Run-Sort-ReRun: Escaping Batch Size Limitations in Sliced Wasserstein Generative Models

José Lezama [1] **Wei Chen** [2] **Qiang Qiu** [2]

Abstract

When training an implicit generative model, ideally one would like the generator to reproduce all the different modes and subtleties of the target distribution. Naturally, when comparing two empirical distributions, the larger the sample population, the more these statistical nuances can be captured. However, existing objective functions are computationally constrained in the amount of samples they can consider by the memory required to process a batch of samples. In this paper, we build upon recent progress in sliced Wasserstein distances, a family of differentiable metrics for distribution discrepancy based on the Optimal Transport paradigm. We introduce a procedure to train these distances with virtually any batch size, allowing the discrepancy measure to capture richer statistics and better approximating the distance between the underlying continuous distributions. As an example, we demonstrate the matching of the distribution of Inception features with batches of tens of thousands of samples, achieving FID scores that outperform state-of-the-art implicit generative models.

1. Introduction

A fundamental resource for training implicit deep generative models is the ability to characterize the discrepancy between the distribution of generated samples and that of real training data. Naturally, as one would like the generative model to capture the subtleties of the target distribution, the larger the population this characterization takes into account, the more accurate it will be. In recent years, a lot of progress has been made in developing and improving differentiable measures of discrepancy between empirical distributions (Goodfellow et al., 2014; Arjovsky et al., 2017; Li et al., 2017; Mroueh et al., 2017; Heusel et al., 2017; Deshpande et al., 2018). However, the number of samples that one can consider when optimizing a generative model is typically constrained by the memory required to store the model, activations and gradients in the computing device (i.e. the GPU). This limits the richness and complexity of the statistics that the objective function can extract from the available samples. Figure 1 illustrates this point with an extreme example: learning a data distribution containing four modes, but using only four samples at each training iteration. Clearly this is not enough to capture the complexity of the target distribution for existing models.

In this paper, we present a method to overcome the limitations in the number of samples that can be considered when computing the mismatch between the generated and real distributions. The algorithm, termed Run-Sort-ReRun, allows to efficiently compute a measure of distribution mismatch, effectively considering tens of thousands of samples in each training iteration. The algorithm is thus able to capture the complexities of the target distribution, regardless of the hardware constraints imposed on the batch size, as exemplified in Figure 1.

The method is based on the recent success of sliced Wasserstein distances (Rabin et al., 2011; Bonneel et al., 2015; Deshpande et al., 2018; Kolouri et al., 2018; 2019a) for training generative models. The main idea is to use an average of 1-D Wasserstein distances obtained from random projections of the original distributions. These methods have the advantage of efficiently computing a stochastic measure that in expectation approximates the actual Wasserstein distance between the full distributions (Bonnotte, 2013). Moreover, this type of training does not necessarily require adversarial training, which is often unstable. On top of these advantages, we add the ability to use arbitrarily large batch sizes, making the empirical distance even closer to the actual distance between the distributions.

In our experiments, we demonstrate the effectiveness of the method by directly matching the distribution of real and generated image features, considering empirical populations of thousands of samples. By directly optimizing over Inception features (Szegedy et al., 2016), our models obtain FID scores that surpass the most advanced GAN

[1]IIE, Universidad de la República, Montevideo, Uruguay [2]ECE, Purdue University, West Lafayette, USA. Correspondence to: <jlezama@fing.edu.uy>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

models in the literature.

2. Background

The task of generative modeling is to characterize the probability distribution of a given dataset $D = \{(\boldsymbol{x})\}$ with m samples $\boldsymbol{x} \sim P_r$, where P_r is an unknown real data distribution. Implicit generative models learn a parametric transformation from a tractable base distribution to a learned distribution $P_\theta, \boldsymbol{\theta} \in \mathbb{R}^d$. The goal is to make the distribution of generated samples P_θ as similar as possible to P_r.

Generative Adversarial Networks (GANs) treat such a task as a distance minimization problem. Normally, the optimal distance can be achieved by minimizing the Jensen-Shannon divergence between P_r and P_θ (Goodfellow et al., 2014), but doing so often leads to vanishing gradients as the discriminator guiding P_θ saturates. Arjovsky & Bottou (2017) provided important theoretical analysis of this problem, stating that the inverted Kullback-Leibler divergence is prone to gradient vanishing and mode dropping, and proposed in Arjovsky et al. (2017) to replace the Jensen-Shannon divergence with the 1-Wasserstein distance. Specifically, the p-Wasserstein distance between the unknown real data distribution P_r and the parametric distribution P_θ, defined on a compact data space X, is given by

$$W_p(P_r, P_\theta) = \inf_{\gamma \in \Pi(P_r, P_\theta)} \left(\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\gamma}[\|\boldsymbol{x} - \boldsymbol{y}\|^p]\right)^{\frac{1}{p}}, \quad (1)$$

where $\Pi(P_r, P_\theta)$ denotes all joint distributions of $\boldsymbol{x}$ and $\boldsymbol{y}$ that have respective marginals P_r and P_θ. W_p is a metric on the space of probability measures on X (Villani, 2008) [Chapter 6]. Since the actual data distribution P_r is unknown, it is hard to compute the infimum in Equation 1. Therefore, as proposed in (Arjovsky et al., 2017), by employing the Kantorovich-Rubinstein duality the 1-Wasserstein distance yields:

$$W(P_r, P_\theta) = \sup_{\|f\|_L \leq 1} \left(\mathbb{E}_{\boldsymbol{x}\in P_r}[f(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\in P_\theta}[f(\boldsymbol{x})]\right), \quad (2)$$

where the supremum is over all 1-Lipschitz functions f: $X \to \mathbb{R}$. The correctness of the Wasserstein GAN estimation depends on how well the discriminator or critic f has been trained to achieve this supremum, a difficult task due to the instability and complexity of this minimax optimization, while guaranteeing the Lipschitz condition.

