Parallel Wasserstein Generative Adversarial Nets with Multiple Discriminators

Yuxin Su[1] , **Shenglin Zhao**[2*] , **Xixian Chen**[2] , **Irwin King**[1] and **Michael Lyu**[1]
[1]The Chinese University of Hong Kong, Shatin, Hong Kong
[2]Youtu Lab, Tencent, Shenzhen, China
yxsu@cse.cuhk.edu.hk, zsl.zju@gmail.com, xixianchen@tencent.com, {king, lyu}@cse.cuhk.edu.hk

Abstract

Wasserstein Generative Adversarial Nets (GANs) are newly proposed GAN algorithms and widely used in computer vision, web mining, information retrieval, etc. However, the existing algorithms with approximated Wasserstein loss converge slowly due to heavy computation cost and usually generate unstable results as well. In this paper, we solve the computation cost problem by speeding up the Wasserstein GANs from a well-designed communication efficient parallel architecture. Specifically, we develop a new problem formulation targeting the accurate evaluation of Wasserstein distance and propose an easily parallel optimization algorithm to train the Wasserstein GANs. Compared to traditional parallel architecture, our proposed framework is designed explicitly for the skew parameter updates between the generator network and discriminator network. Rigorous experiments reveal that our proposed framework achieves a significant improvement regarding convergence speed with comparable stability on generating images, compared to the state-of-the-art of Wasserstein GANs algorithms.

1 Introduction

Generative Adversarial Networks (GANs) [Goodfellow *et al.*, 2014] have recently achieved significant progress for numerous applications such as computer vision [Wu *et al.*, 2018], information retrieval [Wang *et al.*, 2017; Su *et al.*, 2017] and recommender system [Zhao *et al.*, 2016; 2018; Zhang *et al.*, 2019]. GANs consist of two networks: a generator network that creates plausible data given some noise as the latent seed variable, and a discriminator network that is trained to distinguish between the generator's fake output and real data sample. The critical issue in the GAN training process is to measure the difference between the real data distribution and the generated fake data distribution. For better measurement between the two distributions, Wasserstein GANs [Arjovsky *et al.*, 2017] are proposed with better training stability over prior Jensen-Shannon (JS) divergence and Kullback-Leibler (KL) divergence GANs [Goodfellow *et al.*, 2014].

Wasserstein GANs have attracted much interest in the community recently. The original Wasserstein GAN and its variants employ Kantorovich-Rubinstein duality, which involves a saddle-point objective with a strong 1-Lipschitz constraint for the discriminator function. On the other hand, some researchers approximate the Wasserstein distance in the primal form rather than solving a dual problem to improve stability. [Deshpande *et al.*, 2018] introduces a random projection to estimate the Wasserstein distance from samples directly. [Salimans *et al.*, 2018] employs the Sinkhorn algorithm to calculate the primal form of Wasserstein distance with entropic regularization. Although the performance of the state-of-the-art Wasserstein GAN is better than the original version, the computation complexity of the approximation to dual or primal Wasserstein measures keeps increasing.

In general, the Wasserstein GAN framework consists of two time-consuming components: the forward and backward updates of the discriminator and the generator, and the approximation of Wasserstein distance. Since the discriminator employs the approximated Wasserstein distance to estimate the distance between the real distribution and the generated distribution, we should guarantee the correctness of the Wasserstein distance per generator update. Otherwise, the discriminator may mislead the generator during the training process. Overall, the computation cost is much heavier when the inaccurate approximation involves the frequent update of the discriminator. In practice, we usually spend several days training a generator model with edging NVIDIA GPUs.

In this paper, we explore the parallel computation to speed up the Wasserstein GAN training. A straightforward solution is based on data-parallelism. A set of worker unit containing model replicas is fed with different partitions of the input dataset, and the model replicas are synchronized via a parameter server. Unfortunately, the above approach is not ideal because of the skew parameter updates in GAN framework—most of GAN algorithms update the discriminator several times for every generator update. The frequent update of discriminator easily becomes a bottleneck due to the limited communication bandwidth in all distributed systems.

In order to eliminate the communication bottleneck for the synchronization of the discriminator, we intend to remove the synchronization requirements by allowing the discrimi-

*Corresponding Author

nators different among worker units. Besides this, we also follow the data-parallelism to split the whole dataset into different worker unit. In this case, we can consider that each worker unit contains different real distribution supported by portions of the whole dataset. Then we formulate the objective function as minimizing the averaged Wasserstein distance between the generated distribution and the multiple real distributions. Inspired by [Genevay *et al.*, 2016; Staib *et al.*, 2017], we optimize the objective function by the stochastic algorithm for discrete optimal transport problem, which can be efficiently paralleled.

