implies we should directly constrain the gradient norm of our critic function with respect to its input.

## Gradient penalty

We consider an alternative method to enforce the Lipschitz constraint on the training objective.

A differentiable function is 1-Lipschitz if and only if it has gradients with norm less than or equal to 1 everywhere.

This implies we should directly constrain the gradient norm of our critic function with respect to its input.

Enforcing a soft version of this we get:

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}} \left[ (\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]}_{\text{Our gradient penalty}}$$

## WGAN with gradient penalty

**Algorithm 1** WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0.5$, $\beta_2 = 0.9$.

```
Require: The gradient penalty coefficient $\lambda$, the number of critic iterations per generator iteration
    $n_{\text{critic}}$, the batch size $m$, Adam hyperparameters $\alpha, \beta_1, \beta_2$.
Require: initial critic parameters $w_0$, initial generator parameters $\theta_0$.
 1: while $\theta$ has not converged do
 2:     for $t = 1, ..., n_{\text{critic}}$ do
 3:         for $i = 1, ..., m$ do
 4:             Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0,1]$.
 5:             $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
 6:             $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
 7:             $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
 8:         end for
 9:         $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
10:     end for
11:     Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
12:     $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
13: end while
```