Another convenient approximation consists in using the "sliced" version of the Wasserstein distance (Deshpande et al., 2018; Deshpande et al., 2019; Kolouri et al., 2019b; Wu et al., 2019; Deshpande et al., 2019; Nguyen et al., 2020). In general, the "sliced p-Wasserstein distance" between distributions P_r and P_θ is defined as:

$$\tilde{W}(P_r, P_\theta) = \left[\int_{\boldsymbol{\omega}\in\Omega} W_p^p(P_r^{\boldsymbol{\omega}}, P_\theta^{\boldsymbol{\omega}}) d\boldsymbol{\omega}\right]^{\frac{1}{p}}, \quad (3)$$

where $P_r^{\boldsymbol{\omega}}$, $P_\theta^{\boldsymbol{\omega}}$ denote the projection (*i.e.*, marginal) of P_r and P_θ onto the direction $\boldsymbol{\omega}$, and Ω is the set of all possible directions on the unit sphere.

In practice, these methods approximate the sliced 2-Wasserstein distance between the distributions by using samples $D \sim P_r$, $F \sim P_\theta$, and replacing the integration over Ω with a Monte Carlo approximation over a randomly chosen set of unit vectors $\hat{\Omega}$. With P_θ (and hence, F) being implicitly parametrized by $\boldsymbol{\theta}$, the following program is used for generative modeling:

$$\min_{\boldsymbol{\theta}} \frac{1}{|\hat{\Omega}|} \sum_{\boldsymbol{\omega}\in\hat{\Omega}} W_2^2(D^{\boldsymbol{\omega}}, F^{\boldsymbol{\omega}}). \quad (4)$$

The 2-Wasserstein distance between the projected samples $D^{\boldsymbol{\omega}}$ and $F^{\boldsymbol{\omega}}$ can be computed by finding the optimal transport map. For 1-D distributions, this can be done through sorting (Villani, 2008):

$$W_2^2(D^{\boldsymbol{\omega}}, F^{\boldsymbol{\omega}}) = \frac{1}{|D|} \sum_i \|F^{\boldsymbol{\omega}}_{\pi_F(i)} - D^{\boldsymbol{\omega}}_{\pi_D(i)}\|_2^2, \quad (5)$$

where $\pi_D(i)$ and $\pi_F(i)$ are permutations that sort the projected sample sets $D^{\boldsymbol{\omega}}$ and $F^{\boldsymbol{\omega}}$ respectively, *i.e.*, $D^{\boldsymbol{\omega}}_{\pi_D(1)} \leq D^{\boldsymbol{\omega}}_{\pi_D(2)} \leq \dots D^{\boldsymbol{\omega}}_{\pi_D(|D|)}$.

It can be shown that the empirical distributions D and F arbitrarily approximate their unknown continuous counterparts as the number of samples increases under Wasserstein distance (Weed et al., 2019), and therefore that the empirical p-Wasserstein distance converges to the continuous distance, although with a slow sample complexity rate of $\mathcal{O}\left(n^{-1/(2p)}\right)$ (Sommerfeld et al., 2019). This motivates the main goal of this work: the search for a practical algorithm that can effectively utilize arbitrarily large populations of samples.

3. Method

3.1. Motivating Example

We consider a motivating example in Figure 1, where we want to learn a distribution with four modes (leftmost subfigure), but having an allowance of four samples per training batch. This extreme example illustrates the usual case where a restrictive training batch size results in practice in an objective function that only compares two distributions using relatively few samples, not enough to capture the complexity of the target distribution.

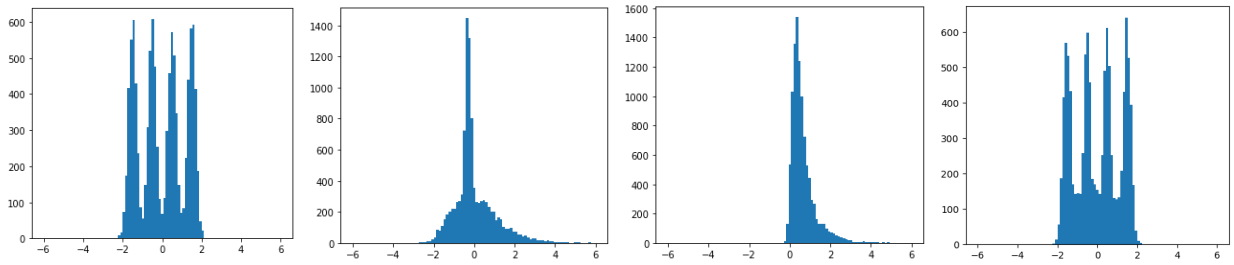


Figure 1. Comparison of empirical distributions obtained by different methods. In this example we show the shortcomings of training generative models with a batch size that is too small compared to the complexity of the target distribution, a common scenario under memory limitations. We train simple 1-D generative models to replicate a mixture of 4 Gaussians (leftmost), using batches of 4 samples. The standard GAN suffers from "mode collapse". Optimizing the empirical 1-D Wasserstein distance also fails to capture the 4 modes with such few samples. Our method, Run-Sort-ReRun is able to process one small batch (4 samples here) at a time, yet virtually support arbitrarily large batch sizes (in this case 1024), overcoming the memory limitations and successfully capturing all the modes of the target distribution.