In summary, our contributions to parallel Wasserstein GANs are highlighted as follows:

- To our best knowledge, we are the first to propose a parallel architecture for Wasserstein GANs framework to speed up the training of GANs from the computational perspective.
- Different from common Wasserstein GANs, we develop an efficient stochastic algorithm to approximate the Wasserstein distance with higher accuracy for the newly proposed parallel framework.
- In experiments, we show that the proposed parallel architecture enjoys the superior convergence speed and comparable stability.

2 Preliminary and Related Work

The GAN defines two competing networks: The generator network G produces a data x from a source of noise $\mathbf{z} \sim \mathbb{P}_z$. The discriminator network D is trained to distinguish between the generated fake sample and a real sample. Formally, the GAN defines the following minimax objective:

$$\min_G \max_D \mathbb{E}_{x_r \sim \mathbb{P}_r} \left[\log\left(D\left(x_r\right)\right)\right] + \mathbb{E}_{x_g \sim \mathbb{P}_g} \left[\log\left(1 - D\left(x_g\right)\right)\right], \tag{1}$$

where $\mathbb{P}_r$ and $\mathbb{P}_g$ denote the real distribution and the generated distribution respectively. Practically, the expectations are empirically evaluated using samples. Unfortunately, well-learned discriminators may suppress the training of generators. We need to well tune the rounds of discriminator updates after every generator update to ease this training instability.

To enhance the training stability, [Arjovsky *et al.*, 2017] introduces Wasserstein-1 distance to GAN framework. The Wasserstein distance is derived from the optimal transport problem, which is defined as follows:

$$W_c\left(\mathbb{P}_r, \mathbb{P}_g\right) = \min_{\gamma \in \Pi\left(\mathbb{P}_r, \mathbb{P}_g\right)} \int_{\mathcal{X} \times \mathcal{X}} c\left(x_r, x_g\right) \mathrm{d}\gamma(x_r, x_g), \tag{2}$$

where $\Pi\left(\mathbb{P}_r, \mathbb{P}_g\right)$ represents all joint distributions, and $c\left(x_r, x_g\right)$ is the cost to move a unit of mass from x_r to x_g. When $c\left(x_r, y_g\right) = d_{\mathcal{X}}\left(x_r, x_g\right)^p$ in Eq. (2), $W_c^{1/p}$ is called Wasserstein-p distance between probability measures. The Wasserstein distance is made known as a more geometric-aware cost function for learning the distributions supported by the low-dimensional manifold, which is a widely adopted assumption in the feed-forward neural network.

Wasserstein GAN (WGAN) employs Kantorovich-Rubinstein duality of Wasserstein-1 distance, which relies on 1-Lipschitz continuity of the discriminator:

$$W_1\left(\mathbb{P}_r, \mathbb{P}_g\right) = \sup_{\|D_\theta\|_L \leq 1} \mathbb{E}_{x_r \sim \mathbb{P}_r}\left[D_\theta\left(x_r\right)\right] - \mathbb{E}_{x_g \sim \mathbb{P}_g}\left[D_\theta\left(x_g\right)\right], \tag{3}$$

where $\mathbb{P}_g \sim G_\vartheta\left(\mathbb{P}_z\right)$. To approximate the supremum in Eq. (3), [Arjovsky *et al.*, 2017] proposes to clip the parameters θ to enforce the discriminator D_θ to be 1-Lipschitz.

Currently, the Wasserstein GAN approach is considered as the state-of-the-art method due to the theoretical contributions and competitive performance. However, to approximate the 1-Lipschitz constraint is very challenging in the Kantorovich-Rubinstein dual form of the Wasserstein-1 metric. [Gulrajani *et al.*, 2017] introduces a soft penalty for the violation of 1-Lipschitzness (WGAN-GP). The gradient is evaluated as a linear interpolation between the training data and generated samples as a proxy to the optimal coupling. The gradient penalty only takes effect on the observed data $\{\mathbf{x}\}$, the other support domain is not covered. [Wei *et al.*, 2018] follows WGAN-GP to enforces the Lipschitz continuity over all the data manifold and its surrounding regions.

An alternative way to solve the problem in Eq. (2) is to minimize the primal of optimal transport. The primal formulation is numerically stable because it does not involve differentiating the dual solution. [Bousquet *et al.*, 2017] proposes to minimize a regularized primal form of optimal transport problem. [Petzka *et al.*, 2018] explores the theoretical properties of such regularization under Wasserstein GANs framework. [Deshpande *et al.*, 2018] employs random projection to approximate the Wasserstein distance directly. The discriminator is not mandatory in their approach. [Genevay *et al.*, 2018b] proposes a divergence measurement based on the Sinkhorn algorithm, which is originally designed for discrete optimal transport with entropic regularization.

