Nonparametric Generative Modeling with Conditional Sliced-Wasserstein Flows

Chao Du [1] **Tianbo Li** [1] **Tianyu Pang** [1] **Shuicheng Yan** [1] **Min Lin** [1]

Abstract

Sliced-Wasserstein Flow (SWF) is a promising approach to nonparametric generative modeling but has not been widely adopted due to its suboptimal generative quality and lack of conditional modeling capabilities. In this work, we make two major contributions to bridging this gap. First, based on a pleasant observation that (under certain conditions) the SWF of joint distributions coincides with those of conditional distributions, we propose Conditional Sliced-Wasserstein Flow (CSWF), a simple yet effective extension of SWF that enables nonparametric conditional modeling. Second, we introduce appropriate inductive biases of images into SWF with two techniques inspired by local connectivity and multiscale representation in vision research, which greatly improve the efficiency and quality of modeling images. With all the improvements, we achieve generative performance comparable with many deep parametric generative models on both conditional and unconditional tasks in a purely nonparametric fashion, demonstrating its great potential.

1. Introduction

Deep generative models have made several breakthroughs in recent years (Brock et al., 2018; Durkan et al., 2019; Child, 2020; Song et al., 2020), thanks to the powerful function approximation capability of deep neural networks and the various families of models tailored for probabilistic modeling. One recurring pattern in deep generative models for continuous variables is to map a simple prior distribution to the data distribution (or vice versa). Examples include those performing single-step mappings via a neural network, such as GANs (Goodfellow et al., 2014), VAEs (Kingma & Welling, 2013), and normalizing flows (Rezende & Mohamed, 2015), and those taking iterative steps to transform the distributions characterized by ODEs (Grathwohl et al., 2018) and diffusion SDEs (Song et al., 2020). The latter have achieved great success recently due to the decomposition of complex mappings into multi-step easier ones.

While most existing models train parametric neural networks as the mapping functions, a less popular but promising alternative is to perform nonparametric mappings, for which preliminary attempts (Liutkus et al., 2019; Dai & Seljak, 2021) have been made recently. These works further decompose the mapping between multidimensional distributions into mappings between one-dimensional distributions, which have closed-form solutions based on the optimal transport theory (Villani, 2008). The closed-form solutions enable a nonparametric way to construct the mappings, which makes no (or weak) assumptions about the underlying distributions and is therefore more flexible. A promising work is the Sliced-Wasserstein Flows (SWF) (Liutkus et al., 2019), which achieves generative modeling by building nonparametric mappings to solve the gradient flows in the space of probability distribution.

Despite the potential advantages, nonparametric methods like SWF are still in the early stage of development. First, it is underexplored how they can be applied to various tasks besides unconditional generation, such as conditional generation and image inpainting (Meng et al., 2021). In contrast, parametric probability models have mature techniques for constructing conditional distributions and end-to-end gradient descent training, making tasks such as text-to-image generation and image inpainting more straightforward to implement (Mirza & Osindero, 2014; Van den Oord et al., 2016; Perez et al., 2018). Similar mechanisms have yet to be developed for SWF, obstructing its applications in conditional modeling. Second, the quality and efficiency are still quite limited when applied to high-dimensional data such as images. While parametric models can leverage the sophisticated inductive biases of neural networks (e.g., the U-Net (Ronneberger et al., 2015) in diffusion models) and are able to process full-resolution images directly, the nonparametric counterpart either operates on low-dimensional features processed from a pre-trained auto-encoder (Kolouri et al., 2018; Liutkus et al., 2019) or relies on a carefully designed patch-based approach (Dai & Seljak, 2021). Although several variants have been proposed to incorporate convolutional architectures (Nguyen & Ho, 2022b; Laparra

[1]Sea AI Lab, Singapore. Correspondence to: Chao Du <duchao@sea.com>, Min Lin <linmin@sea.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

et al., 2022), to our knowledge, none has demonstrated empirical success in image generation.