The second sub-figure in Figure 1 shows the result of training a standard GAN with an adversarial discriminator on the ground truth distribution. One can immediately observe the well known "mode collapse" problem, where the generator is only able to replicate one of the modes of the distribution. The third sub-figure shows the result of trying to optimize the 1-D empirical 2-Wasserstein distance with only four random samples, which is clearly not enough to capture this distribution. A typical solution for emulating large batch sizes in stochastic gradient descent is to accumulate the gradient of multiple small training batches, but the same mode collapse behavior is observed even when accumulating the gradient of 1024 samples. Finally, the rightmost sub-figure shows the result of the proposed Run-Sort-ReRun method, processing a batch of four samples a time for an effective large batch size of 1024. By effectively optimizing the distance of empirical populations of 1024 samples, the proposed algorithm is the only one that is able to reproduce all the modes of the ground truth distribution. Moreover, although we used a batch size of four for illustration, exactly the same behavior would be obtained by this algorithm with a smaller batch size, to the extreme of using a single sample per batch. We present next the proposed Run-Sort-ReRun algorithm.

3.2. Run-Sort-ReRun Algorithm

In this section we describe the Run-Sort-ReRun algorithm, which can effectively overcome batch size limitations imposed by memory constraints. These memory constraints are typically due to the capacity of the GPU to store all the activations and gradients of the model for backpropagation.

To illustrate the proposed algorithm, we focus on the problem of computing the 1-D empirical Wasserstein distance, which is the building block of the sliced Wasserstein distance in (4). The 2-Wasserstein distance between two 1-D empirical distributions of the same size can be computed as in Equation (5). We begin by noticing that the 2-Wasserstein distance in (5) is a particular type of regression problem, where the regression target for each generated sample is given by the corresponding sample from the real distribution. Namely, the i^{th} sample from the generated distribution F is mapped according to

$$i \to \pi_D^{-1} \circ \pi_F(i). \tag{6}$$

The central idea of the Run-Sort-ReRun algorithm is to compute the regression targets in (6) as they would be if the number of samples was very large, such that the empirical distance arbitrarily approximates the actual distance between the continuous distributions. Let $G(\boldsymbol{z}; \boldsymbol{\theta})$ be a generative model with parameters $\boldsymbol{\theta}$ whose input is a latent vector $\boldsymbol{z} \in \mathbb{R}^h$ and output is a generated synthetic sample. The proposed algorithm iterates over the following *Run*, *Sort* and *ReRun* steps:

- In the ***Run*** step, one small batch of size m at a time, both real and synthetic empirical distributions $D^{\boldsymbol{\omega}}$ and $F^{\boldsymbol{\omega}}$ are computed and stored ($|D^{\boldsymbol{\omega}}| \gg m$). For the synthetic data, the latent vectors $\boldsymbol{z}$ are also stored so they can be used to regenerate the synthetic samples in the *ReRun* step.
- In the ***Sort*** step, both stored distributions are sorted obtaining $\pi_D(i)$ and $\pi_F(i)$, so the corresponding regression targets in (6) are known.
- In the ***ReRun*** step, small batches of samples are regenerated from the stored latent vectors, and compared to the targets obtained in the previous step. The

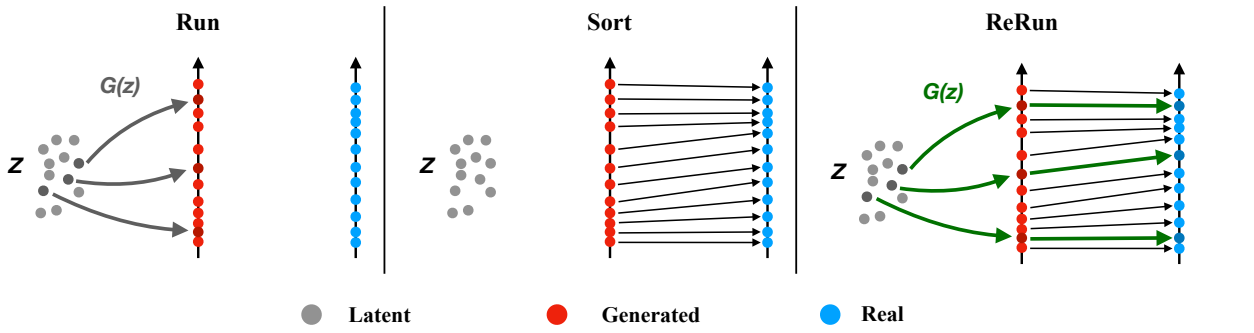


Figure 2. Schematic representation of the Run-Sort-ReRun algorithm. In this example, we assume that the generator G can only process 3 samples at a time, and that the desired full sample size is 12. We consider a single projection for illustration. In the **Run** step, the latent vectors $\boldsymbol{z}$ (gray) are used to generate synthetic samples (red), 3 at a time (dark gray/dark red). A group of 12 real samples (blue) is also collected. In the **Sort** step, the 12 synthetic and the 12 real samples are sorted, thus establishing the correspondences (straight arrows) for computing (5). In the **ReRun** step, synthetic samples are generated again, 3 at a time, from the same latent vectors as in the Run step. The distances to their correspondences (dark blue) are computed, so the gradient of the 1-D Wasserstein on the 12 samples can be computed, even though no more than 3 samples were ran through the generator at any time.

1-D Wasserstein loss (5), corresponding to a large number of samples, can now be backpropagated using only small batches of samples.

Note that the only step requiring automatic differentiation is the *ReRun* step. Thus, the algorithm only needs to store intermediate network activations and gradients for a small number of samples, yet the true sliced Wasserstein distance for an arbitrary large number of samples is computed and optimized. Detailed pseudocode of the algorithm, including the projection step is presented in Algorithm 1 and illustrated in Figure 2. Source code is publicly available[1].