To tackle with parallel computation, [Chavdarova and Fleuret, 2018] proposes an ensemble method of pairs of the generator and discriminator, which can be trained in parallel naturally. However, the communication demand is severe, because the "messenger" discriminators and generators with a massive number of parameters need to be synchronized among workers.

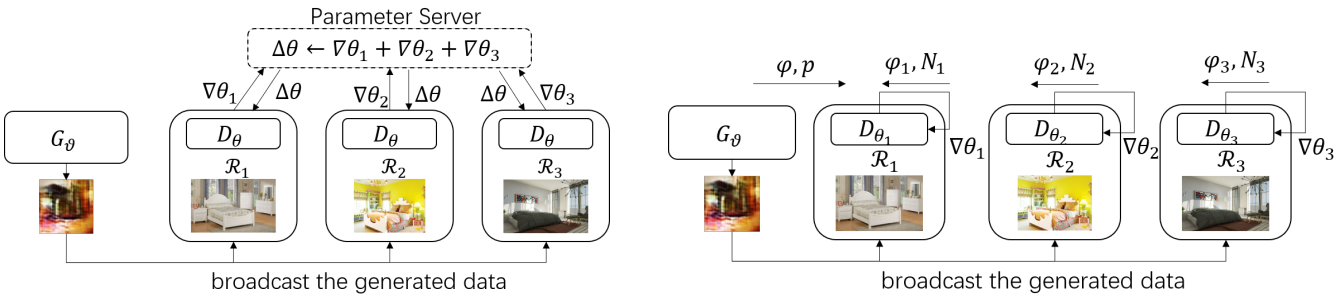
3 Proposed Approach

In all Wasserstein GANs framework, the Wasserstein distance plays a pivotal role to estimate the distance between the real distribution $\mathbb{P}_r$ and the generated distribution $\mathbb{P}_g$. In real-world applications, we estimate the distributions by the empirical probabilistic measure with sampled data. Formally, we formulate the empirical measure of the real distribution μ_r and the artificially generated distribution μ_g by Dirac mass as follows:

$$\mu_r = \sum_{i=1, x \in \mathcal{R}}^{n_r} p_i^r \delta_{x_i^r}, \tag{4}$$

$$\mu_g = \sum_{j=1, x \in \mathcal{G}}^{n_g} p_j^g \delta_{x_j^g}, \tag{5}$$

where $\delta_{x_i^r}$ and $\delta_{x_j^g}$ are Dirac function at location x_i^r and x_j^g, p_i^r and p_j^g are probability masses associated to the i-th real



(a) Parameter Server (b) Proposed Architecture

Figure 1: Comparisons of the popular parameter server and the proposed architecture.

and j-th generated samples respectively, and n_r and n_g are the cardinality of the set of real data and generated fake data in a mini-batch respectively.

When the number of data in each mini-batch increases, the empirical distribution μ_r approximates the real distribution $\mathbb{P}_r$ with higher accuracy [Genevay *et al.*, 2018a]. However, the size of the mini-batch is constrained due to the limited memory in each worker (e.g., GPU). To solve this problem, we turn to a parallel training mechanism in a distributed environment.

3.1 Framework

We propose a communication-efficient framework for parallel Wasserstein GAN. The details are shown in Figure 1.

Figure 1 (a) shows the traditional parallel approach [Dean and et al., 2012; Li *et al.*, 2016]. In this framework, we split the training dataset across several workers first. Then, for each iteration, each worker with a model replica conducts forward-backward computation to calculate the gradients in parallel. Finally, the model parameters are synchronized with a global parameter server before the next iteration starts. This framework is widely used in distributed deep learning systems like Tensorflow [Abadi *et al.*, 2016].

However, the traditional parallel approach in Figure 1 (a) contains two drawbacks. The parameter server is not proper to train the discriminator for GAN framework. Because the frequent synchronization of the parameters in the discriminator and the generator leads to massive communication demand, which degrades the training speed in any parallel solutions. Another drawback of the parameter server comes from the gradient aggregation for a large-scale batch. Traditionally, the gradient used for back-propagation is the arithmetic mean of gradients from different portions of mini-batch. The averaged gradient can not preserve the geometry information, which is vital to depict the semantic information.

