Amortized Projection Optimization for Sliced Wasserstein Generative Models

Khai Nguyen[†] Nhat Ho[†]

University of Texas, Austin[†]

March 25, 2022

Abstract

Seeking informative projecting directions has been an important task in utilizing sliced Wasserstein distance in applications. However, finding these directions usually requires an iterative optimization procedure over the space of projecting directions, which is computationally expensive. Moreover, the computational issue is even more severe in deep learning applications, where computing the distance between two mini-batch probability measures is repeated several times. This nested-loop has been one of the main challenges that prevent the usage of sliced Wasserstein distances based on good projections in practice. To address this challenge, we propose to utilize the *learning-to-optimize* technique or *amortized optimization* to predict the informative direction of any given two mini-batch probability measures. To the best of our knowledge, this is the first work that bridges amortized optimization and sliced Wasserstein generative models. In particular, we derive linear amortized models, generalized linear amortized models, and non-linear amortized models which are corresponding to three types of novel mini-batch losses, named *amortized sliced Wasserstein*. We demonstrate the favorable performance of the proposed sliced losses in deep generative modeling on standard benchmark datasets[1].

1 Introduction

Generative modeling is one of the most important tasks in machine learning and data science. Leveraging the expressiveness of neural networks in parameterizing the model distribution, deep generative models such as GANs [17], VAEs [22], and diffusion models [19, 51], achieve a significant quality of sampling images. Despite differences in the way of modeling the model distribution, optimization objectives of training generative models can be written as minimizing a discrepancy $\mathcal{D}(\cdot,\cdot)$ between data distribution μ and the model distribution ν_ϕ with $\phi \in \Phi$, parameter space of neural networks weights, namely, we solve for $\hat{\phi} \in \arg\min_{\phi\in\Phi} \mathcal{D}(\mu, \nu_\phi)$. For example, Kullback–Leibler divergence is used in VAEs and diffusion models, Jensen–Shannon divergence appears in GANs, and f-divergences are utilized in f-GANs [40]. Because of the complexity of the neural networks ϕ, closed-form optimal solutions of these optimization problems are intractable. Therefore, gradient-based methods and their stochastic versions are widely used in practice to approximate these solutions.

Recently, optimal transport-based losses, which we denote as $\mathcal{D}(\cdot,\cdot)$, are utilized to train generative models due to their training stability, efficiency, and geometrically meaning. Examples of these models include Wasserstein GAN [3] with the dual form of Wasserstein-1 distance [43], and OT-GANs [14, 48] with the primal form of Wasserstein distance and with Sinkhorn divergence [8] between mini-batch probability measures. Although these models considerably improve the generative performance, there have been remained certain problems. In particular, Wasserstein GAN is reported to fail

[1] Code for the paper will be published at https://github.com/UT-Austin-Data-Science-Group/AmortizedSW.

to approximate the Wasserstein distance [52] while OT-GAN suffers from high computational complexity of Wasserstein distance: $\mathcal{O}(m^3 \log m)$ and its curse of dimensionality: the sample complexity of $\mathcal{O}(m^{-1/d})$ where m is the number of supports of two mini-batch measures. The entropic regularization [8] had been proposed to improve the computational complexity of approximating optimal transport to $\mathcal{O}(m^2)$ [1, 29, 30, 28] and to remove the curse of dimensionality [33]. However practitioners usually choose to use the slicing (projecting version) of Wasserstein distance [54, 11, 24] due to a fast computational complexity $\mathcal{O}(m \log m)$ and no curse of dimensionality $\mathcal{O}(m^{-1/2})$. The distance is known as sliced Wasserstein distance (SW) [4]. Sliced Wasserstein is defined as the expected one-dimensional Wasserstein distance between two projected measures over the uniform distribution over the unit sphere. Due to the intractability of the expectation, Monte Carlo samples from the uniform distribution over the unit sphere are used to approximate the distance. The number of samples is often called the number of projections and it is denoted as L.

From applications, practitioners observe that sliced Wasserstein distance requires a sufficiently large number of projections L relative to the dimension of data to perform well [24, 11]. Increasing L leads to a linear increase in computational time and memory. However, when data lie in a low dimensional manifold, several projections are redundant since they collapse projected measures to a Dirac-Delta measure at zero. There are some attempts to overcome that issue including sampling orthogonal directions [46] and mapping the data to a lower-dimensional space [11]. The most popular approach is to search for the direction that maximizes the projected distance, which is known as max-sliced Wasserstein distance (Max-SW) [10]. Nevertheless, in the context of deep generative models and deep learning in general, the optimization over the unit sphere requires iterative projected gradient descent methods that can be computationally expensive. In detail, each gradient-update of the model parameters (neural networks) requires an additional loop for optimization of Max-SW between two mini-batch probability measures. Therefore, we have two nested optimization loops: the global loop (optimizing model parameters) and the local loop (optimizing projection). These optimization loops can slow down the training considerably.

Contribution. To overcome the issue, we propose to leverage *learning to learn* techniques (*amortized optimization*) to predict the optimal solution of the local projection optimization. We bridge the literature of amortized optimization and optimal transport by designing amortized models to solve the iterative optimization procedure of finding optimal slices in the sliced Wasserstein generative model. To the best of our knowledge, this is the first time amortized optimization is used in sliced Wasserstein literature. In summary, our main contributions are two-fold:

1. First, we introduce a novel family of mini-batch sliced Wasserstein losses that utilize amortized models to yield informative projecting directions, named amortized sliced Wasserstein ($\mathcal{A}$-SW). We specify three types of amortized models: linear amortized, generalized linear amortized, and non-linear amortized models that are corresponding to three mini-batch losses: linear amortized sliced Wasserstein ($\mathcal{LA}$-SW), generalized linear amortized sliced Wasserstein ($\mathcal{GA}$-SW), and non-linear amortized sliced Wasserstein ($\mathcal{NA}$-SW). Moreover, we discuss some properties of $\mathcal{A}$-SW losses including metricity, complexities, and connection to mini-batch Max-SW.

2. We then introduce the application of $\mathcal{A}$-SW in generative modeling. Furthermore, we carry out extensive experiments on standard benchmark datasets including CIFAR10, CelebA, STL10, and CelebAHQ to demonstrate the favorable performance of $\mathcal{A}$-SW in learning generative models. Finally, we measure the computational speed and memory of $\mathcal{A}$-SW, mini-batch Max-SW, and mini-batch SW to show the efficiency of $\mathcal{A}$-SW.

Organization. The remainder of the paper is organized as follows. We first provide background about Wasserstein distance, sliced Wasserstein distance, max-sliced Wasserstein distance, and amortized optimization in Section 2. In Section 3, we propose amortized sliced Wasserstein distances, analyze some of its theoretical properties. The discussion on related works is given in Section 4. Section 5 contains the application of $\mathcal{A}$-SW to generative models, qualitative and quantitative experimental results on standard benchmarks. In Section 6, we make some conclusion. Finally, we defer the proofs of key results and extra materials in the Appendices.

Notation. For any $d \geq 2$, $\mathbb{S}^{d-1} := \{\theta \in \mathbb{R}^d \mid ||\theta||_2^2 = 1\}$ denotes the d dimensional unit hyper-sphere in $\mathcal{L}_2$ norm, and $\mathcal{U}(\mathbb{S}^{d-1})$ is the uniform measure over $\mathbb{S}^{d-1}$. Moreover, δ denotes the Dirac delta function. For $p \geq 1$, $\mathcal{P}_p(\mathbb{R}^d)$ is the set of all probability measures on $\mathbb{R}^d$ that has finite p-moments. For $\mu, \nu \in \mathcal{P}_p(\mathbb{R}^d)$, $\Pi(\mu,\nu) := \{\pi \in \mathcal{P}_p(\mathbb{R}^d \times \mathbb{R}^d) \mid \int_{\mathbb{R}^d} \pi(x,y)dx = \nu, \int_{\mathbb{R}^d} \pi(x,y)dy = \mu\}$ is the set of transportation plans between μ and ν. For $m \geq 1$, we denotes $\mu^{\otimes m}$ as the product measure which has the support is m random variables follows μ. For a vector $X \in (\mathbb{R}^d)^{\otimes m}$, $X := (x_1, \ldots, x_m)$, P_X denotes the empirical measures $\frac{1}{m}\sum_{i=1}^m \delta_{x_i}$. We denote $\theta \sharp \mu$ as the push-forward probability measure of μ through the function $T_\theta : \mathbb{R}^d \to \mathbb{R}$ where $T_\theta(x) = \theta^\top x$.

2 Background

In this section, we first review the definitions of the Wasserstein distance, the sliced Wasserstein distance, and the max-sliced Wasserstein distance. We then formulate generative models based on the max-sliced Wasserstein distances and review the amortized optimization problem and its application to the max-sliced Wasserstein generative models.

2.1 (Sliced)-Wasserstein Distances

We first define the Wasserstein-p distance [53, 42] between two probability measures $\mu \in \mathcal{P}_p(\mathbb{R}^d)$ and $\nu \in \mathcal{P}_p(\mathbb{R}^d)$.

$$\text{W}_p(\mu,\nu) := \Big(\inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathbb{R}^d \times \mathbb{R}^d} \|x - y\|_p^p d\pi(x,y) \Big)^{\frac{1}{p}}, \tag{1}$$

When $d = 1$, the Wasserstein distance has a closed form which is $W_p(\mu,\nu) = (\int_0^1 |F_\mu^{-1}(z) - F_\nu^{-1}(z)|^p dz)^{1/p}$ where F_μ and F_ν are the cumulative distribution function (CDF) of μ and ν respectively. To utilize this closed-form property of Wasserstein distance in one dimension and overcome the curse of dimensionality of Wasserstein distance in high dimension, the sliced Wasserstein distance [4] between μ and ν had been introduced and admitted the following formulation:

$$\text{SW}_p(\mu,\nu) := \left(\int_{\mathbb{S}^{d-1}} \text{W}_p^p(\theta\sharp\mu, \theta\sharp\nu) d\theta \right)^{\frac{1}{p}}. \tag{2}$$

For each $\theta \in \mathbb{S}^{d-1}$, $\text{W}_p^p(\theta\sharp\mu, \theta\sharp\nu)$ can be computed in linear time $\mathcal{O}(n \log n)$ where n is the number of supports of μ and ν. However, due to the integration over the unit sphere, the sliced Wasserstein distance does not have closed-form expression. To approximate the intractable expectation, Monte Carlo scheme is used, namely, we draw uniform samples $\theta_1, \ldots, \theta_L \sim \mathcal{U}(\mathbb{S}^{d-1})$ from the unit sphere

Algorithm 1 Max-SW

```
**Input:** Probability measures: $\mu, \nu$, learning rate $\eta$, max number of iterations $T$.
Initialize $\theta$
**while** $\theta$ not converge or reach $T$ **do**
  $\theta = \theta - \nabla_\theta \mathrm{W}_p(\theta\sharp\mu, \theta\sharp\nu)$
  $\theta = \frac{\theta}{||\theta||_2}$
**end while**
**Return:** $\theta$
```

and obtain the following approximation:

$$\mathrm{SW}_p(\mu, \nu) \approx \left(\frac{1}{L} \sum_{i=1}^{L} \mathrm{W}_p^p(\theta_i\sharp\mu, \theta_i\sharp\nu) \right)^{\frac{1}{p}}. \tag{3}$$

In practice, L should be chosen to be sufficiently large compared to the dimension d. It is not appealing since the computational complexity of SW is linear with L. To reduce projection complexity, max-sliced Wasserstein (Max-SW) is introduced [10] . In particular, the max-sliced Wasserstein distance between μ and ν is given by:

$$\text{Max-SW}(\mu, \nu) := \max_{\theta \in \mathbb{S}^{d-1}} \mathrm{W}_p(\theta\sharp\mu, \theta\sharp\nu). \tag{4}$$

To solve the optimization problem, a projected gradient descent procedure is used. We present a simple algorithm in Algorithm 1. In practice, practitioners often set a fixed number of gradient updates, e.g., $T = 100$.