3.3. On the Choice of Projection Directions

A critical aspect of the sliced Wasserstein distance is the choice of directions of $\hat{\Omega}$ onto which the empirical samples are projected to compute the Monte Carlo approximation in (4). Various strategies have been proposed in the literature for choosing the sampling distribution of random directions. A simple choice is to use random Gaussian directions (Wu et al., 2019; Kolouri et al., 2019b), however a lot of computation is wasted on directions that are irrelevant or poorly informative for distinguishing both distributions. A deterministic approach in (Deshpande et al., 2019) aims at using the direction of maximum separation, motivated by the fact that random directions are with high probability orthogonal to it. More recently, (Nguyen et al., 2020) demonstrated even improved results by optimizing over a parametric family of distributions of random directions, with a variability constraint. When no variability is enforced, this amounts to the maximum direction of separation as in (Deshpande et al., 2019).

[1] https://github.com/jlezama/RSR

While the Run-Sort-ReRun algorithm is agnostic to the choice of projection directions, and can be used together with any of these strategies, here we introduce a different strategy, that requires no extra training as in (Deshpande et al., 2019; Nguyen et al., 2020), and is simple to compute. It is motivated by the following observation: We note that the contribution to the sliced Wasserstein distance (5) of a generated sample $\boldsymbol{y}_i$ under projection $\boldsymbol{\omega}$ is given by

$$c_i = ((\boldsymbol{y}_i - \boldsymbol{x}_{[i]})^\top \boldsymbol{\omega})^2, \tag{7}$$

where $\boldsymbol{x}_{[i]}$ is the corresponding real sample after the sorting. Thus, if we wanted to steer $\boldsymbol{\omega}$ to a direction of greater separation between the samples, the gradient $\nabla_{\boldsymbol{\omega}} c_i$ would point towards $(\boldsymbol{y}_i - \boldsymbol{x}_{[i]})$. On the other hand, the line joining two random datapoints is typically much closer to the data manifold than random Gaussian directions. Consider for example the case of two separated Gaussian distributions, whose covariance matrices are close to singular, lying in a very high-dimensional space.

Based on this observation, we construct the distribution of projection directions by sampling random pairs of generated and real samples, giving the alternative formulation:

$$\hat{W}_p^p(P_r, P_\theta) = \mathbb{E}_{\boldsymbol{x}\sim P_r, \boldsymbol{y}\sim P_\theta}\left[W_p^p(P_r^{\omega(\boldsymbol{x},\boldsymbol{y})}, P_\theta^{\omega(\boldsymbol{x},\boldsymbol{y})})\right], \tag{8}$$

where $\omega(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})/||\boldsymbol{x} - \boldsymbol{y}||$.

Finally, to maintain a balance between exploiting informative directions and exploring new directions, in each new iteration we re-utilize a subset of the random directions that produced the largest separation according to the 1-D Wasserstein distance. The procedure is described in Algorithm 2.

Algorithm 1 Run-Sort-ReRun

Input: Real data $X \in \mathbb{R}^{d\times n}$, large batch size b, small batch size m, generator G with parameters $\boldsymbol{\theta}$, number of projections p, step size α, latent dimension d_z

repeat

(auto-diff OFF)

Initialize $Z \in \mathbb{R}^{d_z\times b}, D \in \mathbb{R}^{d\times b}, F \in \mathbb{R}^{d\times b}$: storage for latent vectors, real data and model output resp.

Initialize $\Omega \in \mathbb{R}^{d\times p}$: projection matrix

RUN

for $i = 1$ **to** b/m **do**

Z_i = m samples from $\mathcal{N}(0, I)$

$Z[:, (i-1)\times m : i\times m] \leftarrow Z_i$ *// column indexing*

$F[:, (i-1)\times m : i\times m] \leftarrow G(Z_i; \boldsymbol{\theta})$

$D[:, (i-1)\times m : i\times m] \leftarrow m$ samples from X

end for

$D = \Omega D$, $F = \Omega F$ *// project*

SORT

D = sort(D) *// row-wise*

Π_F^{-1} = argsort(argsort(F)) *// row-wise*

$D = D[:, \Pi_F^{-1}]$ *// aligns rows of F and D as per (6)*

(auto-diff ON)

RERUN

Initialize $\boldsymbol{g} = \mathbf{0}$

for $i = 1$ **to** b/m **do**

$Z_i = Z[:, (i-1)\times m : i\times m]$

$F_i = G(Z_i; \boldsymbol{\theta})$

$D_i = D[:, (i-1)\times m : i\times m]$

$c = ||\Omega F_i - D_i||_F^2/m$

$\boldsymbol{g} = \boldsymbol{g} + \nabla_{\boldsymbol{\theta}} c$

end for

$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \boldsymbol{g}$

until convergence

Algorithm 2 Choice of Projection Directions

Input: Number of projections p, previous projection directions $\Omega_t \in \mathbb{R}^{p\times d}$, number of re-utilized directions r, real data $X \in \mathbb{R}^{d\times n}$, synthetic data $F \in \mathbb{R}^{d\times n}$

Compute $\boldsymbol{d}_{\Omega_t} \in \mathbb{R}^p$, vector of sliced distances for each row of Ω_t using *Run-Sort-ReRun*