In this work, we propose two improvements for SWF to address the above limitations. First, we extend the framework of SWF for conditional probabilistic modeling based on an empirical observation that the collection of SWFs w.r.t. conditional distributions (approximately) coincide with the SWF w.r.t. the joint distribution, subject to a few conditions that can be easily met. Based on this finding, we propose a simple yet effective algorithm, named *Conditional Sliced-Wasserstein Flows* (CSWF), where we obtain conditional samples by simulating the SWF w.r.t. the joint distribution. CSWF enjoys the same nonparametric advantages as SWF while being able to perform various types of conditional inference, such as class-conditional generation and image inpainting. Second, we introduce the *locally-connected projections* and *pyramidal schedules* techniques to enhance the quality of image generation, motivated by the common notions of local connectivity and pyramidal representation in computer vision. By solving optimal transport problems in domain-specific rather than isotropic directions, we successfully incorporate visual inductive biases into SWF (and our CSWF) for image tasks.

Our method and techniques have made several remarkable achievements. First, to the best of our knowledge, the proposed CSWF is the first nonparametric generative approach that is able to handle general conditional distribution modeling tasks. Second, our proposed techniques greatly improve the generation quality of images and also reveals a general way to introduce inductive biases into SWF and CSWF. Last but not least, we achieve comparable performance to many parametric models on both unconditional and conditional image generation tasks, showing great promise.

2. Related Work

Parametric Generative Models Generative models are one of the core research areas in machine learning. Several classes of parametric generative models are widely studied, including GANs (Goodfellow et al., 2014), VAEs (Kingma & Welling, 2013), flow-based models (Durkan et al., 2019), autoregressive models (Van den Oord et al., 2016), energy-based models (Du & Mordatch, 2019) and diffusion/score-based models (Ho et al., 2020; Song et al., 2020). In this work, we instead study a completely different approach to generative modeling using nonparametric methods.

Generative Models based on Optimal Transport Optimal transport (OT) have been widely adopted in generative modeling (Arjovsky et al., 2017; Gulrajani et al., 2017; Tolstikhin et al., 2017; Kolouri et al., 2018; Deshpande et al., 2018; 2019; Meng et al., 2019; Wu et al., 2019; Knop et al., 2020; Nguyen et al., 2020a;b; Nguyen & Ho, 2022b; Bonet et al., 2021; Nguyen & Ho, 2022a). Meng et al. (2019) build iterative normalizing flows where they identify the most informative directions to construct OT maps in each layer. Arjovsky et al. (2017); Wu et al. (2019) propose to train GANs using different distances based on OT. Nguyen et al. (2020a; 2022) propose difference variants of the sliced-Wasserstein distance and apply them on generative models. These works adopt OT in different dimensions, while they are all parametric and based on end-to-end training.

Nonparametric Generative Models A less popular area of research is nonparametric generative models, which holds a lot of potential. These methods utilize tools from nonparametric statistics such as kernel methods (Shi et al., 2018; Li & Turner, 2017; Zhou et al., 2020), Gaussianization (Chen & Gopinath, 2000), and independent component analysis (Laparra et al., 2011). Meng et al. (2020) propose a trainable Gaussianization layer via kernel density estimation and random rotations. SINF (Dai & Seljak, 2021) iteratively transform between Gaussian and data distribution using sliced OT. Note, however, that none of these nonparametric generative models can perform conditional inference.

Conditional Generative Models Generating samples conditioned on additional input information is crucial for achieving controllable generation. Prior research has successfully demonstrated class-conditional image generation (Mirza & Osindero, 2014; Sohn et al., 2015). Recent advancements in this area have yielded notable success in synthesizing high-quality images from textual input (Ramesh et al., 2022; Saharia et al., 2022; Rombach et al., 2022; Hertz et al., 2022; Feng et al., 2022; Yu et al., 2022). The conditional generative capabilities in these models are facilitated by well-established pipelines that construct parametric conditional probability distributions using deep neural networks. In contrast, our work achieves conditional generation within a nonparametric framework, which is significantly different.

3. Preliminaries

We briefly review some prior knowledge, including optimal transport, the Wasserstein space, and gradient flows. Most results presented here hold under certain mild assumptions that can be easily met in practical problems and we have omitted them for simplicity. For more details, we refer the readers to Ambrosio et al. (2005); Villani (2008).

Notations We denote by $\mathcal{P}(\mathcal{X})$ the set of probability distributions supported on $\mathcal{X} \subseteq \mathbb{R}^d$. Given $p \in \mathcal{P}(\mathcal{X}_1)$ and a measurable function $T : \mathcal{X}_1 \to \mathcal{X}_2$, we denote by $q = T_\sharp p \in \mathcal{P}(\mathcal{X}_2)$ the pushforward distribution, defined as $q(E) = p(T^{-1}(E))$ for all Borel set E of $\mathcal{X}_2$. We slightly abuse the notation and denote both the probability distribution and its probability density function (if exists) by p.