To overcome the drawbacks in traditional parallel approaches, we propose a new parallel framework for Wasserstein GAN training, shown in Figure 1 (b). In our approach, we assign K worker unit with a discriminator D_{θ_k} and a master unit with a generator G_ϑ. θ_k and ϑ represent the parameters of the neural networks in the discriminator k and the generator respectively. Note that $\theta_k \neq \theta_{k'}$ in our proposed architecture generally. Our proposed framework relaxes the communication constraints by eliminating the synchronization of the discriminator among workers. In addition, this relaxed requirement could improve the diversity of the discriminator model and intuitively enhance the generator by increasing the difficulty of cheating the discriminator.

In our proposed architecture, the training real dataset $\mathcal{R}$ is split into K workers. Then, for worker k, the empirical measure μ_k^r is supported by the subset of real data $\mathcal{R}_k$.

Similar to the original Wasserstein GAN, we try to minimize the sum of the Wasserstein distance between the generated distribution μ_g and all real distribution μ_k^r with empirical measures as follows:

$$\mathcal{L}\left(\mu_g, \{\mu_k^r\}_{k=1}^K\right) = \frac{1}{K}\sum_{k=1}^{K} W_c\left(\mu_g, \mu_k^r\right). \tag{6}$$

3.2 Parallel Wasserstein GAN Algorithm

We develop a novel parallel Wasserstein GAN algorithm to solve the objective function in the our proposed architecture, namely Eq. (6). The loss function defined in Eq. (6) is similar to the Wasserstein barycenter problem introduced in [Agueh and Carlier, 2011]. Intuitively, the optimal generated fake distribution should be approximated to the barycenter of the real distributions in all workers. Specifically, we employ the empirical distribution of the barycenter of all real distributions as the supervised information to train the generator network. In order to solve the problem in Eq. (6) with more accuracy, we consider to calculate the Wasserstein distance directly, which is different from the recently popular solutions with entropic regularization [Genevay *et al.*, 2016; Cuturi, 2013].

In the proposed framework, we follow [Genevay *et al.*, 2016; Claici *et al.*, 2018] to consider the semi-discrete Kantorovich's dual formulations of the primal optimal transport problem in Eq. (2):

$$W_c\left(\mu_g, \mu_r\right) = \max_{\varphi \in \mathbb{R}^{n_g}} \left\{ \sum_{j=1}^{n_g} \varphi_j p_j^g + \int_\Omega \varphi^c(x_r) \mathrm{d}\mu_r(x_r) \right\}, \tag{7}$$

where $\varphi^c\left(x_r\right) = c\left(x_r, x_j^g\right) - \varphi_j$ is the c-transform of the Kantorovich potential φ_j [Villani, 2008]. Therefore, $\mathcal{L}$ is reformulated as:

Algorithm 1: Generator as Master Unit

Input: K: number of worker unit, n_g: batch size in generator, n_r: batch size in worker, T: number of iteration for the estimation of the probability Dirac mass.
Output: G_ϑ: the generator model.
while ϑ *has not converged* **do**
  Sample Gaussian noise $z_1, \ldots, z_{n_g}$ from $\mathcal{N}(0,1)$
  Generate fake images $\left\{x_j^g; x_j^g = G_\vartheta\left(z_j\right)\right\}_{j=1}^{n_g}$
  $\varphi_j \leftarrow 0\ \forall j \in [1, n_r]$
  Broadcast $\{x_j^g\}_{j=1}^{n_g}$ to workers
  for *t=1,…,T* **do**
    $\varphi_j \leftarrow \frac{1}{K}\sum_{k=1}^{K}\varphi_j^k\ \forall j \in [1, n_g]$
    Broadcast $\{\varphi_j, p_j^g\}_{j=1}^{n_g}$ to workers
    Receive $\{\varphi_j^k, N_j^k\}_{j=1}^{n_g}$ from workers
    $p_j^g \leftarrow \frac{1}{K}\sum_{k=1}^{K} N_j^k\ \forall j \in [1, n_g]$
  end
  Conduct back-propagation with the loss: $\text{Loss}(\{D_\theta^g(G_\vartheta(z_j))\}_{j=1}^{n_g}, \{p_j^g\})$
end

$$\mathcal{L}\left(\mu_g, \{\mu_k^r\}_{k=1}^K\right) = \max_{\varphi \in \mathbb{R}^{n_g}} \left\{ F(\varphi) = \sum_{j=1}^{n_g} \varphi_j p_j^g + \frac{1}{K}\sum_{k=1}^{K}\int_{\Omega_k} \varphi^c(x_r)\mathrm{d}\mu_k^r(x_r) \right\} \tag{8}$$

