A Two-Step Computation of the Exact GAN Wasserstein Distance

Huidong Liu [1] **Xianfeng Gu** [1] **Dimitris Samaras** [1]

Abstract

In this paper, we propose a two-step method to compute the Wasserstein distance in Wasserstein Generative Adversarial Networks (WGANs): 1) The convex part of our objective can be solved by linear programming; 2) The non-convex residual can be approximated by a deep neural network. We theoretically prove that the proposed formulation is equivalent to the discrete Monge-Kantorovich dual formulation. Furthermore, we give the approximation error bound of the Wasserstein distance and the error bound of generalizing the Wasserstein distance from discrete to continuous distributions. Our approach optimizes the exact Wasserstein distance, obviating the need for weight clipping previously used in WGANs. Results on synthetic data show that the our method computes the Wasserstein distance more accurately. Qualitative and quantitative results on MNIST, LSUN and CIFAR-10 datasets show that the proposed method is more efficient than state-of-the-art WGAN methods, and still produces images of comparable quality.

1. Introduction

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) are deep generative models that have been receiving increased attention. GANs have been widely used in image inpainting (Yeh et al., 2017), semantic segmentation (Zhu & Xie, 2016; Nguyen et al., 2017), face editing (Liu & Tuzel, 2016; Shu et al., 2017), etc. GANs use a generator to generate synthetic samples from a simple low dimensional distribution, and then feed the generated samples together with real samples to a discriminator, or critic, that maximizes the probability of real samples and minimizes the probability of synthetic samples.

GANs are widely used and unfortunately, are known to be hard to train. GANs (Goodfellow et al., 2014) employ the Jensen-Shannon (JS) divergence to measure the distance between the distribution of real data and generated data. The JS divergence is discontinuous w.r.t. the generator's parameters (Arjovsky et al., 2017). Other probability measures, such as Kullback-Leibler (KL) divergence, Total Variation (TV) distance, or Pearson χ^2 divergence used in Least Square GANs (Mao et al., 2017), also suffer from this drawback (Arjovsky et al., 2017). Fedus et al. show that minimizing a non-saturating objective for the generator can handle this problem. Roth et al. propose regularization techniques to address this issue. BGANs (Hjelm et al., 2018) apply the importance weights on generated samples to train the generator. Popular methods that tackle this problem are based on i) Mean Maximum Discrepancy (MMD) (Gretton et al., 2012), and ii) the Wasserstein distance, employing the Integral Probability Metrics (IPMs) (Müller, 1997; Mroueh & Sercu, 2017); however they need to restrict the functions in a constrained set. MMD-based methods like MMD-NET (Dziugaite et al., 2015) and MMD GANs (Bińkowski et al., 2018) need to select suitable kernel parameters, but it is very hard to set kernel parameters for better results. Wasserstein distance-based methods, such as WGAN (Arjovsky et al., 2017), McGAN (Mroueh et al., 2017), MMD-GAN (Li et al., 2017), RWGAN (Guo et al., 2017), etc., need weight clipping. WGAN-GP (Gulrajani et al., 2017), and Spectral Normalization (SN) (Miyato et al., 2018) (with Wasserstein distance) apply regularizations on the critic trying to make the critic 1-Lipschitz, but they fail to optimize the true Wasserstein distance.

The objectives of Wasserstein distance-based GANs are, in essence, practical implementations of the Monge-Kantorovich dual formulation (Villani, 2008). The solution to the Monge-Kantorovich primal formulation is a joint distribution, represented as a 2D array, and the solution to the dual formulation is a function. It is easier for a Deep Neural Network (DNN) to represent a function for the discriminator. However, solving the Monge-Kantorovich dual formulation is still an open problem in practice. WGAN, RWGAN, etc., use weight clipping to ensure the 1-Lipschitz condition of the Kantorovich potential, such that the Monge-Kantorovich dual formulation is simpler and easier to optimize. How-

[1] Department of Computer Science, Stony Brook University, New York, USA. Correspondence to: Huidong Liu <huidliu@cs.stonybrook.edu>, Xianfeng Gu <gu@cs.stonybrook.edu>, Dimitris Samaras <samaras@cs.stonybrook.edu>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

ever, weight clipping limits the critic's functional space and can cause gradients in the critic to explode or vanish if the clipping parameters are not carefully chosen (Gulrajani et al., 2017). There is no guarantee that the weight penalty based method WGAN-GP optimizes the true Wasserstein distance. Spectral Normalization with Wasserstein Distance (SN-WD) sets the upper bound of the Lipschitz constant of the critic to be 1, but it is not guaranteed that the Lipschitz constant can reach 1 to compute the Wasserstein distance.

In summary, this paper has four main contributions:

- We propose a new formulation to compute the Wasserstein distance. The proposed formulation can be solved in a Two-Step fashion, i.e., 1) apply linear programming to solve the convex part of the formulation to get an exact computation of the Wasserstein distance and 2) adopt a deep neural network to solve the remaining non-convex part to get an approximate solution. We theoretically prove that the new proposed formulation is equivalent to the Monge-Kantorovich dual formulation, and thus, we are optimizing the exact Wasserstein distance. We name our method TS.

- We compute the approximation error bound of the Wasserstein distance using the proposed TS method. In addition, we also compute the error bound of generalizing the Wasserstein distance from the discrete to the continuous case.

- We apply the TS method for computing the Wasserstein distance in WGAN. We name our GAN method WGAN-TS. Instead of minimizing the Wasserstein distance with weight clipping or a weight penalty, which requires hyper-parameters that are hard to tune, the critic in WGAN-TS does not need any hyper-parameters to enforce weight constraints.

- Results on the eight-Gaussian synthetic dataset show that the our method computes a more accurate Wasserstein distance than WGAN, WGAN-GP and SN-WD. Results on the MNIST, LSUN and CIFAR-10 datasets show that WGAN-TS is comparable to WGAN-GP and SN-WD. Furthermore, the proposed WGAN-TS is faster than other WGAN methods under the commonly used batch size of 64, improving the performance of the previous fastest WGAN method by more than 30%.