2.2 Learning Generative Models with Max-Sliced Wasserstein and Amortized Optimization

We now provide an application of (sliced)-Wasserstein distances to generative models settings. The problem can be seen as the following optimization:

$$\min_{\phi \in \Phi} \mathcal{D}(\mu, \nu), \tag{5}$$

where $\mathcal{D}(\cdot, \cdot)$ can be Wasserstein distance or SW distance or Max-SW distance. Despite the recent progress on scaling up Wasserstein distance in terms of the size of supports of probability measures [1, 29], using the original form of Wasserstein distances is still not tractable in real training due to both the memory constraint and time constraint. In more detail, the number of training samples is often huge,e.g., one million, and the dimension of data is also large,e.g., ten thousand. Therefore, mini-batch losses based on Wasserstein distances have been proposed [50, 12, 13, 38, 39]. The corresponding population form of these losses between two probability measures μ and ν is:

$$\tilde{\mathcal{D}}(\mu, \nu) := \mathbb{E}_{X, Y \sim \mu^{\otimes m} \otimes \nu^{\otimes m}} \mathcal{D}(P_X, P_Y), \tag{6}$$

where $m \geq 1$ is the mini-batch size and $\mathcal{D}$ is a Wasserstein metric.

Algorithm 2 Training generative models with mini-batch max-sliced Wasserstein

```
Input: Data probability measure $\mu$, model learning rate $\eta_1$, slice learning rate $\eta_2$, model maximum number of iterations $T_1$, slice maximum number of iterations $T_2$, number of mini-batches $k$ (is often set to 1).
Initialize $\phi$, the model probability measure $\nu_\phi$
while $\phi$ not converge or reach $T_1$ do
  $\nabla_\phi = 0$
  Sample $(X_1, Y_{\phi,1}), \ldots, (X_k, Y_{\phi,k}) \sim \mu^{\otimes m} \otimes \nu_\phi^{\otimes m}$
  for $i = 1$ to $k$ do
    while $\theta$ not converge or reach $T_2$ do
      $\theta = \theta - \nabla_\theta \mathrm{W}_p(\theta\sharp P_{X_i}, \theta\sharp P_{Y_{\phi,i}})$
      $\theta = \frac{\theta}{||\theta||_2}$
    end while
    $\nabla_\phi = \nabla_\phi + \frac{1}{k}\nabla_\phi \mathrm{W}_p(\theta\sharp P_{X_i}, \theta\sharp P_{Y_{\phi,i}})$
  end for
  $\phi = \phi - \nabla_\phi$
end while
Return: $\phi, \nu_\phi$
```

In the generative model context [17], a stochastic gradient of the parameters of interest is utilized to update these parameters, namely,

$$\nabla_\phi \tilde{\mathcal{D}}(\mu, \nu_\phi) \approx \frac{1}{k}\sum_{i=1}^{k} \nabla_\phi \mathcal{D}(P_{X_i}, P_{Y_{\phi,i}}), \tag{7}$$

where k is the number of mini-batches (is often set to 1), and (X_i, Y_{ϕ_i}) is i.i.d samples from $\mu^{\otimes m} \otimes \nu_\phi^{\otimes m}$. The exchangeability between derivatives and expectation, unbiasedness of the stochastic gradient is proven in [13]. Mini-batch losses are not distances; however, we can derive mini-batch energy distances from them [48].

Learning generative models via max-sliced Wasserstein: As we mentioned in Section 2.1, the max-sliced Wasserstein distance can overcome the curse of dimensionality of the Wasserstein distance and the issues of Monte Carlo samplings in sliced-Wasserstein distance. Therefore, it is an appealing divergence for learning generative models. By replacing the Wasserstein metric in equation (6), we arrive at the following formulation of the mini-batch max-sliced Wasserstein, which is given by:

$$\text{m-Max-SW}(\mu, \nu) = \mathbb{E}_{X,Y \sim \mu^{\otimes m} \otimes \nu^{\otimes m}} \left[\max_{\theta \in \mathbb{S}^{d-1}} \mathrm{W}_p(\theta\sharp P_X, \theta\sharp P_Y) \right]. \tag{8}$$

Here, we can observe that each pair of mini-batch contains its own optimization problem of finding the "max" slice. Placing this in the context of iterative training of generative models, we can foresee its expensive computation. For a better understanding, we present an algorithm of training generative models with mini-batch max-sliced Wasserstein in Algorithm 2. In practice, there are some modifications of training generative models with mini-batch Max-SW for dealing with unknown metric space [11]. We defer the details of these modifications in Appendix B.

Amortized optimization: A natural question appears: "How can we avoid the nested-loop in mini-batch Max-SW due to several local optimization problems?". In this paper, we propose a practical solution for this problem, which is known as *amortized optimization* [2]. In amortized optimization, instead of solving all optimization problems independently, an amortized model is trained to predict optimal solutions to all problems. We now state the adapted definition of amortized models based on that in [2]:

Definition 1. *For each context variable x in the context space $\mathcal{X}$, $\theta^\star(x)$ is the solution of the optimization problem $\theta^\star(x) = \arg\min_{\theta\in\Theta}\mathcal{L}(\theta, x)$, where Θ is the solution space. A parametric function $f_\psi : \mathcal{X} \to \Theta$, where $\psi \in \Psi$, is called an amortized model if*

$$f_\psi(x) \approx \theta^\star(x), \quad \forall x \in \mathcal{X}. \tag{9}$$

The amortized model is trained by the amortized optimization objective which is defined as:

$$\min_{\psi\in\Psi} \mathbb{E}_{x\sim p(x)}\mathcal{L}(f_\psi(x), x), \tag{10}$$

where $p(x)$ is a probability measure on $\mathcal{X}$ which measures the "importance" of optimization problems.

The amortized model in Definition 1 is sometimes called a *fully* amortized model for a distinction with the other concept of *semi* amortized model [2]. The gap between the predicted solution and the optimal solution $\mathbb{E}_{x\sim p(x)}||f_\psi(x) - \theta^\star(x)||_2$ is called the amortization gap. However, understanding this gap depends on specific configurations of the objective $\mathcal{L}(\cdot, x)$, such as convexity and smoothness, which are often non-trivial to obtain in practice.

3 Amortized Sliced Wasserstein

In this section, we discuss an application of amortized optimization to the mini-batch max-sliced Wasserstein. In particular, we first formulate the approach into a novel family of mini-batch losses, named *Amortized Sliced Wasserstein*. Each member of this family utilizes an amortized model for predicting informative slicing directions of mini-batch measures. We then propose several useful amortized models in practice, including linear model, generalized linear model, and non-linear model.

3.1 Amortized Sliced Wasserstein and Amortized Models

We extend the definition of the mini-batch max-sliced Wasserstein in equation (8) with the usage of an amortized model to obtain the amortized sliced Wasserstein as follows.

Definition 2. *Let $p \geq 1$, $m \geq 1$, and μ, ν are two probability measure in $\mathcal{P}(\mathbb{R}^d)$. Given an amortized model $f_\psi : \mathbb{R}^{dm} \times \mathbb{R}^{dm} \to \mathbb{S}^{d-1}$ where $\psi \in \Psi$, the* amortized sliced Wasserstein *between μ and ν is defined as:*

$$\mathcal{A}\text{-}SW(\mu, \nu) := \max_{\psi\in\Psi} \mathbb{E}_{(X,Y)\sim\mu^{\otimes m}\otimes\nu^{\otimes m}}[W_p(f_\psi(X, Y)\sharp P_X, f_\psi(X, Y)\sharp P_Y)]. \tag{11}$$

From the definition, we can see that the amortized model maps each pair of mini-batches to the optimal projecting direction on the unit hypersphere between two corresponding mini-batch probability measures. We have the following result about the symmetry and positivity of the amortized sliced Wasserstein.

Proposition 1. *The amortized sliced Wasserstein losses are positive and symmetric. However, they are not metrics since they do not satisfy the identity property, namely,* $\mathcal{A}\text{-}SW(\mu,\nu) = 0 \not\Leftrightarrow \mu = \nu$.

Proof of Proposition 1 is in Appendix A.1. Our next result indicates that we can upper bound the amortized sliced Wasserstein in terms of mini-batch max-sliced Wasserstein.

Proposition 2. *The amortized sliced Wasserstein losses are lower-bounds of the mini-batch max-sliced Wasserstein loss, namely,* $\mathcal{A}\text{-}SW(\mu,\nu) \leq m\text{-}Max\text{-}SW(\mu,\nu)$ *for all probability measures* μ *and* ν *on* $\mathbb{R}^d$.

Proof of Proposition 2 is in Appendix A.2.

Parametric forms of the amortized model: Now we define three types of amortized models that we will use in the experiments.

Definition 3. *Given* $X, Y \in \mathbb{R}^{dm}$, *and the one-one "reshape" mapping* $T : \mathbb{R}^{dm} \to \mathbb{R}^{d\times m}$, *the* linear amortized model *is defined as:*

$$f_\psi(X,Y) := \frac{w_0 + T(X)w_1 + T(Y)w_2}{||w_0 + T(X)w_1 + T(Y)w_2||_2^2}, \tag{12}$$

where $w_1, w_2 \in \mathbb{R}^m$, $w_0 \in \mathbb{R}^d$ *and* $\psi = (w_0, w_1, w_2)$.

In Definition 3, the assumption is that the optimal projecting direction lies on the subspace that is spanned by the basis $\{x_1, \ldots, x_m, y_1, \ldots, y_m, w_0\}$ where $X = (x_1, \ldots, x_m)$ and $Y = (y_1, \ldots, y_m)$. The computational complexity of this function is $\mathcal{O}((2m+1)d)$ since those of the operators $T(X)w_1$ and $T(Y)w_2$ are $\mathcal{O}(md)$ while adding the bias w_0 costs an additional computational complexity $\mathcal{O}(d)$. The number of parameters in linear amortized model is $2m + d$.