Initialize $\Omega_{t+1} \in \mathbb{R}^{d\times p}$

$\boldsymbol{z}_t$ = argsort($\boldsymbol{d}_{\Omega_t}$) *// descending*

$\Omega_{t+1}[0{:}r, :] = \Omega_t[\boldsymbol{z}_t[0{:}r], :]$ *// keep rows with largest distances*

for $i = 1$ **to** $p - r$ **do**

Sample $\boldsymbol{x}_i$ from X, $\boldsymbol{y}_i$ from F

$\Omega_{t+1}[r+i, :] = \{(\boldsymbol{x}_i - \boldsymbol{y}_i)/\|\boldsymbol{x}_i - \boldsymbol{y}_i\|\}$

end for

return Ω_{t+1}

3.4. Algorithmic Complexity Analysis

The time complexity of the Run-Sort-ReRun algorithm, similarly to the standard sliced Wasserstein distance, is dominated by the sorting of each projection of the samples. Considering the cost of processing b samples under p projections, Run-Sort-ReRun requires $\mathcal{O}(b \log b \cdot p)$ operations, due to the full batch sorting. In comparison, the standard sliced Wasserstein distance, requires $\mathcal{O}(b \log m \cdot p)$ operations to process batches of m samples. This results in an a increment of $\log(b) - \log(m)$, which is only logarithmic in the large batch size b. Besides this theoretical increment, Run-Sort-ReRun also requires an extra forward pass for the *Run* step.

The space complexity of Run-Sort-ReRun is dominated by the storage of b latent vectors, b, generated and b real samples for sorting. Compared to the standard sliced Wasserstein distance, the increment is linear in the large batch size b. In practice, this is generally much smaller than the memory required by model intermediate activations, weights and gradients, especially for very large modern generative networks, so it is possible to store these in the GPU. Alternatively, they can be stored in the main system memory, and the sorting performed by the CPU.

In summary, Run-Sort-ReRun offers a trade-off between extra training time and memory, and the ability to capture more nuances of the target distribution by effectively considering larger batch sizes. Moreover, it still remains practically superior to computing to the full Wasserstein distance, which requires solving the optimal assignment problem which can be $\mathcal{O}(b^3 \log b)$ (Pele & Werman, 2009).

4. Experiments

4.1. Matching Distributions of Inception Features

A common metric for evaluating the performance of a generative model is the Fréchet Inception Distance or FID score (Heusel et al., 2017). This metric consists in fitting Gaussian distributions to the real and generated image features as extracted by an Inception network (Szegedy et al., 2016), and then computing the 2-Wasserstein distance (in closed form) between the two Gaussians. Typically, tens of thousands of samples from both the real data and the generated data distribution are used to fit the Gaussians. Training a generator to directly optimize this metric remained impractical due to the large number of samples required and the memory needed to use both the generator and the Inception networks.

In this section, we experimentally validate the ability of the Run-Sort-ReRun algorithm to overcome limitations in

batch size by demonstrating the direct optimization of the sliced Wasserstein distance between Inception features of thousands of samples of real and generated images. In the *Run* step, we randomly sample thousands of real images from the dataset and generate the same amount of synthetic images with the generator. Then, we project the Inception features to 1-D using the projection directions as described in Section 3.3. We *Sort* the real images and synthetic images according to their projected Inception features. In the *ReRun* step, we backpropagate and accumulate gradients for the sliced Wasserstein distance loss between the sorted embeddings of synthetic images and their corresponding embeddings of real images. Note that the memory required to store both the generator and Inception models is not a problem for Run-Sort-ReRun since it is designed to process arbitrarily small batches at a time. Importantly, we directly minimize the discrepancy between the two full distributions using arbitrarily population sizes, without any assumptions of Gaussianity.

In our experiments, we used the Run-Sort-ReRun algorithm to fine-tune two state-of-the-art GAN models. The first model is E2GAN (Tian et al., 2020), a Generator architecture obtained by neural architecture search. We use this model on CIFAR-10 (Krizhevsky et al., 2009), with 50,000 training images of size 32×32 and STL-10 (Coates et al., 2011), with 105,000 training images of size 96×96. We use large batch size $b = 16000$ and $p = 4000$ projection directions and $r = 1333$. To compute the FID scores, we use the statistics provided by the authors of (Heusel et al., 2017). The second model is StyleGAN-v2 (Karras et al., 2020b). For this model we conducted experiments on the LSUN Church datasets (Yu et al., 2015) for images of size 256x256, using large batch size $b = 8192$ and $p = 4000$, $r = 1333$. For all models we use the Adam optimizer with learning rate 10^{-4}.

We feed the output of the generator model to an Inception V3 (Szegedy et al., 2016) network and extract feature embeddings of dimension of 2048, which are then projected and used to compute the average 1-D 2-Wasserstein distance with Run-Sort-ReRun. Thus, we directly optimize the generator to match the distribution of inception features. Note that linear projections in the space of Inception features correspond to non-linear projections in image space. This can be seen as a particular case of the Generalized sliced Wasserstein distance (Kolouri et al., 2019a). Table 1 shows FID scores on test set obtained by this procedure[2]. Notably, the FID decreases significantly with respect to the baseline, and the obtained scores outperform the most advanced GAN models. Figure 3 shows a PCA plot of the inception embeddings for the CIFAR-10 dataset.

[2]FID score on CIFAR-10 train set (50K samples) is 3.13 for ours, 11.0 for baseline.

CIFAR-10

Method	FID ↓
E2GAN (Tian et al., 2020)	11.26
DiffAugment-CR-BigGAN (Zhao et al., 2020)	8.49
Mix-BigGAN (Tang, 2020)	8.17
StyleGAN-v2 + ADA (Karras et al., 2020a)	7.1
E2GAN + RSR (Ours)	**4.9**

STL-10

Method	FID ↓
Auto GAN (Gong et al., 2019)	31.01
DEGAS (Doveh & Giryes, 2019)	28.76
E2GAN (Tian et al., 2020)	25.35
E2GAN + RSR (Ours)	**8.6**

LSUN Church

Method	FID ↓
StyleGAN-v2 (Karras et al., 2020b)	3.86
StyleGAN-v2 + converted weights	5.93
StyleGAN-v2 + converted weights + RSR (Ours)	**3.14**

Table 1. FID scores on CIFAR-10, STL-10 and LSUN Church. For CIFAR-10 and STL-10, we finetune a baseline method (Tian et al., 2020) using the Run-Sort-ReRun (RSR) algorithm with large batch sizes of 16,000 samples. For LSUN Church, we finetune a StyleGAN-v2 model (Karras et al., 2020b) with 8192 samples per batch.