2. Optimal Mass Transport

The goal of Optimal Mass Transport (OMT) is to transform one distribution into another and minimize the total transport cost. This minimized cost is the Wasserstein distance.

Since solving the original OMT problem is hard, Kantorovich relaxed the original transport mapping problem into a transport plan problem. The transport map is a special case of the transport plan and for each of them we can define the Wasserstein distance. The solution to the Monge-Kantorovich formulation for computing Wasserstein distances is a transport plan table which is discontinuous w.r.t. the generator's parameters and cannot be used as the discriminator in GANs. In contrast, the solution to the Monge-Kantorovich dual formulation is a continuous function and thus better for machine learning.

2.1. The Monge-Kantorovich Dual Formulation

The Monge-Kantorovich dual problem (Villani, 2008) is:

Problem 1. *Suppose X and Y are two bounded domains in $\mathbb{R}^n$. Given two probability measures $\mu \in \mathbb{P}(X)$, $\nu \in \mathbb{P}(Y)$, and a cost function $c : X \times Y \mapsto [0, +\infty]$. Find functions ϕ and ψ such that*

$$C(\mu, \nu) = \sup_{\phi-\psi \leq c} \left\{ \int \phi(y) d\nu(y) - \int \psi(x) d\mu(x) \right\} \quad (1)$$

where $C(\mu,\nu)$ is the Wasserstein distance between μ and ν for transport plan problem.

The Kantorovich duality Theorem (Villani, 2008) further transforms the above problem into the following problem:

Problem 2. *Find a function ψ such that*

$$C(\mu, \nu) = \sup_{\psi} \left\{ \int \psi^c(y) d\nu(y) - \int \psi(x) d\mu(x) \right\} \quad (2)$$

where $C(\mu,\nu)$ is the Wasserstein distance between μ and ν and ψ^c is the c-transform of ψ defined below:

$$\forall y \in Y \qquad \psi^c(y) = \inf_{x \in X} \big(\psi(x) + c(x, y)\big) \quad (3)$$

When training GANs, we have empirical distributions. So, we are interested in the discrete case of Problem 2. Assume $\hat{X} = \{x_j\}_{j \in \mathcal{J}}$ sampled from μ and $\hat{Y} = \{y_i\}_{i \in \mathcal{I}}$ sampled from ν, where $\mathcal{I}$ and $\mathcal{J}$ are disjoint index sets. Let $m = |\mathcal{I}|$ and $n = |\mathcal{J}|$ be the numbers of elements in the two sets.

Problem 3. *(Discrete Case of Problem 2) Let*

$$\hat{d}(\psi) = \frac{1}{m} \sum_{i \in \mathcal{I}} \psi^c(y_i) - \frac{1}{n} \sum_{j \in \mathcal{J}} \psi(x_j) \quad (4)$$

Find a function ψ such that $\hat{C}(\mu, \nu) = \sup_{\psi} \hat{d}(\psi)$ where $\hat{C}(\mu, \nu)$ is the Wasserstein distance between μ and ν and ψ^c is the c-transform of ψ defined below:

$$\forall y_i \in \hat{Y} \qquad \psi^c(y_i) = \inf_{x \in \hat{X}} \big(\psi(x) + c(x, y_i)\big) \quad (5)$$

In order to make $\psi^c = \psi$ to simplify this problem, the WGAN restricts function ψ to be 1-Lipschitz (Arjovsky et al., 2017). We claim that we do not need to explicitly restrict ψ to be 1-Lipschitz during optimization. In the next section, we propose a new formulation of the Monge-Kantorovich duality that optimizes the exact Wasserstein distance.

3. A New Formulation of the Monge-Kantorovich Duality

3.1. An Equivalent Monge-Kantorovich Dual Formulation

Problem 4 is a new formulation of the Monge-Kantorovich formulation. Solving Problem 4 is equivalent to solving Problem 3 under a mild assumption that the cost function $c(\cdot,\cdot)$ satisfies the triangle inequality in Lemma 3.1.

Problem 4. *Solve the following problem:*

$$\begin{aligned} \max_f \quad & \hat{h}(f) = \left\{ \frac{1}{m}\sum_{i\in\mathcal{I}} f(y_i) - \frac{1}{n}\sum_{j\in\mathcal{J}} f(x_j) \right\} \\ \text{s.t.} \quad & f(y_i) - f(x_j) \le c(x_j, y_i), \;\; \forall j \in \mathcal{J}, \; \forall i \in \mathcal{I} \end{aligned}$$

In order to prove that Problem 3 and Problem 4 are equivalent, we introduce Lemma 3.1 and Lemma 3.2.

Lemma 3.1. *If the cost function $c(\cdot,\cdot)$ satisfies the triangle inequality, i.e., $c(x,y)+c(y,z) \ge c(x,z), \forall x,y,z,$ then $\forall x_j \in \hat{X}$, $\forall y_i \in \hat{Y}$, if $x_j = y_i$, and ψ^* is the optimizer to Problem 3, then $(\psi^c)^*(y_i) = \psi^*(x_j)$, where $(\psi^c)^*(y_i) = \inf_{x\in\hat{X}}\left(\psi^*(x) + c(x, y_i)\right)$.*

The proof is in the supplemental materials.

Lemma 3.2. *Suppose f^* is an optimizer to Problem 4, then i) $f^*(y) = \inf_{x\in\hat{X}}\{f^*(x) + c(x,y)\}$, $\forall y \in \hat{Y}$, and ii) $f^*(x) = \sup_{y\in\hat{Y}}\{f^*(y) - c(x,y)\}$, $\forall x \in \hat{X}$*

The proof is in the supplemental materials.

Next we state that Problem 3 and Problem 4 are equivalent.