To increase the expressiveness of the linear amortized model, we apply some (non-linear) mappings to the inputs X and Y, which results in the generalized linear amortized model as follows.

Definition 4. *Given* $X, Y \in \mathbb{R}^{dm}$, *and the one-one "reshape" mapping* $T : \mathbb{R}^{dm} \to \mathbb{R}^{d\times m}$, *the* generalized linear amortized model *is defined as:*

$$f_\psi(X,Y) := \frac{w_0 + T(g_{\psi_1}(X))w_1 + T(g_{\psi_1}(Y))w_2}{||w_0 + T(g_{\psi_1}(X))w_1 + T(g_{\psi_1}(Y))w_2||_2^2}, \tag{13}$$

where $w_1, w_2 \in \mathbb{R}^m$, $w_0 \in \mathbb{R}^d$, $\psi_1 \in \Psi_1$, $g_{\psi_1} : (\mathbb{R}^d)^{\otimes m} \to (\mathbb{R}^d)^{\otimes m}$ *and* $\psi = (w_0, w_1, w_2, \psi_1)$.

In Definition 4, the assumption is that the optimal projecting direction lies on the subspace that is spanned by the basis $\{x_1', \ldots, x_m', y_1', \ldots, y_m'\}$ where $g_{\psi_1}(X) = (x_1', \ldots, x_m')$ and $g_{\psi_1}(Y) = (y_1', \ldots, y_m')$. To specify, we let $g_{\psi_1}(X) = (W_2\sigma(W_1x_1) + b_0, \ldots, W_2\sigma(W_1x_m) + b_0)$, where $\sigma(\cdot)$ is the Sigmoid function, $W_1 \in \mathbb{R}^{d\times d}$, $W_2 \in \mathbb{R}^{d\times d}$, and $b_0 \in \mathbb{R}^d$. Compared to the linear model, the generalized linear model needs additional computations for $g_\psi(T(X))$ and $g_\psi(T(Y))$, which are at the order of $\mathcal{O}(2m(d^2+d))$. It is because we need to include the complexity for matrix multiplication, e.g., W_1x_1 that costs $\mathcal{O}(d^2)$, for Sigmoid function that costs $\mathcal{O}(d)$, and for adding bias b_0 that costs $\mathcal{O}(d)$. Therefore, the total computational complexity of the function f_ψ is $\mathcal{O}(4md^2 + 6md + d)$ while the number of parameters is $2(m + d^2 + d)$.

We finally propose another amortized model where we instead consider some mapping on the function $\omega_0 + T(X)\omega_1 + T(Y)\omega_2$ in the linear amortized model so as to increase the approximation power of the function f_ψ.

Definition 5. *Given $X, Y \in \mathbb{R}^{dm}$, and the one-one mapping $T : \mathbb{R}^{dm} \to \mathbb{R}^{d\times m}$, the* non-linear amortized model *is defined as:*

$$f_\psi(X, Y) := \frac{h_{\psi_2}(w_0 + T(X)w_1 + T(Y)w_2)}{||h_{\psi_2}(w_0 + T(X)w_1 + T(Y)w_2)||_2^2}, \tag{14}$$

where $w_1, w_2 \in \mathbb{R}^m$, $w_0 \in \mathbb{R}^d$, $\psi_2 \in \Psi_2$, $h_{\psi_2} : \mathbb{R}^d \to \mathbb{R}^d$ and $\psi = (w_0, w_1, w_2, \psi_2)$.

In Definition 5, the assumption is that the optimal projecting direction lies on the image of the function $h_{\psi_2}(\cdot)$ that maps from the subspace spanned by $\{x_1, \ldots, x_m, y_1, \ldots, y_m\}$ where $X = (x_1, \ldots, x_m)$ and $Y = (y_1, \ldots, y_m)$. The computational complexity for $h_{\psi_2}(x) = W_4\sigma(W_3x)) + b_0$ when $x \in \mathbb{R}^d$, $W_3 \in \mathbb{R}^{d\times d}$, $W_4 \in \mathbb{R}^{d\times d}$, and $b_0 \in \mathbb{R}^d$ is at the order of $\mathcal{O}(2(d^2 + d))$. Therefore, the total computational complexity of the function f_ψ is $\mathcal{O}(2md + 2d^2 + 3d)$ while the number of parameters is $2(m + d^2 + d)$.

Using amortized models in Definitions 3-5 leads to three corresponding amortized sliced Wasserstein, which are linear amortized sliced Wasserstein ($\mathcal{LA}$-SW), generalized linear amortized sliced Wasserstein ($\mathcal{GA}$-SW), and non-linear amortized sliced Wasserstein ($\mathcal{NA}$-SW) in turn.

Remark. *The parametric forms in Definitions 3-5 are chosen as they are well-known choices for parametric functions. There are still several other ways of parameterization that can be utilized in practice based on prior knowledge about data, e.g., we can use convolution operator for saving parameters or we can strengthen the dependence between samples via recursive functions. We leave the design of these amortized models for future work.*

3.2 Amortized Sliced Wasserstein Generative Models

Based on the amortized sliced Wasserstein losses, our objective function for training a generative model ν_ϕ parametrized by $\phi \in \Phi$ now becomes:

$$\min_{\phi\in\Phi} \max_{\psi\in\Psi} \mathbb{E}_{(X,Y_\phi)\sim\mu^{\otimes m}\otimes\nu_\phi^{\otimes m}}[\text{W}_p(f_\psi(X, Y_\phi)\sharp P_X, f_\psi(X, Y_\phi)\sharp P_{Y_\phi})] := \min_{\phi\in\Phi} \max_{\psi\in\Psi} \mathcal{L}(\mu, \nu_\phi, \psi). \tag{15}$$

Since the optimization (15) forms a minimax problem, we can use an alternating stochastic gradient descent-ascent algorithm to solve it. In particular, the stochastic gradients of ϕ and ψ can be estimated from mini-batches $(X_1, Y_{\phi,1}) \ldots (X_k, Y_{\phi,k}) \sim \mu^{\otimes m} \otimes \nu_\phi^{\otimes m}$ as follows:

$$\nabla_\phi \mathcal{L}(\mu, \nu_\phi, \psi) = \frac{1}{k}\nabla_\phi \text{W}_p(f_\psi(X_i, Y_{\phi,i})\sharp P_{X_i}, f_\psi(X_i, Y_{\phi,i})\sharp P_{Y_{\phi,i}}), \tag{16}$$

$$\nabla_\psi \mathcal{L}(\mu, \nu_\phi, \psi) = \frac{1}{k}\nabla_\psi \text{W}_p(f_\psi(X_i, Y_{\phi,i})\sharp P_{X_i}, f_\psi(X_i, Y_{\phi,i})\sharp P_{Y_{\phi,i}}). \tag{17}$$

For more details, we present the procedure in Algorithm 3.

Computational complexity: From Algorithm 2 and Algorithm 3, we can see that training with $\mathcal{A}$-SW can escape the inner while-loop for finding the optimal projecting directions. In each iteration of the global while-loop, the computational complexity of computing the mini-batch Max-SW is $\mathcal{O}(2kT_2(m\log m + dm))$, which is composed by k mini-batches with T_2 loops of the projection to one-dimension operator which costs $\mathcal{O}(2dm)$ and the computation of the sliced Wasserstein which costs $\mathcal{O}(2m\log m)$. For the mini-batch sliced Wasserstein, the overall computational complexity is

Algorithm 3 Training generative models with amortized sliced Wasserstein

```
Input: Data probability measure $\mu$, model learning rate $\eta_1$, amortized learning rate $\eta_2$, maximum number of iterations $T$, number of mini-batches $k$ (is often set to 1).
Initialize $\phi$, the model probability measure $\nu_\phi$.
Initialize $\psi$, the amortized model $f_\psi$.
while $\phi, \psi$ not converge or reach $T$ do
  $\nabla_\phi = 0; \nabla_\psi = 0$
  Sample $(X_1, Y_{\phi,1}), \ldots, (X_k, Y_{\phi,k}) \sim \mu^{\otimes m} \otimes \nu_\phi^{\otimes m}$
  for $i = 1$ to $k$ do
    $\nabla_\phi = \nabla_\phi + \frac{1}{k}\nabla_\phi \mathrm{W}_p(f_\psi(X_i, Y_{\phi,i})\sharp P_{X_i}, f_\psi(X_i, Y_{\phi,i})\sharp P_{Y_{\phi,i}})$
    $\nabla_\psi = \nabla_\psi + \frac{1}{k}\nabla_\psi \mathrm{W}_p(f_\psi(X_i, Y_{\phi,i})\sharp P_{X_i}, f_\psi(X_i, Y_{\phi,i})\sharp P_{Y_{\phi,i}})$
  end for
  $\phi = \phi - \nabla_\phi$
  $\psi = \psi + \nabla_\psi$
end while
Return: $\phi, \nu_\phi$
```

$\mathcal{O}(2kL(m\log m + dm))$ where L is the number of projections. For $\mathcal{LA}$-SW, the overall computation complexity is $\mathcal{O}(2k(m\log m + 3md + d))$ where the extra complexity $\mathcal{O}((2m+1)d)$ comes from the computation of $f_\psi(\cdot)$ (see Section 3.1). Similarly, the computational complexities of $\mathcal{GA}$-SW and $\mathcal{NA}$-SW are respectively $\mathcal{O}(2k(m\log m + 4md^2 + 7md + d))$ and $\mathcal{O}(2k(m\log m + 3md + 2d^2 + 3d))$.

Projection Complexity: Compared to the sliced Wasserstein, Max-SW reduces the space for projecting directions from $\mathcal{O}(L)$ to $\mathcal{O}(1)$. For $\mathcal{LA}$-SW, $\mathcal{GA}$-SW, and $\mathcal{NA}$-SW, the projection complexity is also $\mathcal{O}(1)$. However, compared to d parameters of Max-SW, $\mathcal{LA}$-SW needs $2m + d$ parameters for creating the projecting directions while $\mathcal{GA}$-SW and $\mathcal{NA}$-SW respectively need $\mathcal{O}(2(m + d^2 + d))$ parameters for producing the directions (see Section 3.1).