With Run-Sort-ReRun the generated distribution is refined to more accurately match the ground truth. Although the Gaussian assumption of the FID score is clearly violated, the fitted Gaussians also get closer when the full distributions are matched, decreasing the FID score.

4.2. Ablation Study on Effective Batch Size

The central motivation for Run-Sort-ReRun is that computing the empirical distance between distributions benefits from observing larger batches of samples (Weed et al., 2019; Deshpande et al., 2019). We validate this idea by performing an ablation study on the large batch size parameter b. The results are shown in Table 2. For different values of b, we adapt the number of iterations such that in all experiments the total number of samples seen during training is the same (in this case $\sim 5M$). We observe a notorious decrease in FID score at $b = 1024$, a batch size that would be completely impractical under traditional methods, justifying the utility of the proposed method.

4.3. On Generated Image Quality

A natural question is whether the direct optimization of the distribution of Inception features leads to higher quality samples. We observed that in general there is not a no-

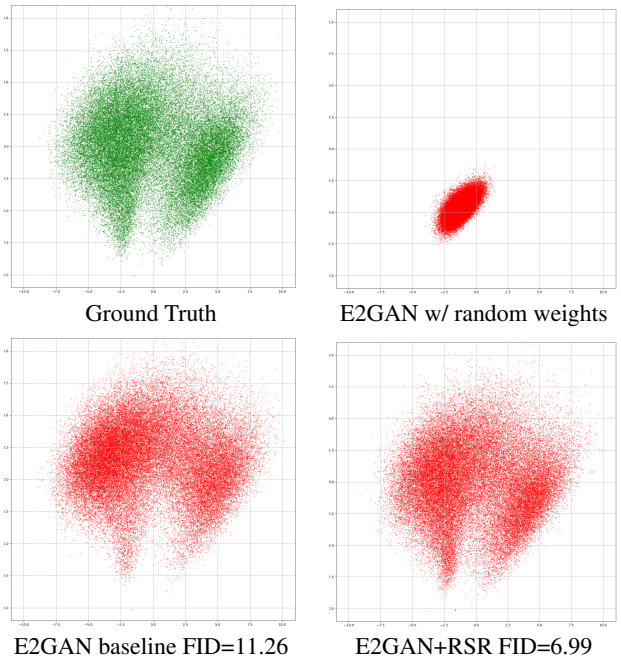


Figure 3. PCA projections of Inception features for CIFAR-10 dataset. Run-Sort-ReRun refines the distribution of samples generated by the baseline E2GAN (Tian et al., 2020) to more accurately match the ground truth distribution.

torious increase in image quality (see Figure 4). However, improvements in generated image quality can be obtained with Run-Sort-ReRun by using four rather than just one layer of the Inception network. In Figure 5 we show the result of applying Run-Sort-ReRun to the concatenation of the output of the first four blocks of the Inception-V3 network. To reduce the dimensionality of the first layers, we applied spatial average pooling to finally obtain features of dimensions 2304, 1728, 3072 and 2048 which we concatenate into a vector of dimension 9152. We use large batch size $b = 16000$, $p = 4000$ and $r = 1333$. A comparison between images generated by the baseline (Tian et al., 2020) trained on STL-10 and the baseline finetuned with Run-Sort-ReRun is shown in Figure 5. To better distinguish the improvement in synthetic image quality, we separated the generated images by class using an external classifier trained on STL-10. Investigating other image features that can result in better image quality when the distribution is matched to the extent that Run-Sort-ReRun allows remains an interesting avenue for future work.

CIFAR-10

Large batch size b	FID ↓
Baseline	11.26
RSR, $b = 256$	10.02
RSR, $b = 1024$	7.39
RSR, $b = 2048$	7.23
RSR, $b = 8192$	6.99

Table 2. Ablation study on the large batch size parameter b. We finetune the baseline method (Tian et al., 2020) using Run-Sort-ReRun (RSR) for different values of b. The number of iterations is adjusted so that all models see the same total number of samples.