Theorem 3.3. *If the cost function $c(\cdot,\cdot)$ satisfies the triangle inequality, then solving Problem 4 is equivalent to solving Problem 3, i.e., the optimal objectives of Problem 3 and 4 are equal and $f^*(x_j) = \psi^*(x_j)$ and $f^*(y_i) = (\psi^c)^*(y_i)$.*

Proof. First, we prove that any optimal solution to Problem 4 is a feasible solution to Problem 3. Suppose f^* is the optimal solution to Problem 4, from Lemma 3.2 we know that $f^*(y) = \inf_{x\in\hat{X}}\{f^*(x)+c(x,y)\}$, $\forall y \in \hat{Y}$, and from the definition of c-transform in Eq. (5) we have $(f^*)^c(y) = \inf_{x\in\hat{X}}\{f^*(x)+c(x,y)\}$, $\forall y \in \hat{Y}$. Hence, f^* is a feasible solution to Problem 3. Therefore, $\hat{d}(\psi^*) \ge \hat{h}(f^*)$.

Then, we prove that any optimal solution to Problem 3 is a feasible solution to Problem 4. Suppose ψ^* is an optimizer to Problem 3. According to Lemma 3.1, for any $x_j = y_i$, we have $(\psi^c)^*(y_i) = \psi^*(x_j)$ given that the cost function $c(\cdot,\cdot)$ satisfies the triangle inequality. Therefore, we can find a function $\phi(x_j) = \psi^*(x_j)$ and $\phi(y_i) = (\psi^c)^*(y_i)$. So, $\hat{d}(\psi^*)$ in Eq. (4) can be rewritten as

$$\hat{d}(\psi^*) \;=\; \frac{1}{m}\sum_{i\in\mathcal{I}}\phi(y_i) - \frac{1}{n}\sum_{j\in\mathcal{J}}\phi(x_j) \tag{6}$$

From the definition of $(\psi^c)^*(y_i)$, we have $\phi(y_i) - \phi(x_j) \le c(x_j, y_i)$. Therefore, ϕ is a feasible solution to Problem 4, and hence $\hat{d}(\psi^*) \le \hat{h}(f^*)$. So, we have $\hat{d}(\psi^*) = \hat{h}(f^*)$. Therefore, ϕ is also an optimal solution to Problem 4. So $\phi = f^*$, and hence $f^*(x_j) = \psi^*(x_j)$ and $f^*(y_i) = (\psi^c)^*(y_i)$.

□

3.2. Solving the Monge-Kantorovich Dual Formulation

Directly solving the Monge-Kantorovich Dual problem (Problem 4) is difficult. We propose to solve it in two steps; First, a linear programming problem, and second, a deep regression problem.

Step 1: Solve the following linear programming problem:

$$\begin{aligned} \max_T \quad & \frac{1}{m}\sum_{i\in\mathcal{I}} T_i - \frac{1}{n}\sum_{j\in\mathcal{J}} T_j \\ \text{s.t.} \quad & T_i - T_j \le c_{ij}, \;\; \forall i \in \mathcal{I}, \; \forall j \in \mathcal{J} \end{aligned} \tag{7}$$

Optimal solutions to formula (7) are unique up to a scalar. Suppose T^* is the optimizer to formula (7), then $T^* + C$ is also the optimizer to formula (7), where C is a constant scalar. After we find an optimal solution to (7) we subtract the mean of T^* by $T^*_t \leftarrow T^*_t - (\sum_{k\in\mathcal{I}\cup\mathcal{J}} T^*_k)/(m+n), \forall t \in \mathcal{I}\cup\mathcal{J}$.

Step 2: After solving the linear programming problem, we optimize the following regression problem:

$$\min_f \quad \frac{1}{m+n}\left(\sum_{i\in\mathcal{I}}\left(f(y_i) - T^*_i\right)^2 + \sum_{j\in\mathcal{J}}\left(f(x_j) - T^*_j\right)^2\right) \tag{8}$$

We use a DNN to parameterize the function f. If the DNN has enough parameters, the deep regression problem can be exactly solved. We name our Two-Step method to compute the Wasserstein distance, TS.

Intuitively, Step 1 can compute the Wasserstein distance exactly, however the result is not differentiable. The DNN of Step 2 is trained to take the same inputs and regress the values produced by Step 1, thus providing a differentiable approximation of the Wasserstein distance.

3.3. Exactness of the Wasserstein Distance

Our algorithm computes the exact Wasserstein distance by the duality in linear programming. The original Monge-Kantorovich problem in the continuous case is:

Problem 5. *Find a transport plan to minimize the following total cost:*

$$C(\pi^*) = \inf_{\pi \in \Pi(\mu,\nu)} \int_{X \times Y} c(x,y) d\pi(x,y) \qquad (9)$$

where $\Pi(\mu,\nu)$ is the set of transport plans defined as:

$$\Pi(\mu,\nu) = \{\pi \in \mathbb{P}(X \times Y) : \pi_x = \mu, \pi_y = \nu\} \qquad (10)$$

where π_x and π_y are marginal distributions of π on X and Y, respectively.

Next, we introduce the discrete case of the Monge-Kantorovich problem. Suppose $\hat{Y}$ is the set of real data samples, and $\hat{X}$ is the set of the generated samples. Each y_i has a Dirac measure of $1/m$, and each x_j of $1/n$. In practice, the data sample distribution and the generated sample distribution are formulated as:

$$\mu = \frac{1}{n} \sum_{i \in \mathcal{I}} \delta(x - x_j), \;\; \nu = \frac{1}{m} \sum_{j \in \mathcal{J}} \delta(y - y_i)$$

The ℓ_1 distance between x_j and y_i is $c_{ij} = ||x_j - y_i||_1$, γ_{ij} is the mass transported from x_j to y_i, namely the joint distribution between μ and ν. The discrete Monge-Kantorovich problem is formulated as a linear programming problem:

$$\begin{aligned} \min_{\gamma} \quad & \sum_{ij} c_{ij}\gamma_{ij} \\ \text{s.t.} \quad & \sum_{j} \gamma_{ij} = 1/m, \; i \in \mathcal{I} \\ & \sum_{i} -\gamma_{ij} = -1/n, \; j \in \mathcal{J} \\ & \gamma_{ij} \geq 0, \; i \in \mathcal{I}, j \in \mathcal{J} \end{aligned} \qquad (11)$$