Remark. *The computational complexities and the projection complexities of $\mathcal{GA}$-SW and $\mathcal{NA}$-SW are based on the specific parameterization that we choose in Section 3. We would like to recall that these complexities can be reduced by lighter parameterization as in the remark at the end of Section 3.1.*

4 Related Works

Generalized sliced Wasserstein [23] was introduced by changing the push-forward function from linear $T_\theta(x) = \theta^\top x$ to non-linear $T_\theta(x) = g(\theta, x)$ for some non-linear function $g(\cdot, \cdot)$. To cope with the projection complexity of sliced Wasserstein, a biased approximation based on the concentration of Gaussian projections was proposed in [36]. An implementation technique that utilizes two types of memories for training sliced Wasserstein generative model was introduced in [26]. Augmenting the data to a higher-dimensional space for a better linear separation results in augmented sliced Wasserstein [6]. Projected Robust Wasserstein (PRW) metrics appeared in [41] that finds the best orthogonal linear projecting operator onto $d' > 1$ dimensional space. Riemannian optimization techniques for solving PRW were proposed in [27, 20]. We would like to recall that, amortized optimization techniques can be also applied to the case of PRW, max-K-sliced Wasserstein [9],

Table 1: Summary of FID and IS scores of methods on CIFAR10 (32x32), CelebA (64x64), STL10 (96x96), and CelebA-HQ (128x128). We observe that $\mathcal{A}$-SW losses provide the best results among all the training losses.

Method	CIFAR10 (32x32)		CelebA (64x64)		STL10 (96x96)		CelebA-HQ (128x128)	
	FID (↓)	IS (↑)	FID (↓)	IS (↑)	FID (↓)	IS (↑)	FID (↓)	IS (↑)
SNGAN (baseline)	17.09	8.07	12.41	2.61	59.48	9.29	19.25	2.32
SW	14.11	8.19	10.45	2.70	56.32	10.37	16.17	2.65
Max-SW	34.41	6.52	11.28	2.60	77.40	9.46	29.50	2.36
$\mathcal{LA}$-SW (ours)	**12.51**	8.22	9.82	2.72	**52.08**	**10.52**	**14.94**	2.50
$\mathcal{GA}$-SW (ours)	13.54	8.33	9.21	2.78	53.80	10.40	18.97	2.34
$\mathcal{NA}$-SW (ours)	14.44	**8.35**	**8.91**	**2.82**	53.90	10.14	15.17	**2.72**

sliced divergences [35], and might be applicable for sliced mutual information [16]. For theoretical investigation, statistical guarantees of training generative models with sliced Wasserstein were derived in [37].

Amortized optimization was first introduced in the form of amortized variational inference [22, 44]. Several techniques were proposed to improve the usage of amortized variational inference such as using meta sets in [55], using iterative amortized variational inference in [32], using regularization in [49]. Amortized inference was also applied into many applications such as probabilistic reasoning [15], probabilistic programming [45], and structural learning [5]. However, to the best of our knowledge, it is the first time that amortized optimization is used in the literature of optimal transport. We refer to [2] for a tutorial about the amortized optimization.

5 Experiments

In this section, we focus on comparing $\mathcal{A}$-SW generative models with SNGAN [34], the sliced Wasserstein generator [11], and the max-sliced Wasserstein generator [10]. The parameterization of model distribution is based on the neural network architecture of SNGAN [34]. The detail of the training processes of all models is given in Appendix B. For datasets, we choose standard benchmarks such as CIFAR10 (32x32) [25], STL10 (96x96) [7], CelebA (64x64), and CelebAHQ (128x128) [31]. For quantitative comparison, we use the FID score [18] and the Inception score (IS) [47]. We also show some randomly generated images from different models for qualitative comparison. We give full experimental results in Appendix C. The detailed settings about architectures, hyperparameters, and evaluation of FID and IS are given in Appendix D. We would like to recall that all losses that are used in this section are in their mini-batch version.

We first demonstrate the quality of using $\mathcal{A}$-SW in the training generative model compared to the baseline SNGAN, and other mini-batch sliced Wasserstein variants. Then, we investigate the convergence of generative models trained by different losses including the standard SNGAN's loss, mini-batch SW, mini-batch Max-SW, and $\mathcal{A}$-SW by looking at their FID scores and IS scores over training epochs. After that, we compare models qualitatively by showing their randomly generated images. Finally, we report the training speed (number of training iterations per second) and the training memory (megabytes) of all settings of all training losses.

Summary of FID and IS scores: We show FID scores and IS scores of all models at the

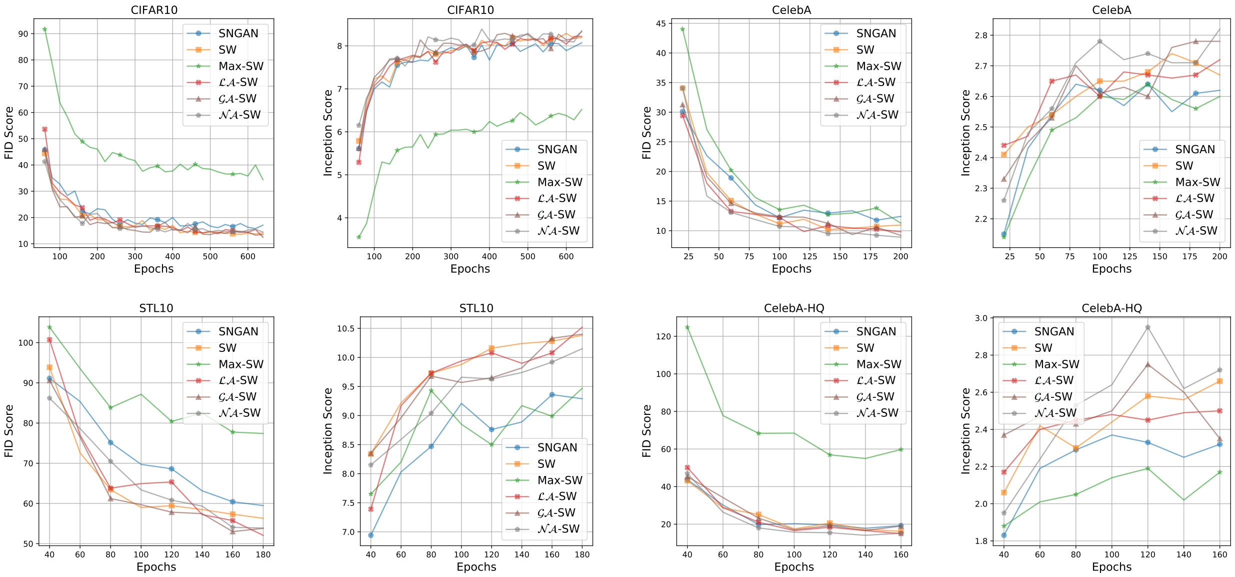
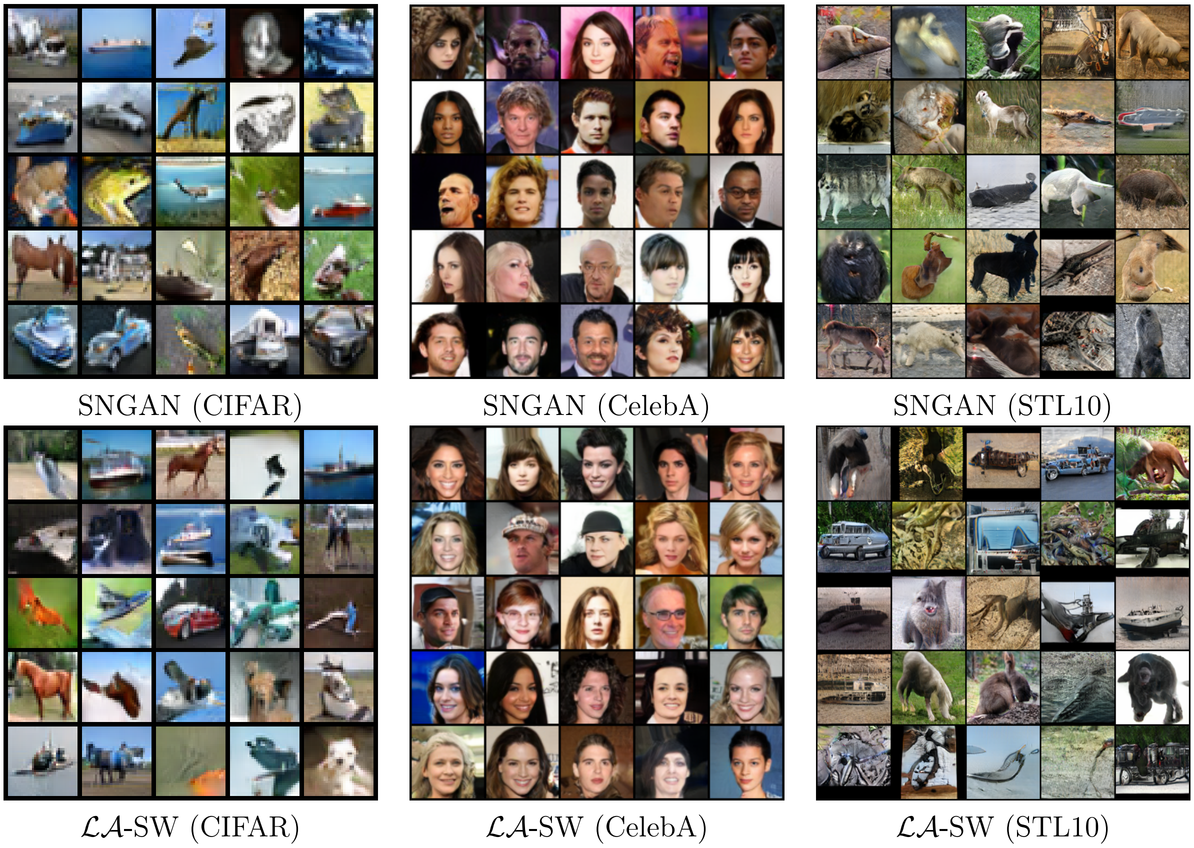


Figure 1: FID scores and IS scores over epochs of different training losses on datasets. We observe that members of $\mathcal{A}$-SW usually help the generative models converge faster.

last training step on all datasets in Table 1. For SW and Max-SW, we select the best setting of hyperparameters for each score. In particular, we search for the best setting of the number of projections $L \in \{1, 100, 1000, 10000\}$. Also, we do a grid search on two hyperparameters of Max-SW, namely, the slice maximum number of iterations $T_2 \in \{1, 10, 100\}$ and the slice learning rate $\eta_2 \in \{0.001, 0.01, 0.1\}$. The detailed FID scores and IS scores for all settings are reported in Table 3 in Appendix C. For amortized models, we fix the slice learning rate $\eta_2 = 0.01$. From the table, we observe that the best amortized model always provides lower FID scores and IS scores than SNGAN, SW, and Max-SW on all datasets of multiple image resolutions. We would like to recall that, SNGAN is reported to be better than WGAN [3] in [34]. We see that the best generative models trained by $\mathcal{A}$-SW are always better than models trained with SNGAN, SW, and Max-SW. Interestingly, the $\mathcal{LA}$-SW performs consistently well compared to other members of $\mathcal{A}$-SW. Also, we observe that Max-SW performs worse than both $\mathcal{A}$-SW and SW. This might be because the local optimization of Max-SW gets stuck at some bad optimum. However, we would like to recall that Max-SW is still better than SW with $L = 1$ (see Table 3 in Appendix C). It emphasizes the benefit of searching for good direction for projecting.