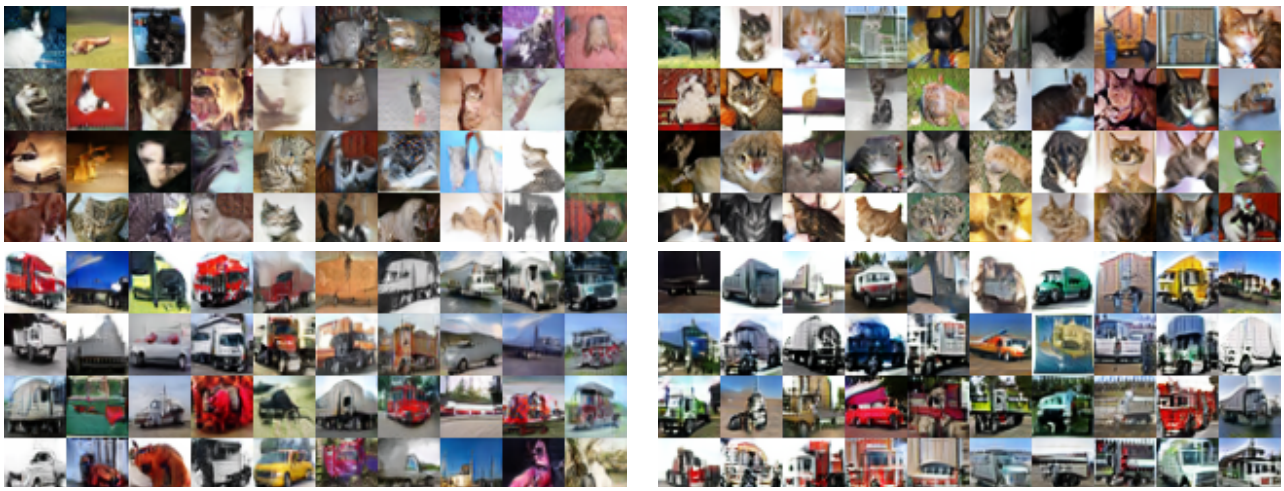
5. Related Work

Generative Adversarial Networks Generative adversarial networks (GANs) formulate generative modeling as a two-player game. A generator learns to generate plausible data, and a discriminator learns to distinguish the synthetic data from real data (Goodfellow et al., 2014). Both generator and discriminator are parameterized using deep neural networks and trained via stochastic gradient descent. There are various applications of GANs in different areas, such as computer vision (Zhu et al., 2017; Radford et al., 2015; Denton et al., 2015; Dumoulin et al., 2016; Ledig et al., 2017; Reed et al., 2016; Isola et al., 2017) and language translation (Conneau et al., 2018; Wu et al., 2018). Some popular models include DCGAN (Radford et al., 2015), PGGAN (Karras et al., 2017), StyleGAN (Karras et al., 2019; 2020b), BigGAN (Brock et al., 2018), to name a few. In the original formulation (Goodfellow et al., 2014), a GAN minimizes the Jensen-Shannon divergence between the true data distribution and the generated probability distribution induced in the data space by the generator. However, this divergence can lead to gradient vanishing or mode collapse. To mitigate the problems, many other variants have been proposed, which use some divergence like KL-divergence (Sønderby et al., 2016), Least Squares (Mao et al., 2017) and Wasserstein distance (Arjovsky et al., 2017), to measure the distance between the distributions.

Wasserstein Distance Wasserstein distance (Rubner et al., 1998; Peyré et al., 2019) considers the minimal transformation between two distributions of points, according to some measure of transportation cost. Due to its ability to overcome vanishing gradients, some works train GANs based on the Wasserstein distance (Arjovsky et al., 2017; Arjovsky & Bottou, 2017). Although the Wasserstein distance-based GANs have been successful in several complex generative tasks, they suffer from the "curse of dimensionality" (Weed et al., 2019). To tackle the instability and complexity, sliced versions of the 2-Wasserstein distance were proposed, which only requires estimating distances of 1-D distributions (Gulrajani et al., 2017; Kolouri et al., 2019b; Deshpande et al., 2018; Deshpande et al., 2019; Wu et al., 2019; Nguyen et al., 2020). However, in those methods, the batch size one can consider is typically under unavoidable real-world constraints, e.g., GPU memory limi-

Baseline: StyleGAN2 (Karras et al., 2020b), (FID 5.93) StyleGAN2 finetuned with RSR, (FID 3.14)

Figure 4. Results of StyleGAN2 on LSUN Churches[4]. We observe that directly optimizing the distribution of Inception features improves the FID score over the state-of-the-art methods.



Baseline (Tian et al., 2020) (FID 25.35) Baseline + RSR (FID 14.39)

Figure 5. Qualitative comparison between baseline E2GAN (Tian et al., 2020) and the same baseline finetuned with Run-Sort-ReRun on the features produced by four different layers of the Inception network, using an effective batch size of 16,000. An external pre-trained classifier was used on the generated samples to separate samples of different classes (top: cat, bottom: truck). Besides decreasing the FID score, the samples of the model trained with the proposed method exhibit better defined shapes. Best appreciated in electronic format.

tation. This limits the possibility of representing complex target distributions and may lead to the classical "mode collapse" problem (see Figure 1), which is addressed in this paper. The Wasserstein distance is also often used in measuring the performance of a generator with Fréchet inception distances (FID) (Heusel et al., 2017).

6. Conclusion

Since the surge in popularity of implicit deep generative models began in 2014, a lot of progress has been achieved both in training strategies and loss functions. Central to this progress has been the quest for useful trainable metrics for comparing two empirical distributions. In this paper, we focus on a key aspect of this comparison: the number of samples considered when computing a cost for the discrepancy between distributions. The motivation is that the larger the number of samples, the more the statistical nuances and complexities of the target distribution can be captured. However, existing methods are constrained in the number of samples they can process in one learning iteration, due to hardware memory limitations. With this in mind, we proposed the Run-Sort-ReRun algorithm. Building on the successful sliced Wasserstein distance, Run-Sort-ReRun is able to compute a cost function for the mismatch between real and generated distributions considering an arbitrarily large number of samples, yet in practice processing only one small batch at a time. Experimentally, we demonstrated the validity of the approach by matching the distributions of deep image features using batches of thousands of samples. When the deep features are extracted with an Inception network, we can directly optimize the popular FID score, obtaining simple models that outperform the most advanced GAN models under this metric.

Acknowledgements

JL was supported by SNI, ANII and CSIC grant I+D 2018-256. Experiments were partially carried out using ClusterUY, the National Center for Supercomputing, Uruguay. WC and QQ were supported by the DARPA TAMI program.

References

Arjovsky, M. and Bottou, L. Towards principled methods for training generative adversarial networks. *International Conference on Learning Representations*, 2017.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International conference on machine learning*, 2017.