According to linear programming theory, a unique global optimal solution exists (Karmarkar, 1984), giving the exact Wasserstein distance between two Dirac measures $W(\mu,\nu)$. According to the duality in linear programming, the dual problem of (11) is formulated as:

$$\begin{aligned} \max_{T} \quad & \frac{1}{m} \sum_{i \in \mathcal{I}} T_i - \frac{1}{n} \sum_{j \in \mathcal{J}} T_j \\ \text{s.t.} \quad & T_i - T_j \leq c_{ij}, \; i \in \mathcal{I}, j \in \mathcal{J} \end{aligned} \qquad (12)$$

where T_i and T_j are the dual variables. (12) is exactly the same formula as formula (7). Therefore, our linear programing step computes the exact Wasserstein distance between the two empirical distributions.

3.4. Approximation Error Bound

Theorem 3.4. *If the optimization error of formula (8) is ϵ, i.e., ϵ-suboptimal, and supposing $m = n$, then the Wasserstein distance is bounded by $2\sqrt{\epsilon}$, i.e. $|\hat{h}(\hat{f}) - \hat{h}(f^*)| \leq 2\sqrt{\epsilon}$ where f^* is the optimal solution to formula (8) and $\hat{f}$ is the approximate solution to formula (8).*

Proof. Since f^* is the optimal solution to formula (8), $f^*(y_i) = T_i^*$ and $f^*(x_j) = T_j^*$, $\forall i \in \mathcal{I}, \forall j \in \mathcal{J}$. After the optimization of formula (8), the objective of formula (8) is ϵ and $m = n$ means that

$$\sum_{i \in \mathcal{I}} \left(\hat{f}(y_i) - f^*(y_i)\right)^2 + \sum_{j \in \mathcal{J}} \left(\hat{f}(x_j) - f^*(x_j)\right)^2 = 2m\epsilon$$

and

$$\begin{aligned} m|\hat{h}(\hat{f}) - \hat{h}(f^*)| \;\; & \leq \;\; \sum_{i \in \mathcal{I}} \left|\hat{f}(y_i) - f^*(y_i)\right| + \\ & \;\;\;\;\;\; \sum_{j \in \mathcal{J}} \left|\hat{f}(x_j) - f^*(x_j)\right| \\ & \leq \;\; 2m\sqrt{\epsilon} \end{aligned}$$

Therefore, we have $|\hat{h}(\hat{f}) - \hat{h}(f^*)| \leq 2\sqrt{\epsilon}$ □

Theorem 3.4 provides the theoretical guarantee of the Wasserstein distance w.r.t. the optimization error of formula (8).

3.5. Error Bounds of Generalizing the Wasserstein Distance from the Discrete to Continuous Case

The following problem is the continuous case of Problem 4:

Problem 6. *Solve the following problem:*

$$\begin{aligned} \max_{f} \quad & h(f) = \mathbb{E}_y[f(y)] - \mathbb{E}_x[f(x)] \\ \text{s.t.} \quad & f(y) - f(x) \leq c(x,y), \;\; \forall x \in X, \; \forall y \in Y \end{aligned}$$

In Problem 4, we use empirical data $\hat{X}$ and $\hat{Y}$ to compute the Kantorovich potential. $\hat{X}$ and $\hat{Y}$ are randomly sampled from two distributions μ and ν, respectively. We need to analyze the error bound of generalizing the Wasserstein distance from the discrete to the continuous case. The error bound reflects how statistically close the computed Wasserstein distance is from the discrete to the continuous case.

Theorem 3.5. *Suppose f is the optimal solution to formula (8) with 0 optimization error, i.e., 0-suboptimal. Let θ_{ij} be a Bernoulli random variable such that if $f(y_i) - f(x_j) > c(y_i, x_j)$ then $\theta_{ij} = 1$, otherwise $\theta_{ij} = 0$. Let $e = \mathbb{E}[\theta_{ij}]$ be the expectation of the probability that constraints in Problem 4 violate the inequality constraints. The error bound of the Wasserstein distance from the discrete to the continuous case is:*

$$P(|\hat{h}(f) - h(f)| > \epsilon) \leq 2\exp(-m\epsilon^2/2) + 2\exp(-n\epsilon^2/2)$$

The error bound of the constraint violation in Problem 4 is:

$$P\left(|e| > \epsilon\right) \leq 2\exp(-2mn\epsilon^2)$$

The proof of Theorem 3.5 can be found in the supplemental materials. Theorem 3.5 shows that the more data we use to compute the Wasserstein distance, the better the approximation of the Wasserstein distance in the continuous case.

4. Wasserstein GAN with TS (WGAN-TS)

In the previous section, we proposed Problem 4 to compute the Wasserstein distance. We proved in Theorem 3.3 that instead of using weight clipping (Arjovsky et al., 2017) or weight penalty methods (Gulrajani et al., 2017) that require additional hyper-parameters to approximate the Wasserstein distance, we can use a two-step method to optimize the exact Wasserstein distance that does not require any hyper-parameters to enforce weight constraints. Let D be the critic and G be the generator. Based on this new formulation and TS optimization method, we propose the WGAN-TS model as follows:

$$\begin{aligned}\min_G \max_D \quad & \hat{C}(f) = \left\{ \frac{1}{m}\sum_{i\in\mathcal{I}} D(y_i) - \frac{1}{n}\sum_{j\in\mathcal{J}} D(G(z_j)) \right\} \\ \text{s.t.} \quad & D(y_i) - D(G(z_j)) \le c(y_i, G(z_j)),\ \ \forall i,\ \forall j\end{aligned} \tag{13}$$