FID and IS scores over training epochs: We show the values of FID scores and Inception scores over epochs on CIFAR10, CelebA, STL10, and CelebA-HQ in Figure 1. According to the figures in Figure 1, we observe that using SW and $\mathcal{A}$-SW helps the generative models converge faster than SNGAN. Moreover, FID lines of $\mathcal{A}$-SW are usually under the lines of other losses and the IS lines of $\mathcal{A}$-SW are usually above the lines of other's. Therefore, $\mathcal{A}$-SW losses including $\mathcal{LA}$-SW, $\mathcal{GA}$-SW, and $\mathcal{NA}$-SW can improve the convergence of training generative models.

Generated images: We show generated images on CIFAR10, CelebA, STL10, and CelebA-HQ from SNGAN and $\mathcal{LA}$-SW in Figure 2 and Figure 3 as a qualitative comparison. We show generated images for other models in Appendix C. From these images, we observe that the quality of generated images is consistent with the FID scores and the IS scores. Therefore, it reinforces the benefits of



SNGAN (CIFAR) SNGAN (CelebA) SNGAN (STL10)

$\mathcal{LA}$-SW (CIFAR) $\mathcal{LA}$-SW (CelebA) $\mathcal{LA}$-SW (STL10)

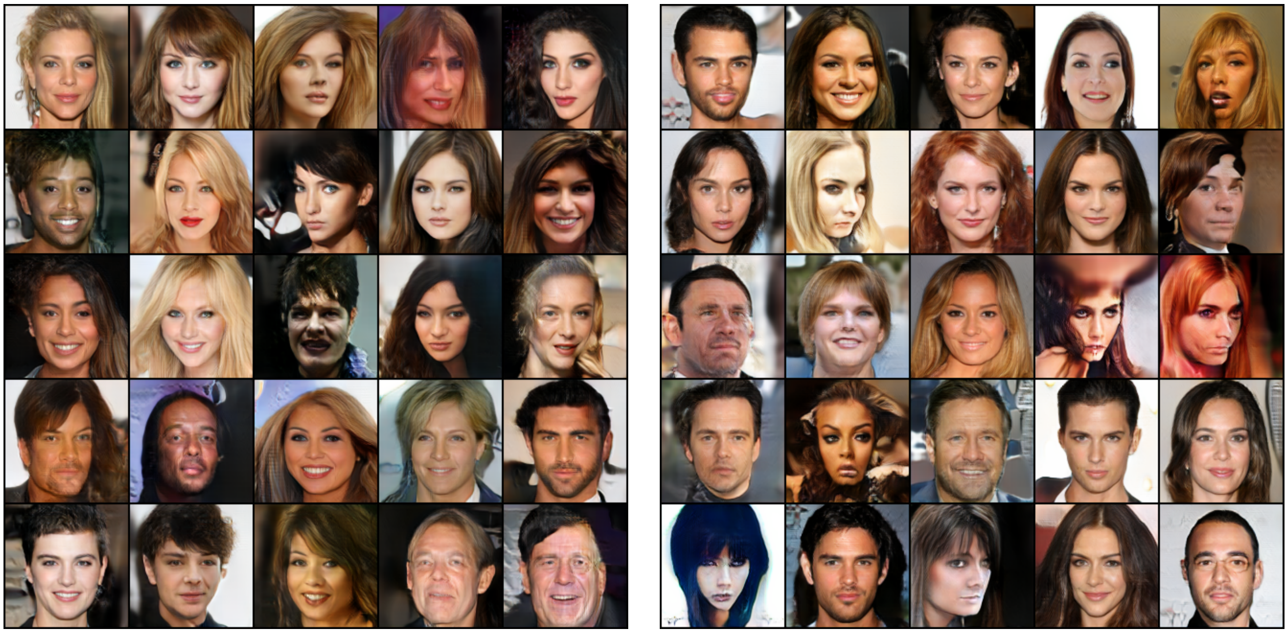
Figure 2: Random generated images of SGAN and $\mathcal{LA}$-SW from CIFAR10, CelebA, and STL10.

Table 2: Computational time and memory of methods (reported in the number of iterations per a second and megabytes (MB).

Method	CIFAR10 (32x32)		CelebA (64x64)		STL10 (96x96)		CelebA-HQ	
	Iters/s (↑)	Mem (↓)	Iters/s (↑)	Mem (↓)	Iters/s (↑)	Mem (↓)	Iters/s (↑)	Mem (↓)
SNGAN (baseline)	19.97	1740	6.31	6713	9.33	3866	10.41	3459
SW (L=1)	18.73	2078	6.17	8011	9.31	4597	10.25	4111
SW (L=100)	18.42	2093	6.15	8015	9.11	4609	10.17	4120
SW (L=1000)	14.96	2112	6.13	8047	9.03	4616	9.63	4143
SW (L=10000)	5.84	2421	4.21	8353	6.50	4780	5.17	4428
Max-SW (T_2=1)	18.61	2078	6.17	8011	9.23	4597	10.22	4111
Max-SW (T_2=10)	18.16	2078	6.15	8011	9.17	4597	10.16	4111
Max-SW (T_2=100)	13.47	2078	5.78	8011	8.32	4597	8.13	4111
$\mathcal{LA}$-SW (ours)	18.58	2086	6.17	8021	9.23	4600	10.19	4115
$\mathcal{GA}$-SW (ours)	17.27	4151	6.07	10083	9.08	5251	10.11	6163
$\mathcal{NA}$-SW (ours)	17.67	4134	6.13	10068	9.11	5249	10.15	6152

using $\mathcal{A}$-SW to train generative models. Again, we would like to recall that all generated images are completely random without cherry-picking.

Computational time and memory: We report the number of training iterations per second and



SNGAN (CelebA-HQ) $\mathcal{LA}$-SW (CelebA-HQ)

Figure 3: Random generated images of SNGAN and $\mathcal{LA}$-SW from CelebAHQ.

the memory in megabytes (MB) in Table 2. We would like to recall that reported numbers are under some errors due to the state of the computational device. From the table, we see that $\mathcal{LA}$-SW is comparable to Max-SW and SW (L=1) about the computational memory and the computational time. More importantly, $\mathcal{LA}$-SW is faster and consumes less memory than SW ($L \geq 100$) and Max-SW ($T_2 \geq 10$). Compared to SNGAN, $\mathcal{LA}$-SW increases the demand for memory and computation slightly. From $\mathcal{LA}$-SW to $\mathcal{GA}$-SW and $\mathcal{NA}$-SW, the computational time is slower slightly; however, we need between 800 to 2100 MB memory in extra. Again, the additional memory depends on the chosen parameterization (see Section 3). From this table, we can see that using sliced Wasserstein models gives better generative quality but it also costs more computational time and memory. However, $\mathcal{LA}$-SW is the best option of sliced Wasserstein models since it costs the least additional memory and time while it gives consistently good results. We refer to Section 3 for discussion of the time and projection complexities of $\mathcal{A}$-SW.

6 Conclusion

We have proposed using amortized optimization for speeding up the training of generative models that are based on mini-batch sliced Wasserstein with projection optimization. We introduce three types of amortized models including the linear amortized model, the generalized linear amortized model, and the non-linear amortized model for predicting optimal projecting directions between all pairs of mini-batch probability measures. Moreover, using three types of amortized models leads to three corresponding mini-batch losses which are linear amortized sliced Wasserstein, generalized linear amortized sliced Wasserstein, and non-linear amortized sliced Wasserstein. We then show that these losses can improve the result of training deep generative models in both training speed and generative performance. In the future, we will adapt amortized optimization into Wasserstein metrics that are based on orthogonal projecting on linear subspaces.

7 Acknowledgements

This work was partially supported by the NSF IFML 2019844 award and research gifts by UT Austin ML grant to NH.

Supplement to "Amortized Projection Optimization for Sliced Wasserstein Generative Models"

In this supplement, we first collect some proofs in Appendix A . Next, we discuss the training detail of generative models with different mini-batch losses in Appendix B. Moreover, we present detailed results on the deep generative model in Appendix C. Finally, we report the experimental settings including neural network architectures, hyper-parameter choices in Appendix D.

A Proofs

In this appendix, we provide proofs for main results in the main text.

A.1 Proof of Proposition 1

Recall that, the definition of $\mathcal{A}$-SW(μ, ν) is as follows:

$$\mathcal{A}\text{-SW}(\mu,\nu) = \max_{\psi \in \Psi} \mathbb{E}_{(X,Y)\sim \mu^{\otimes m} \otimes \nu^{\otimes m}}[\text{W}_p(f_\psi(X,Y)\sharp P_X, f_\psi(X,Y)\sharp P_Y)].$$

For the symmetric property of the amortized sliced Wasserstein, we have

$$\begin{aligned}
\mathcal{A}\text{-SW}(\nu,\mu) &= \max_{\psi \in \Psi} \mathbb{E}_{(Y,X)\sim \nu^{\otimes m} \otimes \mu^{\otimes m}}[\text{W}_p(f_\psi(Y,X)\sharp P_Y, f_\psi(Y,X)\sharp P_X)] \\
&= \max_{\psi \in \Psi} \mathbb{E}_{(Y,X)\sim \nu^{\otimes m} \otimes \mu^{\otimes m}}[\text{W}_p(f_\psi(Y,X)\sharp P_X), f_\psi(Y,X)\sharp P_Y] \\
&= \max_{\psi \in \Psi} \mathbb{E}_{(Y,X)\sim \nu^{\otimes m} \otimes \mu^{\otimes m}}[\text{W}_p(f_\psi(X,Y)\sharp P_X), f_\psi(X,Y)\sharp P_Y] \\
&= \max_{\psi \in \Psi} \mathbb{E}_{(X,Y)\sim \mu^{\otimes m} \otimes \nu^{\otimes m}}[\text{W}_p(f_\psi(X,Y)\sharp P_X, f_\psi(X,Y)\sharp P_Y)] \\
&= \mathcal{A}\text{-SW}(\mu,\nu),
\end{aligned}$$

where the second equality is because of the symmetry of Wasserstein distance, the third equality is due to the symmetry of $f_\psi(X,Y)$ (see forms of $f_\psi(X,Y)$ in Section 3). The positiveness of $\mathcal{A}$-SW comes directly from the non-negativity of the Wasserstein distance.