From Eq. (8), we can know that $F(\varphi)$ is a concave function. Then we can derive W_c from the optimization using the stochastic gradient ascent method. Since $F(\varphi)$ is derivative with respective to each individual φ_j:

$$\frac{\partial F}{\partial \varphi_j} = p_j^g - \frac{1}{K}\sum_{k=1}^{K}\int_{\mathrm{Vor}_{\varphi_j}^k} \mathrm{d}\mu_k^r\left(x_r\right), \tag{9}$$

where Vor_{φ_j} is Voronoi cells of the point x_j^g [Santambrogio, 2015] defined in the following:

$$\mathrm{Vor}_{\varphi_j} = \left\{ x \in \mathcal{R} \ :\ c\left(x, x_j^g\right) - \varphi_j \le c\left(x, x_{j'}^g\right) - \varphi_{j'} \forall j' \right\}. \tag{10}$$

Therefore, W_c is obtained when $\frac{\partial F}{\partial \varphi_j} = 0$ and the optimal p_j^* is calculated as:

$$p_j^* = \frac{1}{K}\sum_{k=1}^{K}\int_{\mathrm{Vor}_{\varphi_j}^k} \mathrm{d}\mu_k^r\left(x_r\right) \tag{11}$$

$$= \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_{x^r \sim \mu_k^r}\left[I_{x^r \in \mathrm{Vor}_{\varphi_j}^k}\right] \tag{12}$$

$$\approx \frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{n_k^r} I_{x_i^k \in \mathrm{Vor}_{\varphi_j}^k}, \tag{13}$$

Algorithm 2: Discriminator as Worker Unit

Input: K: number of worker unit, n_r: batch size in worker unit, η : learning rate for Kantorovich potential, T: number of iterations for the Dirac mass estimation.
for all *worker* $k \in [1, K]$ **do in parallel**
  Sample $\{x_i\}_{i=1}^{n_r}$ a batch from the real dataset $\mathcal{R}_k$
  Receive fake images $\left\{x_j^g\right\}_{j=1}^{n_g}$ from the master unit
  Evaluate cost $c_{ij} = \left\|D_{\theta_k}\left(x_i^r\right) - D_{\theta_k}\left(x_j^g\right)\right\|\ \forall i \in [1, n_r], j \in [1, n_g]$
  for $t = 1, \ldots, T$ **do**
    Receive $\{\varphi_j, p_j^g\}_{j=1}^{n_g}$ from master
    for $j = 1, \ldots, n_g$ **do**
      Compute Vor_{φ_j} from Eq. (10)
      $N_j^k \leftarrow \sum_{i=1}^{n_k^r} I_{x_i^k \in \mathrm{Vor}_{\varphi_j}^k}$
      $\varphi_j^k \leftarrow \varphi_j - \eta N_j^k$
    end
    Send $\{\varphi_j^k, N_j^k\}_{j=1}^{n_g}$ to master
  end
  Conduct back-propagation with the loss: $\text{Loss}(\{D_\theta^k(x_j^g)\}_{j=1}^{n_g}, \{N_j^k\})$
end

where I is denoted as the indicator function of a given set. p_j^* is the valuable supervised information of our GAN framework for the generated fake point x_j^g since p_j^g represents the weight of the point x_j^g in the empirical distribution.

In our framework, the computation of p_j^* in Eq. (11) is easily decomposed to all workers in parallel, because the Voronoi cells $\mathrm{Vor}_{\varphi_j}^k$ in each worker k are independent with each other. When the empirical estimation in each worker is finished, p_j^* is reduced to the master worker with x_j^g. Algorithm 1 and 2 detail the proposed parallel algorithm with mini-batch.

In many machine learning algorithms armed with Wasserstein distance, the ground cost function on the given metric is relatively simple, such as the ℓ_2 norm in Euclidean space. However, ℓ_2 norm is not proper for high-dimensional image data in GAN framework, because it lacks complex geometric information. Therefore, we parameterize the cost function $c\left(x_r, x_g\right)$ from the discriminator as follows:

$$c_\theta\left(x_r, x_g\right) = \left\|D_\theta\left(x_r\right) - D_\theta\left(x_g\right)\right\|, \tag{14}$$

where D_θ is the discriminator network with parameters θ. When p^* is obtained, we update the discriminator network D_θ^k in all workers and the generator network G_ϑ by back-propagation.

It is important to note that our approach does not contain any cumbersome communication policy that may increase the negative effect of mismatching communication bandwidth in the distribution environment. For each worker unit, they only receive generated data and send a simple vector of size n_g to optimize the proposed objective function collaboratively. Moreover, our approach is compatible with different configurations for parallel computation: (a) All GPUs are installed

via PCI Express in a single workstation. In this setting, each GPU is considered as an isolated computation worker in the parallel concept. (b) There are multiple machines, each of which may contain more than one GPU, deemed as distributed computation. In the following, we take experiments in case (b) to evaluate our approach.