where $c(y_i, G(z_j)) = ||y_i - G(z_j)||_1$ (ℓ_1 distance) or $c(y_i, G(z_j)) = ||y_i - G(z_j)||_2$ (ℓ_2 distance). When the generator G is fixed, we let $x_j = G(z_j)$ and we apply the proposed TS method to optimize formula (13) to compute the discriminator. After we optimize the discriminator D, we fix it and update the generator. We compute the generator loss as follows:

$$\min_G \quad -\frac{1}{n}\sum_{j\in\mathcal{J}} D(G(z_j)) \tag{14}$$

4.1. Weight Scaling

In optimizing GANs, usually the critic is not completely optimized in every generator iteration. We found in practice that satisfying the constraints in formula (13) is more important than purely maximizing the objective while leaving the constraints unsatisfied. Therefore, we propose to scale the discriminator after the discriminator's update so that the constraints in formula (13) are satisfied. The scaling factor β is defined as follows:

$$\beta = \sup\{1, \sup\{\frac{D(y_i) - D(G(z_j))}{||y_i - G(z_j)||_1}, ||y_i - G(z_j)||_1 > 0\}\} \tag{15}$$

After β is calculated, we apply the scaling factor to the weights of the layers in the discriminator such that $D_{w/\beta}(\cdot) = D_w(\cdot)/\beta$. The activation function after the layer performing the scaling should be a ReLU or a leakyReLU to allow scaling.

Algorithm 1 WGAN-TS

```
1:  Input: Real data $Y$, batch size $m$, $n_c = 1$, $n_r = 5$,
    Adam parameters, $\alpha, \beta_1, \beta_2$
2:  Output: $G$, $D$
3:  while $\theta$ has not converged do
4:    for $t_c = 0$ to $n_c$ do
5:      Sample $\{y_i\}_{i\in\mathcal{I}} \sim \mathbb{P}_r$ from real data.
6:      Sample $\{z_j\}_{j\in\mathcal{J}} \sim \mathbb{P}_z$ random noises.
7:      Let $x_j = G(z_j), \forall j \in \mathcal{J}$.
8:      Solve the Linear Programming problem in Eq. (7)
        using $c_{ij} = ||y_i - x_j||_1$, and obtain $T^*$.
9:      $T_t^* \leftarrow T_t^* - (\sum_{k\in\mathcal{I}\cup\mathcal{J}} T_k^*)/(m+n), \forall t \in \mathcal{I}\cup\mathcal{J}$.
10:     for $t_r = 0$ to $n_r$ do
11:       $g_w \leftarrow \nabla_w \frac{1}{m+n}(\sum_{i\in\mathcal{I}}(D_w(y_i) - T_i^*)^2 + \sum_{j\in\mathcal{J}}(D_w(x_j) - T_j^*)^2)$
12:       $w \leftarrow$ Adam($g_w$, $w$, $\alpha$, $\beta_1$, $\beta_2$)
13:     end for
14:     Perform weight scaling on $D$ according to Eq. (15)
15:   end for
16:   $g_\theta \leftarrow \nabla_\theta - \frac{1}{n}\sum_{j\in\mathcal{J}} D(G_\theta(z_j))$
17:   $\theta \leftarrow$ Adam($g_\theta$, $\theta$, $\alpha$, $\beta_1$, $\beta_2$)
18: end while
```

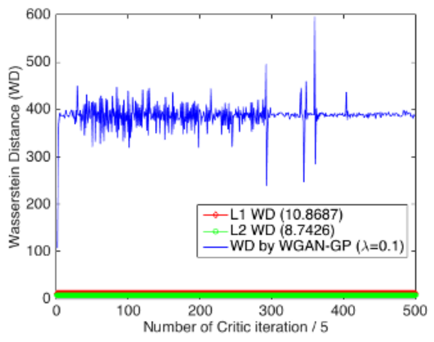
The weight scaling operation of the discriminator will not affect the gradient direction of the discriminator. Therefore, to the generator, weight scaling is equivalent to dynamically adjusting the learning rate of the generator.

4.2. Algorithm

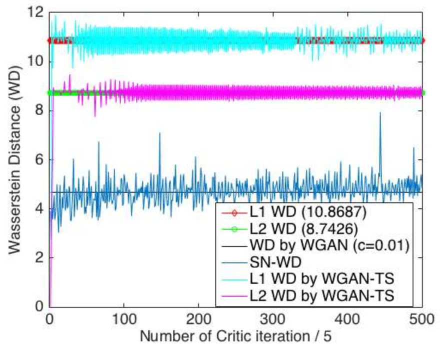
The algorithm of the proposed WGAN-TS is presented in Algorithm 1. In our algorithm, we set the number of critic iterations n_c to 1 but set the number of iterations of the deep regression part n_r to 5. We use ℓ_1 distance for WGAN-TS. Similar to WGAN-GP, we use Adam (Kingma & Ba, 2015) as our optimizer.

5. Experiments

In this section, we will first show on a toy data set that the computed Wasserstein Distance (WD) of the proposed method WGAN-TS is more accurate than the WGAN (Arjovsky et al., 2017), WGAN-GP (Gulrajani et al., 2017) and Spectral Normalization (Miyato et al., 2018) with Wasserstein Distance (SN-WD). In the second part, we will show that WGAN-TS produces better images than WGAN (Arjovsky et al., 2017) and FisherGAN (Mroueh & Sercu, 2017), and produces images comparable to WGAN-GP (Gulrajani et al., 2017) and SN-WD (Miyato et al., 2018) on the MNIST (LeCun et al., 1998), LSUN (Zhang et al., 2015) and CIFAR-10 (Krizhevsky & Hinton, 2009) datasets. WGAN-TS is also more efficient than the compared methods.