To prove that $\mathcal{A}$-SW violates the identity, we use a counter example where $\mu = \nu = \frac{1}{2}\delta_{x_1} + \frac{1}{2}\delta_{x_2}$ $(x_1 \neq x_2)$. In this example, there exists a pair of mini-batches $X = (x_1, x_1)$ and $Y = (x_2, x_2)$. We choose $f_\psi(X,Y) = \frac{x_1+x_2}{||x_1+x_2||_2}$, then $f_\psi(X,Y)\sharp P_X \neq f_\psi(X,Y)\sharp P_Y$ which implies $\text{W}_p(f_\psi(X,Y)\sharp P_X, f_\psi(X,Y)\sharp P_Y) > 0$. Since $\mathcal{A}$-SW defines on the maximum value of $\psi \in \Psi$, $\mathcal{A}\text{-SW}(\mu,\nu) \geq \text{W}_p(f_\psi(X,Y)\sharp P_X, f_\psi(X,Y)\sharp P_Y) > 0$.

Table 3: Summary of FID and IS scores of methods on CIFAR10 (32x32), CelebA (64x64), STL10 (96x96), and CelebA-HQ (128x128).

Method	CIFAR10 (32x32)		CelebA (64x64)		STL10 (96x96)		CelebA-HQ (128x128)	
	FID (↓)	IS (↑)	FID (↓)	IS (↑)	FID (↓)	IS (↑)	FID (↓)	IS (↑)
SNGAN (baseline)	17.09	8.07	12.41	2.61	59.48	9.29	19.25	2.32
SW (L=1)	53.95	5.41	34.47	2.61	144.64	5.82	147.35	2.02
SW (L=100)	15.10	8.10	10.45	2.70	62.44	9.91	17.57	2.43
SW (L=1000)	15.69	8.24	10.96	2.67	57.12	10.25	16.17	2.65
SW (L=10000)	14.11	8.19	10.82	2.66	56.32	10.37	18.08	2.62
Max-SW (T_2=1; η_2=0.001)	34.41	6.52	11.28	2.60	101.37	7.98	34.97	1.98
Max-SW (T_2=10;η_2=0.001)	37.29	6.56	15.98	2.51	77.40	9.46	29.50	2.36
Max-SW (T_2=100; η_2=0.001)	43.20	5.85	16.52	2.46	86.91	9.05	56.20	2.26
Max-SW (T_2=1; η_2=0.01)	45.48	6.09	14.11	2.62	88.29	9.26	43.16	2.36
Max-SW (T_2=10; η_2=0.01)	45.95	6.23	16.89	2.49	90.82	9.18	59.74	2.16
Max-SW (T_2=100; η_2=0.01)	50.73	5.47	12.80	2.70	99.32	8.52	55.94	2.11
Max-SW (T_2=1; η_2=0.1)	36.60	6.58	18.87	2.42	94.33	8.19	52.68	2.16
Max-SW (T_2=10; η_2=0.1)	48.42	6.19	16.22	2.49	90.17	9.70	43.65	2.17
Max-SW (T_2=100; η_2=0.1)	50.74	5.42	14.40	2.59	101.38	8.46	42.81	2.20
$\mathcal{LA}$-SW (ours)	**12.51**	8.22	9.82	2.72	**52.08**	**10.52**	**14.94**	2.50
$\mathcal{GA}$-SW (ours)	13.54	8.33	9.21	2.78	53.80	10.40	18.97	2.34
$\mathcal{NA}$-SW (ours)	14.44	**8.35**	**8.91**	**2.82**	53.90	10.14	15.17	**2.72**

A.2 Proof of Proposition 2

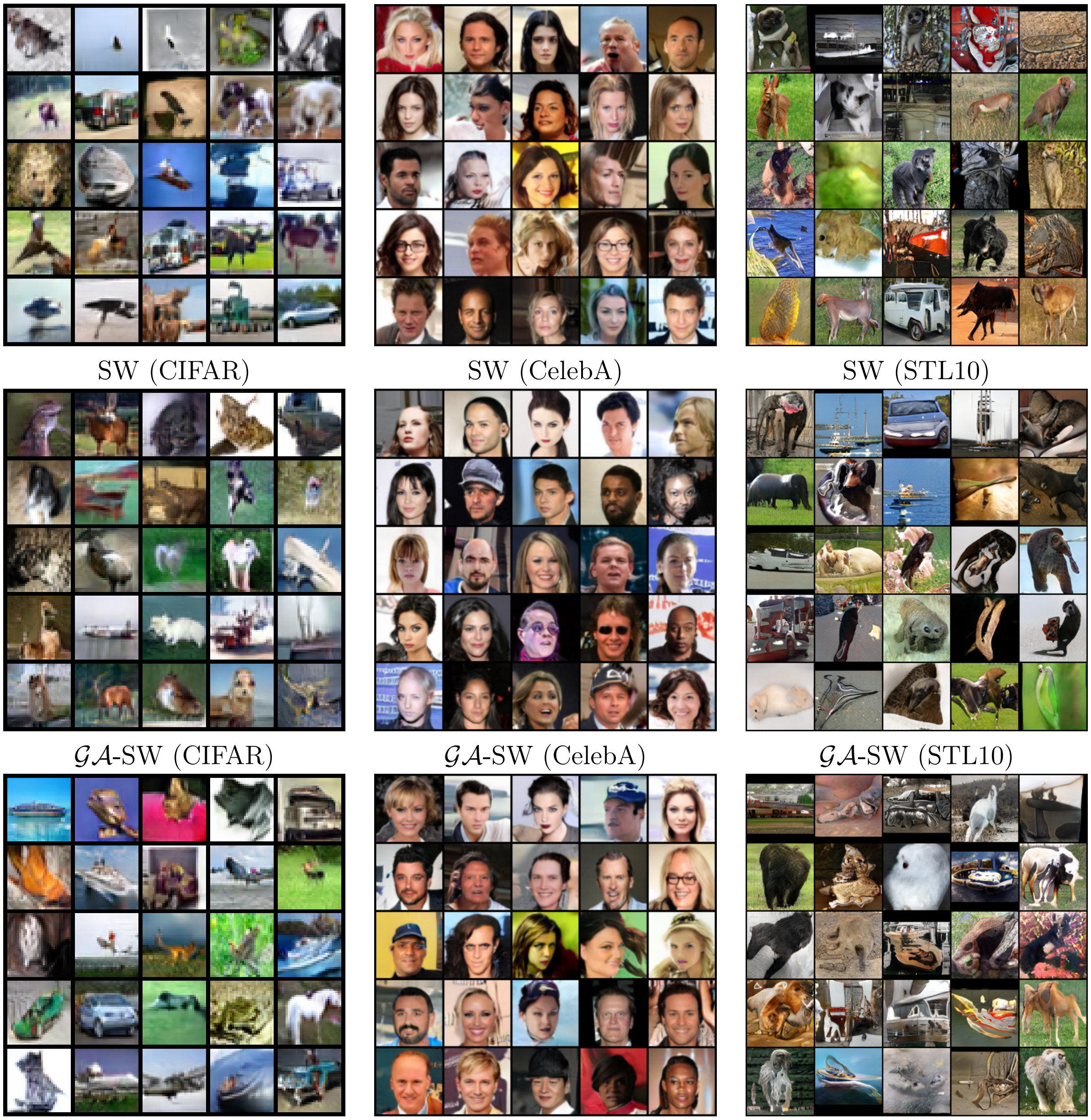
Let $\psi^* := \arg\max_{\psi\in\Psi} \mathbb{E}_{(X,Y)\sim\mu^{\otimes m}\otimes\nu^{\otimes m}}[\mathrm{W}_p(f_\psi(X,Y)\sharp P_X, f_\psi(X,Y)\sharp P_Y)]$. Then, we have

$$\begin{aligned}\mathcal{A}\text{-SW}(\mu,\nu) &= \mathbb{E}_{(X,Y)\sim\mu^{\otimes m}\otimes\nu^{\otimes m}}[\mathrm{W}_p(f_{\psi^*}(X,Y)\sharp P_X, f_{\psi^*}(X,Y)\sharp P_Y)] \\ &= \mathbb{E}_{(X,Y)\sim\mu^{\otimes m}\otimes\nu^{\otimes m}}[\mathrm{W}_p(\theta_{\psi^\star}\sharp P_X, \theta_{\psi^\star}\sharp P_Y)] \\ &\leq \mathbb{E}_{(X,Y)\sim\mu^{\otimes m}\otimes\nu^{\otimes m}}\left[\max_{\theta\in\mathbb{S}^{d-1}} \mathrm{W}_p(\theta\sharp P_X, \theta\sharp P_Y)\right] := \text{m-Max-SW}(\mu,\nu).\end{aligned}$$

As a consequence, we obtain the conclusion of the proposition.

B Training Generative Models

Parametrization: We first discuss the parametrization of the model distribution ν_ϕ. In particular, ν_ϕ is a pushforward probability measure that is created by pushing a unit multivariate Gaussian (ϵ) through a neural network G_ϕ that maps from the realization of the noise to the data space. The detail of the architecture of G_ϕ is given in Appendix D. For training both SNGAN and generative models of SW, Max-SW, and $\mathcal{A}$-SW, we need a second neural network T_β that maps from data space to a single scalar. The second neural network is called *Discriminator* in SGAN or *Feature encoder* in the others. However, the architecture of the second neural network is the same for all models (see Appendix D). For the better distinction between training objectives of SNGAN and the objectives of the others, we denote T_{β_1} is the sub neural network of T_β that maps from the data space to a feature



SW (CIFAR) SW (CelebA) SW (STL10)

$\mathcal{GA}$-SW (CIFAR) $\mathcal{GA}$-SW (CelebA) $\mathcal{GA}$-SW (STL10)

$\mathcal{NA}$-SW (CIFAR) $\mathcal{NA}$-SW (CelebA) $\mathcal{NA}$-SW (STL10)

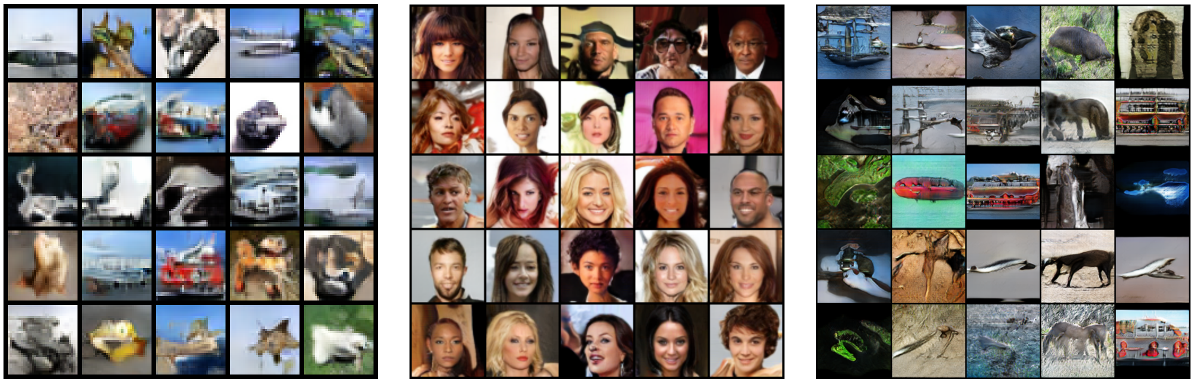
Figure 4: Random generated images of SW, $\mathcal{GA}$-SW, and $\mathcal{NA}$-SW from CIFAR10, CelebA, and STL10.

space (output of the last Resnet block), and T_{β_2} that maps from the feature space (image of T_{β_1}) to a single scalar. More precisely, $T_\beta = T_{\beta_2} \circ T_{\beta_1}$. Again, we specify T_{β_1} and T_{β_1} in Appendix D.