4 Experiments

In the experimental part, we evaluate the proposed parallel approach on two widely used image datasets CIFAR-10 and LSUN [Yu *et al.*, 2015]. CIFAR-10 contains 60,000 small color images with size 32×32. LSUN is a large-scale scene understanding dataset. Following the setting in many Wasserstein GANs papers, we mainly use the bedroom category from LSUN dataset. The bedroom category contains 3 million color images with a size of 128×128.

All experiments in this section are conducted in a cluster with four machines with 2 NVIDIA GTX 1080 GPUs each. Therefore, we could scale our approach up to 8 GPUs setting, which contains one master unit for the generator and seven workers for the discriminators.

We implement our algorithms with PyTorch [Paszke *et al.*, 2017], and utilize the 101-layer ResNet block in generator and discriminator. We assign the number of iteration for per mini-batch update $T = 5$ for all experiments. Intuitively, the probability of Dirac mass for each generator update should be more accurate when T increases. However, larger T means that the communication rounds between the master and workers will increase as well.

4.1 Convergence Speed for Scalability

In a parallel environment, we assume that all workers have the same batch size n_r. Therefore, in the proposed parallel approach, the total number of real images sampled from the dataset for a single generator update is $K \cdot n_r$, where K is the number of worker unit. Usually, the range of n_r is constrained to the memory in GPUs. Then, the batch size of real images scales linearly with the number of machines. For the special "2 GPUs" setting in experiments, we utilize two GPUs equipped at different machines to make a fair comparison, because the inner-machine communication is much faster than the inter-machine protocol.

To demonstrate the advantage of our proposed method compared to the traditional distributed deep learning framework, we implement parallel WGAN-GP through MPI interface in PyTorch. MPI follows All-Reduce policy, equivalent to parameter server framework depicted in Figure 1 (a).

Figure 2 (a) and (b) illustrate the convergence speed with regard to the number of the GPUs, where the y-axis shows the change of estimated Wasserstein distance between the real distribution and the generated distribution during the training. From the figures, we can see both of the proposed method and WGAN-GP with more GPUs always converge faster than the same approach with fewer GPUs.

We also employ the standard speedup measurement in distributed computation to demonstrate the scalability of our proposed method, and the measurement is defined as below:

Method	LSUN bed. (FID)	CIFAR-10 (IS)
WGAN-GP (8 GPUs)	27.3	7.73
Ours (2 GPUs)	23.2	7.12
Ours (4 GPUs)	21.9	7.68
Ours (8 GPUs)	**21.0**	**7.81**

Table 1: Quantitative evaluations on CIFAR-10 and LSUN dataset. Smaller FID is better, larger IS is better.

$$\text{Speedup}(N) = \frac{\text{The execution time of one unit}}{\text{The execution time of } N \text{ units}}. \quad (15)$$

In the experiments, we measure the duration from starts to the first checkpoint where the corresponding estimated Wasserstein distance is less than 5.0. We consider 5.0 as an ending point because all methods converge with less vibration when the estimated distance is less than 5.0. From Figure 2 (c), we can easily conclude that our method enjoys a better convergence speed than the traditional distributed deep learning framework, which is coincident with our expectation.

Due to limited space, we do not demonstrate the figures for CIFAR10 dataset. Actually, the behavior of two methods in CIFAR10 is quite similar to Figure 2–both methods converge faster if more GPUs, and our approach is faster than the traditional parallel framework.

4.2 Quantitative and Qualitative Evaluation

We first conduct the quantitative evaluation to demonstrate the stability of the proposed method. We calculate the Inception Score during the training and evaluate the generator performance on CIFAR-10. A higher Inception Score indicates a better ability of the generative model to produce samples with variability. Practically, we calculate the maximal Inception Score reached in 20,000 generator updates.