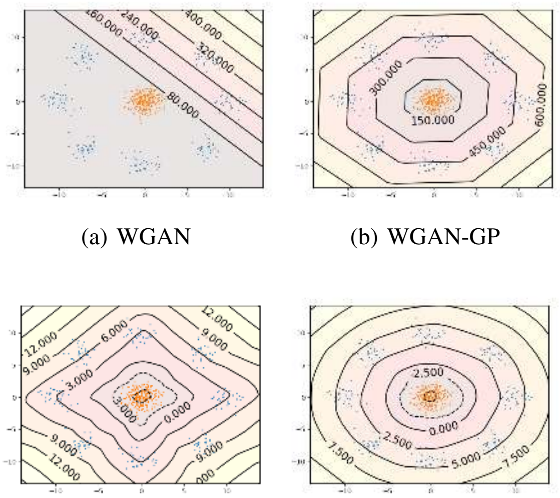


(a)



(b)

Figure 1. Wasserstein Distance (WD) on the 8 Gaussian toy dataset. (a) Accurate ℓ_1 and ℓ_2 WD, WD by WGAN-GP. (b) Accurate ℓ_1 and ℓ_2 WD, WD by WGAN, WD by SN-WD, ℓ_1 and ℓ_2 WD by WGAN-TS (Best viewed in color).



(a) WGAN (b) WGAN-GP

(c) WGAN-TS-ℓ_1 (d) WGAN-TS-ℓ_2

Figure 2. Value surfaces of critics computed by (a) WGAN, (b) WGAN-GP, (c) WGAN-TS-ℓ_1 and (d) WGAN-TS-ℓ_2.

5.1. Results on the Eight Gaussian Toy Dataset

Dataset: Similar to (Gulrajani et al., 2017), to simulate the distribution of real data, we generate 8 Gaussians as the real data distribution, and one Gaussian as the synthetic data distribution. Details of this dataset are in the supplemental materials. Once all the data points of real and synthetic data are generated, we use the same data for all methods.

First, we compute the WD between the real and synthetic distributions based on the Monge-Kantorovich primal form, which is a linear programming problem and has a global solution. The WD is 10.8687 under ℓ_1 distance and 8.7426 under ℓ_2 distance. We approximate the WD using different methods: WGAN, WGAN-GP, SN-WD, WGAN-TS with ℓ_1 distance (WGAN-TS-ℓ_1) and WGAN-TS with ℓ_2 distance (WGAN-TS-ℓ_2). The critics of all compared methods are the same. It is a (2,512)-ReLU-(512,512)-ReLU-(512,512)-ReLU-(512,1) network. The parameter settings of all the methods can be found in the supplemental material. Figure 1 plots the WD of different methods w.r.t. the critic's iterations. From Figure 1(a) we can see that the WGAN-GP does not approximate well either the ℓ_1 WD or the ℓ_2 WD, and has high variance. This means that WGAN-GP is not really computing WD and it is not stable. In Figure 1(b), our method WGAN-TS-ℓ_1 approximates the ℓ_1 WD very well and WGAN-TS-ℓ_2 approximates the ℓ_2 WD very well. WGAN computes a WD of approximately 4.6, which is quite far from the true ℓ_1 WD. SN-WD also computes a WD of approximately 4.6, which is quite far from either the ℓ_1 WD or ℓ_2 WD.

Figure 2 plots the value surfaces of the critics of WGAN, WGAN-GP, WGAN-TS-ℓ_1 and WGAN-TS-ℓ_2. From Figure 2(a) we can see that the critic of the WGAN cannot provide correct gradients which point from the synthetic to the real data. Figure 2(b) shows the value surfaces of WGAN-GP's critic. Although the critic provides good gradients for its generator to update, the computed values are significantly further from the true Wasserstein distance.

Experiments on the toy dataset validate that our method WGAN-TS can approximate the WD much more accurately than WGAN, WGAN-GP and SN-WD.

5.2. Results on MNIST, LSUN and CIFAR-10 datasets

5.2.1. Parameter settings

For images from the MNIST and CIFAR-10 datasets, we resize the image size to 64×64 so that on all the three datasets we can use the standard DCGAN (Radford et al., 2016) as the discriminator and generator. On all the three datasets, all parameters of each method are set the same. The batch size is set to 64 for all methods in all experiments. The dimension of the latent vector is set to 100 for all methods.

Table 1. Inception scores on the MNIST and CIFAR-10 datasets.

METHOD	MNIST	CIFAR-10
WGAN	1.64 ± 0.09	2.77 ± 0.18
WGAN-GP	2.34 ± 0.19	2.99 ± 0.22
FISHERGAN	-	1.00 ± 0.00
SN-WD	2.22 ± 0.23	2.96 ± 0.18
WGAN-TS	$\mathbf{2.35 \pm 0.20}$	$\mathbf{3.13 \pm 0.15}$

The number of critic iterations n_c for all methods is set to 5, except WGAN-TS where the optimization iteration number n_r is set to 5 instead. We use the publicly available implementations of these methods and adopt the default parameter settings. We use RMSProp (Tieleman & Hinton, 2012) as the optimizer for critic and generator in the WGAN and FisherGAN, and set the learning rate is to $5e$-5. The weight clipping parameter c in the WGAN is set to 0.01. For the WGAN-GP and WGAN-TS, we use Adam as the optimizer. We set the learning rate to $1e$-4, $\beta_1 = 0.5$ and $\beta_2 = 0.999$. λ in the WGAN-GP is set to 10. ρ in the FisherGAN is set to $1e$-6 as suggested (Mroueh & Sercu, 2017). We use the ℓ_1 distance for the WGAN-TS.

5.2.2. MNIST DATASET

Figure 3 shows the images generated by different methods. On the MNIST dataset, the generators of all methods iterate 1260000 times. From the generated images by WGAN we can find that there are still some digits that do not appear fully generated. This is mainly because weight clipping limits the critic's functional space so that the critic is not strong enough to distinguish between the real and synthetic data. Digits generated by the WGAN-GP, SN-WD and WGAN-TS have better image quality. The digits are smoother and more similar to real handwritten digits. To illustrate our point, we place red boxes around the generated images for the number 8. Figure 3 shows that images of the number 8 generated by the WGAN-GP, SN-WD, and WGAN-TS are more realistic and have fewer defects.