Training SNGAN: Let μ is theta data probability measure, these two optimization problems are done alternatively in training SNGAN:

$$\min_{\beta_1,\beta_2} \left(\mathbb{E}_{x\sim\mu}[\min(0, -1 + T_{\beta_2}(T_{\beta_1}(x)))] + \mathbb{E}_{z\sim\epsilon}[\min(0, -1 - T_{\beta_2}(T_{\beta_1}(G_\phi(z))))] \right),$$

$$\min_{\phi} \mathbb{E}_{z\sim\epsilon}[-T_{\beta_2}(T_{\beta_1}(G_\phi(z)))].$$



Max-SW (CIFAR) Max-SW (CelebA) Max-SW (STL10)

Figure 5: Random generated images of Max-SW from CIFAR10, CelebA, and STL10.

Training SW, Max-SW, and $\mathcal{A}$-SW: For training these models, we adapt the framework in [11] to SNGAN, namely, we use these two objectives:

$$\min_{\beta_1,\beta_2} \left(\mathbb{E}_{x\sim\mu}[\min(0,-1+T_{\beta_2}(T_{\beta_1}(x)))] + \mathbb{E}_{z\sim\epsilon}[\min(0,-1-T_{\beta_2}(T_{\beta_1}(G_\phi(z))))]\right),$$
$$\min_{\phi} \tilde{\mathcal{D}}(\tilde{T}_{\beta_1,\beta_2}\sharp\mu, \tilde{T}_{\beta_1,\beta_2}\sharp G_\phi\sharp\epsilon),$$

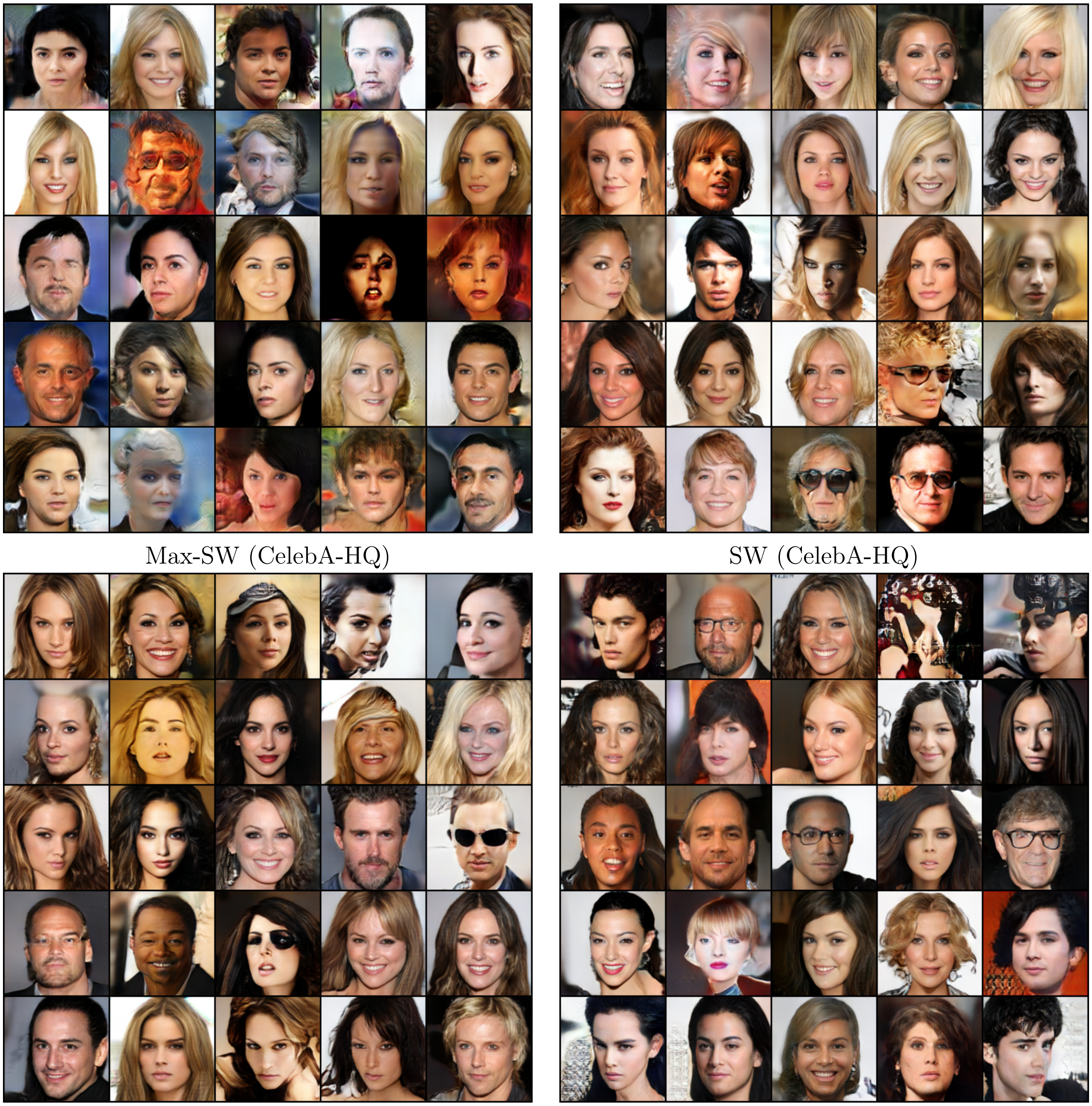
where the function $\tilde{T}_{\beta_1,\beta_2} = [T_{\beta_1}(x), T_{\beta_2}(T_{\beta_1}(x))]$ which is the concatenation vector of $T_{\beta_1}(x)$ and $T_{\beta_2}(T_{\beta_1}(x))$, $\mathcal{D}$ is one of the mini-batch SW, the mini-batch Max-SW (see Equation 8), and $\mathcal{A}$-SW (see Definition 2). This technique is an application of metric learning since $\mathcal{L}_p$ norm is not meaningful on the space of natural images. This observation is mentioned in previous works [11, 14, 52].

Other settings: The information about the mini-batch size, the learning rate, the optimizer, the number of iterations, and so on, are given in Appendix D.

C Full Experimental Results

Detailed FID scores and Inception scores: We first show the detailed FID scores and IS scores of all settings in Table 3. From the table, we can see that the quality of the SW depends on the number of projections. Namely, a higher number of projections often leads to better performance. For Max-SW, we obverse that increasing the number of iterations T_2 might not lead to a lower FID score and a higher IS score. The reason might be that the optimization gets stuck at some local optima. For the choice of the learning rate η_2, we do not see any superior setting for Max-SW.

Generated Images: We show generated images from SW, $\mathcal{GA}$-SW, and $\mathcal{NA}$-SW on CIFAR10, CelebA, and STL10 in Figure 4. The generated images from Max-SW on CIFAR10, CelebA, and STL10 are given in Figure 5. The generated images from SW, Max-SW, $\mathcal{GA}$-SW, and $\mathcal{NA}$-SW on CelebA-HQ are presented in Figure 6. Again, we observe consistent quality results compared to the quantitative results of FID scores and Inception scores.



Max-SW (CelebA-HQ)

SW (CelebA-HQ)

$\mathcal{GA}$-SW (CelebA-HQ)

$\mathcal{NA}$-SW (CelebA-HQ)

Figure 6: Random generated images of Max-SW, SW, $\mathcal{GA}$-SW, and $\mathcal{NA}$-SW from CelebA-HQ.

D Experimental Settings

Neural network architectures: We present the neural network architectures on CIFAR10 in Table 4, CelebA in Table 5, STL10 in Table 6, and CelebA-HQ in Table 7. In summary, we use directly the architectures from `https://github.com/GongXinyuu/sngan.pytorch`.

Table 4: CIFAR10 architectures.

(a) G_ϕ
Input: $\boldsymbol{\epsilon} \in \mathbb{R}^{128} \sim \mathcal{N}(0,1)$
$128 \rightarrow 4 \times 4 \times 256$, dense linear
ResBlock up 256
ResBlock up 256
ResBlock up 256
BN, ReLU, 3×3 conv, 3 Tanh

(b) T_{β_1}
Input: $\boldsymbol{x} \in [-1,1]^{32\times32\times3}$
ResBlock down 128
ResBlock down 128
ResBlock down 128
ResBlock 128
ResBlock 128

(c) T_{β_2}
Input: $\boldsymbol{x} \in \mathbb{R}^{128\times8\times8}$
ReLU
Global sum pooling
$128 \rightarrow 1$ Spectral normalization

Table 5: CelebA architectures.

(a) G_ϕ
Input: $\boldsymbol{\epsilon} \in \mathbb{R}^{128} \sim \mathcal{N}(0,1)$
$128 \rightarrow 4 \times 4 \times 256$, dense linear
ResBlock up 256
ResBlock up 256
ResBlock up 256
ResBlock up 256
ResBlock up 256
BN, ReLU, 3×3 conv, 3 Tanh

(b) T_{β_1}
Input: $\boldsymbol{x} \in [-1,1]^{64\times64\times3}$
ResBlock down 128
ResBlock down 128
ResBlock down 128
ResBlock 128
ResBlock 128
ResBlock 128

(c) T_{β_2}
Input: $\boldsymbol{x} \in \mathbb{R}^{128\times8\times8}$
ReLU
Global sum pooling
$128 \rightarrow 1$ Spectral normalization

Hyper-parameters: For CIFAR10, CelebA, and CelebA-HQ, we set the training iterations to 50000 while we set it to 100000 in STL10. We update T_{β_1} and T_{β_2} every iterations while we update G_ϕ each 5 iterations. The mini-batch size m is set to 128 on CIFAR10 and CelebA, is set to 32 on STL10, is set to 16 on CelebA-HQ. The learning rate of G_ϕ, T_{β_1}, and T_{β_2} is set to 0.0002 and it is decreased linearly through iterations. The optimizers for all optimization problems are Adam [21] with $(\beta_1, \beta_2) = (0, 0.9)$.

FID scores and Inception scores: For these two scores, we calculate them based on 50000 random samples from trained models. For FID scores, the statistics of datasets are calculated on all training samples.