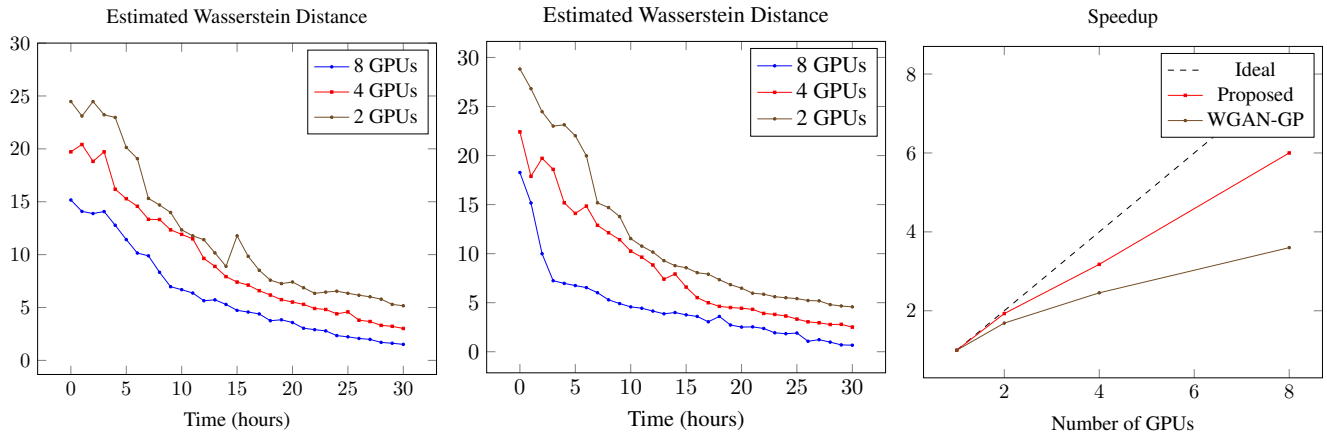
We also follow [Wu *et al.*, 2018] to evaluate the model performance on LSUN dataset by Fréchet Inception Distance (FID) [Heusel *et al.*, 2017], which measures the difference between real and fake data distributions.

From Table 1, we observe that the performance of the proposed method becomes better when the number of worker unit increases, which mainly benefits from larger size of real data batch among multiple worker units. Moreover, our method enjoys much better performance over WGAN-GP.

We also take qualitative evaluation from the visualization performance. Figure 4 indicates that the generated images of our proposals with different number of GPUs are comparable to results in WGAN-GP. Moreover, comparable results in Figure 4 (b-d) show that our parallel method has no restriction on the number of GPUs used to generate plausible results. Figure 3 shows the performance of our proposed method is also comparable with WGAN-GP on CIFAR-10 data. Hence our method enjoys superior performance concerning the convergence speed with the comparable generation quality.

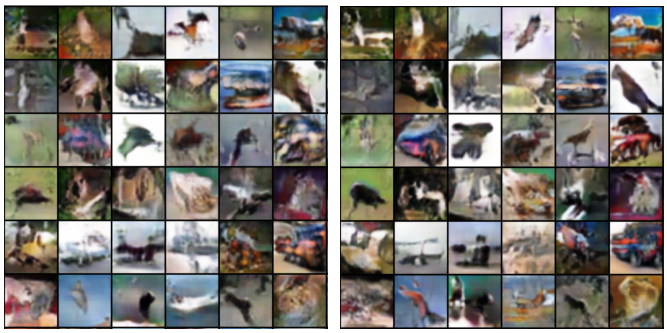
5 Conclusions

In this paper, we introduce a novel parallel architecture designed for GANs framework to speed up the costly compu-



(a) WGAN-GP with default distributed framework (b) Proposed (c) Speedup

Figure 2: The estimated Wasserstein distance of WGAN-GP and the proposed method both trained on LSUN Bedroom dataset. To make fair comparisons, we employ PyTorch's default MPI interface to train WGAN-GP in parallel. In (c), we calculate the speedup based on the execution time from the start to the first checkpoint where the estimated Wasserstein distance is less than 5.0.



(a) WGAN-GP (8 GPUs) (b) Proposed (8 GPUs)

Figure 3: Qualitative comparisons between the state-of-the-art WGAN-GP and the proposed method.

tation in Wasserstein GANs with multiple computation units. Experimental reports demonstrate that our proposed architecture enjoys an attractive scalability performance to decrease the training time remarkably. Moreover, both quantitative scores and qualitative demonstration display the proposed loss function with the Wasserstein metric and the parallel algorithm are reasonable and suitable for the parallel environment.

Acknowledgements

The work described in this paper was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14208815 and No. CUHK 14210717 of the General Research Fund).

(a) WGAN-GP (8 GPUs) (b) Proposed (2 GPUs)

(c) Proposed (4 GPUs) (d) Proposed (8 GPUs)

Figure 4: Qualitative comparisons between WGAN-GP and the proposed method with different configurations. Images in (a) are generated by WGAN-GP; images in (b, c, d) are generated by the proposed model trained with the different number of GPUs. All models are trained on LSUN bedroom dataset within 24 hours.

References

[Abadi *et al.*, 2016] Martín Abadi, Paul Barham, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, pages 265–283, 2016.

[Agueh and Carlier, 2011] Martial Agueh and Guillaume Carlier. Barycenters in the wasserstein space. *SIAM J. Math. Analysis*, 43(2):904–924, 2011.