Bonneel, N., Rabin, J., Peyré, G., and Pfister, H. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015.

Bonnotte, N. *Unidimensional and evolution methods for optimal transportation*. PhD thesis, Paris 11, 2013.

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *International Conference on Learning Representations*, 2018.

Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *international conference on artificial intelligence and statistics*, 2011.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H. Word translation without parallel data. In *International Conference on Learning Representations*, 2018.

Denton, E., Chintala, S., Szlam, A., and Fergus, R. Deep generative image models using a laplacian pyramid of adversarial networks. *Advances in Neural Information Processing Systems*, 2015.

Deshpande, I., Zhang, Z., and Schwing, A. G. Generative modeling using the sliced wasserstein distance. In *Conference on Computer Vision and Pattern Recognition*, 2018.

Deshpande, I., Hu, Y., Sun, R., Pyrros, A., Siddiqui, N., Koyejo, S., Zhao, Z., Forsyth, D., and Schwing, A. G. Max-sliced wasserstein distance and its use for gans. In *Conference on Computer Vision and Pattern Recognition*, 2019.

Deshpande, I., Hu, Y.-T., Sun, R., Pyrros, A., Siddiqui, N., Koyejo, S., Zhao, Z., Forsyth, D., and Schwing, A. G. Max-sliced wasserstein distance and its use for gans. In *Conference on Computer Vision and Pattern Recognition*, 2019.

Doveh, S. and Giryes, R. Degas: Differentiable efficient generator search. *arXiv preprint arXiv:1912.00606*, 2019.

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. Adversarially learned inference. *International Conference on Learning Representations*, 2016.

Gong, X., Chang, S., Jiang, Y., and Wang, Z. Autogan: Neural architecture search for generative adversarial networks. In *International Conference on Computer Vision*, 2019.

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.

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.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *International Conference on Neural Information Processing Systems*, 2017.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Conference on Computer Vision and Pattern Recognition*, 2017.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Conference on Computer Vision and Pattern Recognition*, 2019.

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., and Aila, T. Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676*, 2020a.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and improving the image quality of stylegan. In *Conference on Computer Vision and Pattern Recognition*, 2020b.

Kolouri, S., Rohde, G. K., and Hoffmann, H. Sliced wasserstein distance for learning gaussian mixture models. In *Conference on Computer Vision and Pattern Recognition*, 2018.

Kolouri, S., Nadjahi, K., Simsekli, U., Badeau, R., and Rohde, G. K. Generalized sliced wasserstein distances. *arXiv preprint arXiv:1902.00434*, 2019a.

Kolouri, S., Pope, P. E., Martin, C. E., and Rohde, G. K. Sliced wasserstein auto-encoders. In *International Conference on Learning Representations*, 2019b.

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

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Conference on Computer Vision and Pattern Recognition*, 2017.

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. Mmd gan: Towards deeper understanding of moment matching network. *arXiv preprint arXiv:1705.08584*, 2017.

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

Mroueh, Y., Sercu, T., and Goel, V. Mcgan: Mean and covariance feature matching gan. In *International conference on machine learning*, pp. 2527–2535. PMLR, 2017.

Nguyen, K., Ho, N., Pham, T., and Bui, H. Distributional sliced-wasserstein and applications to generative modeling. *arXiv preprint arXiv:2002.07367*, 2020.

Pele, O. and Werman, M. Fast and robust earth mover's distances. In *2009 IEEE 12th international conference on computer vision*, pp. 460–467. IEEE, 2009.

Peyré, G., Cuturi, M., et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Rabin, J., Peyré, G., Delon, J., and Bernot, M. Wasserstein barycenter and its application to texture mixing. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pp. 435–446. Springer, 2011.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., and Lee, H. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, 2016.

Rubner, Y., Tomasi, C., and Guibas, L. J. A metric for distributions with applications to image databases. In *International Conference on Computer Vision*. IEEE, 1998.

Sommerfeld, M., Schrieber, J., Zemel, Y., and Munk, A. Optimal transport: Fast probabilistic approximation with exact solvers. *Journal of Machine Learning Research*, 20 (105):1–23, 2019.

Sønderby, C. K., Caballero, J., Theis, L., Shi, W., and Huszár, F. Amortised map inference for image super-resolution. *International Conference on Learning Representations*, 2016.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Conference on Computer Vision and Pattern Recognition*, 2016.

Tang, S. Lessons learned from the training of gans on artificial datasets. *IEEE Access*, 8:165044–165055, 2020.

Tian, Y., Wang, Q., Huang, Z., Li, W., Dai, D., Yang, M., Wang, J., and Fink, O. Off-policy reinforcement learning for efficient and effective gan architecture search. In *European Conference on Computer Vision*, 2020.

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

Weed, J., Bach, F., et al. Sharp asymptotic and finite-sample rates of convergence of empirical measures in wasserstein distance. *Bernoulli*, 25(4A):2620–2648, 2019.

Wu, J., Huang, Z., Acharya, D., Li, W., Thoma, J., Paudel, D. P., and Gool, L. V. Sliced wasserstein generative models. In *Conference on Computer Vision and Pattern Recognition*, 2019.

Wu, L., Xia, Y., Tian, F., Zhao, L., Qin, T., Lai, J., and Liu, T.-Y. Adversarial neural machine translation. In *Asian Conference on Machine Learning*, 2018.

Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., and Xiao, J. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Zhao, S., Liu, Z., Lin, J., Zhu, J.-Y., and Han, S. Differentiable augmentation for data-efficient gan training. *arXiv preprint arXiv:2006.10738*, 2020.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision*, 2017.