Table 6: STL10 archtectures.

(a) G_ϕ
Input: $\boldsymbol{\epsilon} \in \mathbb{R}^{128} \sim \mathcal{N}(0,1)$
$128 \to 3 \times 3 \times 256$, dense , linear
ResBlock up 256
ResBlock up 256
ResBlock up 256
ResBlock up 256
ResBlock up 256
BN, ReLU, 3×3 conv, 3 Tanh

(b) T_{β_1}
Input: $\boldsymbol{x} \in [-1,1]^{96\times 96\times 3}$
ResBlock down 128
ResBlock down 128
ResBlock down 128
ResBlock down 128
ResBlock 128
ResBlock 128
ResBlock 128

(c) T_{β_2}
Input: $\boldsymbol{x} \in \mathbb{R}^{128\times 6\times 6}$
ReLU
Global sum pooling
$128 \to 1$ Spectral normalization

Table 7: CelebA-HQ archtectures.

(a) G_ϕ
Input: $\boldsymbol{\epsilon} \in \mathbb{R}^{128} \sim \mathcal{N}(0,1)$
$128 \to 4 \times 4 \times 256$, dense , linear
ResBlock up 256
ResBlock up 256
ResBlock up 256
ResBlock up 256
ResBlock up 256
BN, ReLU, 3×3 conv, 3 Tanh

(b) T_{β_1}
Input: $\boldsymbol{x} \in [-1,1]^{128\times 128\times 3}$
ResBlock down 128
ResBlock down 128
ResBlock down 128
ResBlock down 128
ResBlock 128
ResBlock 128
ResBlock 128

(b) T_{β_2}
Input: $\boldsymbol{x} \in \mathbb{R}^{128\times 8\times 8}$
ReLU
Global sum pooling
$128 \to 1$ Spectral normalization

References

[1] J. Altschuler, J. Niles-Weed, and P. Rigollet. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. In *Advances in Neural Information Processing Systems*, pages 1964–1974, 2017. (Cited on pages 2 and 4.)

[2] B. Amos. Tutorial on amortized optimization for learning to optimize over continuous domains. *arXiv preprint arXiv:2202.00665*, 2022. (Cited on pages 6 and 10.)

[3] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017. (Cited on pages 1 and 11.)

[4] N. Bonneel, J. Rabin, G. Peyré, and H. Pfister. Sliced and Radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 1(51):22–45, 2015. (Cited on pages 2 and 3.)

[5] K.-W. Chang, S. Upadhyay, G. Kundu, and D. Roth. Structural learning with amortized inference. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. (Cited on page 10.)

[6] X. Chen, Y. Yang, and Y. Li. Augmented sliced Wasserstein distances. *International Conference on Learning Representations*, 2022. (Cited on page 9.)

[7] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011. (Cited on page 10.)

[8] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, pages 2292–2300, 2013. (Cited on pages 1 and 2.)

[9] B. Dai and U. Seljak. Sliced iterative normalizing flows. In *International Conference on Machine Learning*, pages 2352–2364. PMLR, 2021. (Cited on page 9.)

[10] I. Deshpande, Y.-T. Hu, R. Sun, A. Pyrros, N. Siddiqui, S. Koyejo, Z. Zhao, D. Forsyth, and A. G. Schwing. Max-sliced Wasserstein distance and its use for GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10648–10656, 2019. (Cited on pages 2, 4, and 10.)

[11] I. Deshpande, Z. Zhang, and A. G. Schwing. Generative modeling using the sliced Wasserstein distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3483–3491, 2018. (Cited on pages 2, 5, 10, and 17.)

[12] K. Fatras, Y. Zine, R. Flamary, R. Gribonval, and N. Courty. Learning with minibatch Wasserstein: asymptotic and gradient properties. In *AISTATS 2020-23nd International Conference on Artificial Intelligence and Statistics*, volume 108, pages 1–20, 2020. (Cited on page 4.)

[13] K. Fatras, Y. Zine, S. Majewski, R. Flamary, R. Gribonval, and N. Courty. Minibatch optimal transport distances; analysis and applications. *arXiv preprint arXiv:2101.01792*, 2021. (Cited on pages 4 and 5.)

[14] A. Genevay, G. Peyré, and M. Cuturi. Learning generative models with Sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617. PMLR, 2018. (Cited on pages 1 and 17.)

[15] S. Gershman and N. Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 36, 2014. (Cited on page 10.)

[16] Z. Goldfeld and K. Greenewald. Sliced mutual information: A scalable measure of statistical dependence. *Advances in Neural Information Processing Systems*, 34, 2021. (Cited on page 10.)

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. (Cited on pages 1 and 5.)

[18] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017. (Cited on page 10.)

[19] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. (Cited on page 1.)

[20] M. Huang, S. Ma, and L. Lai. A Riemannian block coordinate descent method for computing the projection robust Wasserstein distance. In *International Conference on Machine Learning*, pages 4446–4455. PMLR, 2021. (Cited on page 9.)

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. (Cited on page 19.)

[22] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. (Cited on pages 1 and 10.)

[23] S. Kolouri, K. Nadjahi, U. Simsekli, R. Badeau, and G. Rohde. Generalized sliced Wasserstein distances. In *Advances in Neural Information Processing Systems*, pages 261–272, 2019. (Cited on page 9.)

[24] S. Kolouri, P. E. Pope, C. E. Martin, and G. K. Rohde. Sliced Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018. (Cited on page 2.)

[25] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. (Cited on page 10.)

[26] J. Lezama, W. Chen, and Q. Qiu. Run-sort-rerun: Escaping batch size limitations in sliced Wasserstein generative models. In *International Conference on Machine Learning*, pages 6275–6285. PMLR, 2021. (Cited on page 9.)

[27] T. Lin, C. Fan, N. Ho, M. Cuturi, and M. Jordan. Projection robust Wasserstein distance and Riemannian optimization. *Advances in Neural Information Processing Systems*, 33:9383–9397, 2020. (Cited on page 9.)

[28] T. Lin, N. Ho, X. Chen, M. Cuturi, and M. I. Jordan. Fixed-support Wasserstein barycenters: Computational hardness and fast algorithm. In *NeurIPS*, pages 5368–5380, 2020. (Cited on page 2.)

[29] T. Lin, N. Ho, and M. Jordan. On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms. In *International Conference on Machine Learning*, pages 3982–3991, 2019. (Cited on pages 2 and 4.)

[30] T. Lin, N. Ho, and M. I. Jordan. On the efficiency of the Sinkhorn and Greenkhorn algorithms and their acceleration for optimal transport. *ArXiv Preprint: 1906.01437*, 2019. (Cited on page 2.)

[31] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015. (Cited on page 10.)

[32] J. Marino, Y. Yue, and S. Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, pages 3403–3412. PMLR, 2018. (Cited on page 10.)

[33] G. Mena and J. Weed. Statistical bounds for entropic optimal transport: sample complexity and the central limit theorem. In *Advances in Neural Information Processing Systems*, 2019. (Cited on page 2.)

[34] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018. (Cited on pages 10 and 11.)

[35] K. Nadjahi, A. Durmus, L. Chizat, S. Kolouri, S. Shahrampour, and U. Simsekli. Statistical and topological properties of sliced probability divergences. *Advances in Neural Information Processing Systems*, 33:20802–20812, 2020. (Cited on page 10.)

[36] K. Nadjahi, A. Durmus, P. E. Jacob, R. Badeau, and U. Simsekli. Fast approximation of the sliced-Wasserstein distance using concentration of random projections. *Advances in Neural Information Processing Systems*, 34, 2021. (Cited on page 9.)

[37] K. Nadjahi, A. Durmus, U. Simsekli, and R. Badeau. Asymptotic guarantees for learning generative models with the sliced-Wasserstein distance. *Advances in Neural Information Processing Systems*, 32, 2019. (Cited on page 10.)

[38] K. Nguyen, D. Nguyen, Q. Nguyen, T. Pham, H. Bui, D. Phung, T. Le, and N. Ho. On transportation of mini-batches: A hierarchical approach. *arXiv preprint arXiv:2102.05912*, 2021. (Cited on page 4.)

[39] K. Nguyen, D. Nguyen, A. Vu, T. Pham, and N. Ho. Improving mini-batch optimal transport via partial transportation. *arXiv preprint arXiv:2108.09645*, 2021. (Cited on page 4.)

[40] S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in Neural Information Processing Systems*, 29, 2016. (Cited on page 1.)

[41] F.-P. Paty and M. Cuturi. Subspace robust Wasserstein distances. In *International Conference on Machine Learning*, pages 5072–5081, 2019. (Cited on page 9.)

[42] G. Peyré and M. Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019. (Cited on page 3.)

[43] G. Peyré and M. Cuturi. Computational optimal transport, 2020. (Cited on page 1.)

[44] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286. PMLR, 2014. (Cited on page 10.)

[45] D. Ritchie, P. Horsfall, and N. D. Goodman. Deep amortized inference for probabilistic programs. *arXiv preprint arXiv:1610.05735*, 2016. (Cited on page 10.)

[46] M. Rowland, J. Hron, Y. Tang, K. Choromanski, T. Sarlos, and A. Weller. Orthogonal estimation of Wasserstein distances. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 186–195. PMLR, 2019. (Cited on page 2.)

[47] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. *Advances in Neural Information Processing Systems*, 29, 2016. (Cited on page 10.)

[48] T. Salimans, H. Zhang, A. Radford, and D. Metaxas. Improving GANs using optimal transport. In *International Conference on Learning Representations*, 2018. (Cited on pages 1 and 5.)

[49] R. Shu, H. H. Bui, S. Zhao, M. J. Kochenderfer, and S. Ermon. Amortized inference regularization. *Advances in Neural Information Processing Systems*, 31, 2018. (Cited on page 10.)

[50] M. Sommerfeld, J. Schrieber, Y. Zemel, and A. Munk. Optimal transport: Fast probabilistic approximation with exact solvers. *Journal of Machine Learning Research*, 20:105–1, 2019. (Cited on page 4.)

[51] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019. (Cited on page 1.)

[52] J. Stanczuk, C. Etmann, L. M. Kreusser, and C.-B. Schönlieb. Wasserstein GANs work because they fail (to approximate the Wasserstein distance). *arXiv preprint arXiv:2103.01678*, 2021. (Cited on pages 2 and 17.)

[53] C. Villani. *Optimal transport: Old and New*. Springer, 2008. (Cited on page 3.)

[54] J. Wu, Z. Huang, D. Acharya, W. Li, J. Thoma, D. P. Paudel, and L. V. Gool. Sliced Wasserstein generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3713–3722, 2019. (Cited on page 2.)

[55] M. Wu, K. Choi, N. Goodman, and S. Ermon. Meta-amortized variational inference and learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6404–6412, 2020. (Cited on page 10.)