[Arjovsky *et al.*, 2017] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, pages 214–223, 2017.

[Bousquet *et al.*, 2017] Olivier Bousquet, Sylvain Gelly, Ilya Tolstikhin, Carl Johann Simon-Gabriel, and Bernhard Schölkopf. From optimal transport to generative modeling: the vegan cookbook. Technical report, 2017.

[Chavdarova and Fleuret, 2018] Tatjana Chavdarova and François Fleuret. Sgan: An alternative training of generative adversarial networks. In *CVPR*, 2018.

[Claici *et al.*, 2018] Sebastian Claici, Edward Chien, and Justin Solomon. Stochastic wasserstein barycenters. In *ICML*, pages 998–1007, 2018.

[Cuturi, 2013] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, pages 2292–2300, 2013.

[Dean and et al., 2012] Jeffrey Dean and Greg Corrado et al. Large scale distributed deep networks. In *NeurIPS*, pages 1232–1240, 2012.

[Deshpande *et al.*, 2018] Ishan Deshpande, Ziyu Zhang, and Alexander G. Schwing. Generative modeling using the sliced wasserstein distance. In *CVPR*, June 2018.

[Genevay *et al.*, 2016] Aude Genevay, Marco Cuturi, Gabriel Peyré, and Francis R. Bach. Stochastic optimization for large-scale optimal transport. In *NeurIPS*, pages 3432–3440, 2016.

[Genevay *et al.*, 2018a] Aude Genevay, Lénaic Chizat, Francis Bach, Marco Cuturi, and Gabriel Peyré. Sample complexity of sinkhorn divergences. *arXiv preprint arXiv:1810.02733*, 2018.

[Genevay *et al.*, 2018b] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *AISTATS*, pages 1608–1617, 2018.

[Goodfellow *et al.*, 2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014.

[Gulrajani *et al.*, 2017] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *NeurIPS*, pages 5769–5779, 2017.

[Heusel *et al.*, 2017] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, pages 6629–6640, 2017.

[Li *et al.*, 2016] Jinfeng Li, James Cheng, Yunjian Zhao, Fan Yang, Yuzhen Huang, Haipeng Chen, and Ruihao Zhao. A comparison of general-purpose distributed systems for data processing. In *Big Data*, pages 378–383, 2016.

[Paszke *et al.*, 2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[Petzka *et al.*, 2018] Henning Petzka, Asja Fischer, and Denis Lukovnicov. On the regularization of wasserstein gans. *ICLR*, 2018.

[Salimans *et al.*, 2018] Tim Salimans, Han Zhang, Alec Radford, and Dimitris N. Metaxas. Improving gans using optimal transport. *ICLR*, 2018.

[Santambrogio, 2015] Filippo Santambrogio. Optimal transport for applied mathematicians. *Birkäuser, NY*, 55:58–63, 2015.

[Staib *et al.*, 2017] Matthew Staib, Sebastian Claici, Justin M. Solomon, and Stefanie Jegelka. Parallel streaming wasserstein barycenters. In *NeurIPS*, pages 2644–2655, 2017.

[Su *et al.*, 2017] Yuxin Su, Irwin King, and Michael R. Lyu. Learning to rank using localized geometric mean metrics. In *SIGIR*, pages 45–54, 2017.

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

[Wang *et al.*, 2017] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. IRGAN: A minimax game for unifying generative and discriminative information retrieval models. In *SIGIR*, pages 515–524, 2017.

[Wei *et al.*, 2018] Xiang Wei, Boqing Gong, Zixia Liu, Wei Lu, and Liqiang Wang. Improving the improved training of wasserstein gans: A consistency term and its dual effect. *ICLR*, 2018.

[Wu *et al.*, 2018] Jiqing Wu, Zhiwu Huang, Janine Thoma, Dinesh Acharya, and Luc Van Gool. Wasserstein divergence for gans. *ECCV*, 2018.

[Yu *et al.*, 2015] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015.

[Zhang *et al.*, 2019] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems. *arXiv preprint arXiv:1905.13129*, 2019.

[Zhao *et al.*, 2016] Shenglin Zhao, Tong Zhao, Haiqin Yang, Michael R Lyu, and Irwin King. Stellar: spatial-temporal latent ranking for successive point-of-interest recommendation. In *AAAI*, 2016.

[Zhao *et al.*, 2018] Shenglin Zhao, Xixian Chen, Irwin King, and Michael R Lyu. Personalized sequential check-in prediction: Beyond geographical and temporal contexts. In *ICME*, pages 1–6, 2018.