Table 1 compares the Inception Scores (IS) (Salimans et al., 2016) of different methods. It shows that the proposed WGAN-TS is much better than the WGAN, and is comparable to the state-of-the-art WGAN methods, WGAN-GP and SN-WD.

5.2.3. LSUN DATASET

On the LSUN dataset, we let the the generators of all methods iterate 500000 times. Figure 4 shows images generated by the WGAN, WGAN-GP, SN-WD and WGAN-TS. From this figure we can see that images generated by the WGAN are noisier than images generated by the WGAN-TS. We place red boxes around the images that we can recognize as

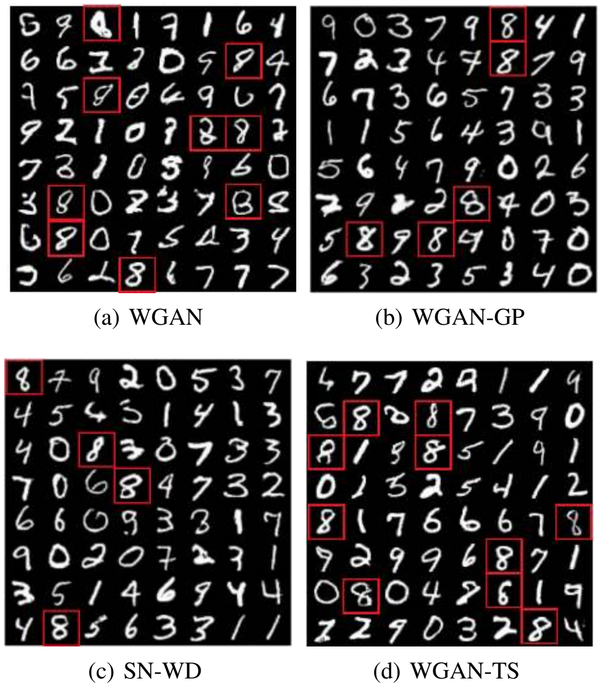


Figure 3. On the MNIST dataset, digits generated by (a) WGAN, (b) WGAN-GP, (c) SN-WD, (d) WGAN-TS. WGAN-GP, SN-WD and WGAN-TS generate more realistic images of digit 8. (Higher resolution comparisons for the selected images can be found in the supplemental materials.)

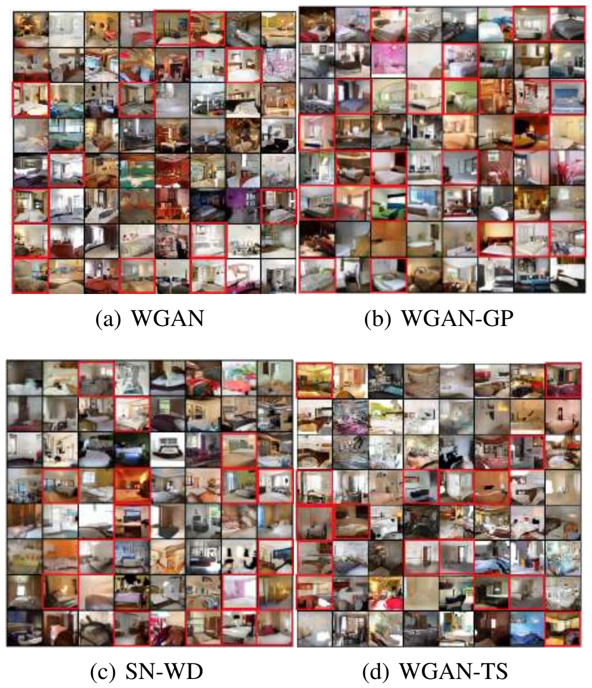


Figure 4. On the LSUN dataset, images generated by (a) WGAN, (b) WGAN-GP (c) SN-WD and (d) WGAN-TS. We mark the images that we can recognize as bedrooms with red boxes.

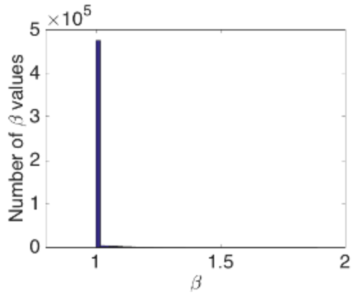


Figure 5. Histogram of β.

bedrooms. The WGAN-TS generates more number of images that are recognizable than the WGAN, and comparable number of images by the WGAN-GP and SN-WD.

Figure 5 shows the histogram of β in WGAN-TS on the LSUN dataset. "$\beta = 1$" means that the constraints in linear programming step are satisfied. Approximately 94% of beta values are 1, meaning that in most cases the constraints are satisfied.

5.2.4. CIFAR-10 DATASET

On the CIFAR-10 dataset, we let the generator of all the methods iterate 400000 times, and we use the IS to measure the quality of the generated images. Table 1 lists the IS of different methods. Note that as we use the standard DCGAN as the discriminator and generator, which is relatively shallow compared to ResNet (He et al., 2016), and the image size of this dataset is resized to 64×64, the inception scores reported here are lower than the best reported scores. In Table 1, the FisherGAN gives the lowest IS. The WGAN-TS, WGAN-GP and SN-WD achieve the highest ISs. The IS of the WGAN is slightly lower than the top three ISs. Figure 6 shows the images generated by these methods. From this figure we can see that images generated by the WGAN and WGAN-GP are slightly blurrier. We place red boxes around the objects in these figures that we believe are the most recognizable. From the selected images that were generated by the same method, we see that the WGAN-TS, WGAN-GP and SN-WD produce comparable numbers of recognizable images while the WGAN produces the least number of recognizable images.

We also evaluate the time used by the critics for each method. Since the generator update is the same for all the methods, the time spent by the critic determines the running time order per generator update. Table 2 presents the running time of the critic update per generator update. The proposed WGAN-TS is the most efficient one. Not surprisingly, the WGAN-GP is the slowest, because it needs to backpropagate twice to compute the gradient of the critic. The FisherGAN and SN-WD are comparable, and both of them are slower than the WGAN-TS.

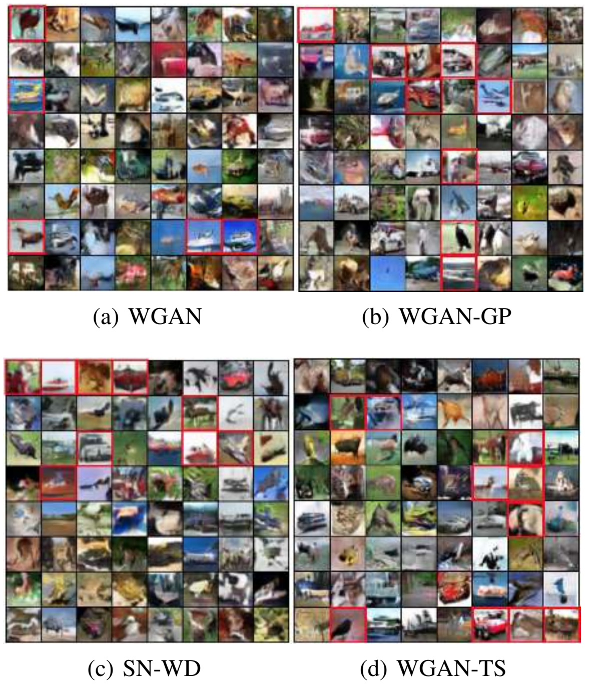


Figure 6. On the CIFAR-10 dataset, images generated by (a) WGAN, (b) WGAN-GP, (c) SN-WD and (d) WGAN-TS. Images that are recognizable are boxed in red. (Higher resolution comparison for the selected images are in the supplemental materials.)

Table 2. Critic time consumption per generator iteration.

METHOD	CRITIC TIME (IN SECONDS)
WGAN	0.373 ± 0.171
WGAN-GP	0.881 ± 0.277
FISHERGAN	0.499 ± 0.307
SN-WD	0.435 ± 0.272
WGAN-TS	$\mathbf{0.278} \pm \mathbf{0.129}$

6. Conclusion

We proposed a new formulation of the Monge-Kantorovich dual formulation to compute the Wasserstein Distance (WD). We showed that the WD can be solved by a combination of linear programming and DNN regression. The approximation error bound and the error bound of generalizing the WD from the discrete to the continuous distributions were also given. Based on the TS method, we proposed the WGAN-TS. Compared to other GANs, the WGAN-TS does not need additional hyper-parameters for weight constraints. Results on the 8 Gaussian dataset showed that the WD computed by the WGAN-TS is more accurate. Results on MNIST, LSUN, and CIFAR-10 datasets show that the proposed method is faster than state-of-the-art WGAN methods, and still produces comparable image quality.

Acknowledgements

We thank Ke Ma and Vu Nguyen for their valuable comments and for contributing some of the figures in this paper. This work was supported by a gift from Adobe, NSF grants CNS-1718014 and DMS 1737876, the Partner University Fund, and the SUNY2020 Infrastructure Transportation Security Center.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *Proceedings of the International Conference on Machine Learning*, 2017.

Bińkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. Demystifying MMD GANs. In *Proceedings of the International Conference on Learning Representations*, 2018.

Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. Training generative neural networks via maximum mean discrepancy optimization. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2015.

Fedus, W., Rosca, M., Lakshminarayanan, B., Dai, A. M., Mohamed, S., and Goodfellow, I. Many paths to equilibrium: GANs do not need to decrease a divergence at every step. In *Proceedings of the International Conference on Learning Representations*, 2018.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *Journal of Machine Learning Research*, 13(3):723–773, 2012.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, 2017.

Guo, X., Hong, J., Lin, T., and Yang, N. Relaxed Wasserstein with applications to GANs. *arXiv preprint arXiv:1705.07164*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Hjelm, R. D., Jacob, A. P., Che, T., Trischler, A., Cho, K., and Bengio, Y. Boundary-seeking generative adversarial networks. In *Proceedings of the International Conference on Learning Representations*, 2018.

Karmarkar, N. A new polynomial-time algorithm for linear programming. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 1984.

Kingma, D. P. and Ba, J. Adam: a method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

Langley, P. Crafting papers on machine learning. In *Proceedings of the International Conference on Machine Learning*, 2000.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. MMD GAN: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, 2017.

Liu, M.-Y. and Tuzel, O. Coupled generative adversarial networks. In *Advances in Neural Information Processing Systems*, 2016.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. Least squares generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *Proceedings of the International Conference on Learning Representations*, 2018.

Mroueh, Y. and Sercu, T. Fisher GAN. In *Advances in Neural Information Processing Systems*, 2017.

Mroueh, Y., Sercu, T., and Goel, V. McGAN: mean and covariance feature matching GAN. In *Proceedings of the International Conference on Machine Learning*, 2017.

Müller, A. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29 (2):429–443, 1997.

Nguyen, V., Vicente, T. F. Y., Zhao, M., Hoai, M., and Samaras, D. Shadow detection with conditional generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of the International Conference on Learning Representations*, 2016.

Roth, K., Lucchi, A., Nowozin, S., and Hofmann, T. Stabilizing training of generative adversarial networks through regularization. In *Advances in Neural Information Processing Systems*, 2017.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, 2016.

Shu, Z., Yumer, E., Hadap, S., Sunkavalli, K., Shechtman, E., and Samaras, D. Neural face editing with intrinsic image disentangling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4 (2):26–31, 2012.

Villani, C. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Yeh, R., Chen, C., Lim, T. Y., Hasegawa-Johnson, M., and Do, M. N. Semantic image inpainting with perceptual and contextual losses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Zhang, Y., Yu, F., Song, S., Xu, P., Seff, A., and Xiao, J. Large-scale scene understanding challenge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015.

Zhu, W. and Xie, X. Adversarial deep structural networks for mammographic mass segmentation. *arXiv preprint arXiv:1612.05970*, 2016.