3.1. Optimal Transport

Given two probability distributions $p, q \in \mathcal{P}(\mathbb{R}^d)$ and a cost function $c : \mathbb{R}^d \times \mathbb{R}^d \to [0, \infty]$, the optimal transport (OT) theory (Monge, 1781; Kantorovich, 2006) studies the problem of finding a distribution $\gamma \in \Gamma(p, q)$ such that $\int c(x, x')\mathrm{d}\gamma(x, x')$ is minimal, where $\Gamma(p, q)$ is the set of all *transport plans* between p and q, i.e., the set of all probability distributions on $\mathbb{R}^d \times \mathbb{R}^d$ with marginals p and q.

Under mild conditions, the solution to the OT problem exists and is strongly connected with its dual formulation: $\min_{\gamma\in\Gamma(p,q)} \int c\mathrm{d}\gamma = \max_{\psi\in L^1(p)} \left\{\int \psi \mathrm{d}p + \int \psi^c \mathrm{d}q\right\}$,[1] where a solution ψ to the RHS is called a *Kantorovich potential* between p and q. In particular, for the quadratic cost $c(x, x') = \|x - x'\|_2^2$, the above equation is realized (under certain conditions) by a unique optimal transport plan γ^* and a unique (up to an additive constant) Kantorovich potential ψ, and they are related through $\gamma^* = (\mathrm{id}, T)_\sharp p$, where $T(x) = x - \nabla\psi(x)$ (Benamou & Brenier, 2000). The function $T : \mathbb{R}^d \to \mathbb{R}^d$ is called the optimal *transport map* as it depicts how to transport p onto q, i.e., $q = T_\sharp p$. While the optimal transport map T is generally intractable, in one-dimensional cases, i.e., $p, q \in \mathcal{P}(\mathbb{R})$, it has a closed-form of $T = F_q^{-1} \circ F_p$, where F_p and F_q are the cumulative distribution functions (CDF) of p and q, respectively.

3.2. Wasserstein Distance and Wasserstein Space

For the cost $c(x, x') = \|x - x'\|_2^2$, the OT problem naturally defines a distance, called the 2-Wasserstein distance:

$$W_2(p, q) \triangleq \left(\min_{\gamma\in\Gamma(p,q)} \int \|x - x'\|_2^2 \mathrm{d}\gamma(x, x')\right)^{1/2}. \quad (1)$$

For $W_2(p, q)$ to be finite, it is convenient to consider $\mathcal{P}_2(\mathbb{R}^d) \triangleq \left\{p \in \mathcal{P}(\mathbb{R}^d) : \int \|x\|_2^2 \mathrm{d}p(x) < \infty\right\}$, which is the subset of all probability distributions on $\mathbb{R}^d$ with finite second moments. The set $\mathcal{P}_2(\mathbb{R}^d)$ equipped with the 2-Wasserstein distance W_2 forms an important metric space for probability distributions, called the *Wasserstein space*.

3.3. Gradient Flows in the Wasserstein Space

Gradient flows in metric spaces are analogous to the steepest descent curves in the classical Euclidean space. Given a functional $\mathcal{F} : \mathcal{P}_2(\mathbb{R}^d) \to \mathbb{R}$, a gradient flow of $\mathcal{F}$ in the Wasserstein space is an absolutely continuous curve $(p_t)_{t\geq 0}$ that minimizes $\mathcal{F}$ as fast as possible (Santambrogio, 2017).

The Wasserstein gradient flows are shown to be strongly connected with partial differential equations (PDE) (Jordan et al., 1998). In particular, it is shown that (under proper conditions) the Wasserstein gradient flows $(p_t)_t$ coincide with the solutions of the *continuity equation* $\frac{\partial}{\partial t}p_t + \nabla\cdot(p_t v_t) = 0$, where $v_t : \mathbb{R}^d \to \mathbb{R}^d$ is a time-dependent velocity field (Ambrosio et al., 2005). Moreover, the solution of the continuity equation can be represented by $p_t = (X_t)_\sharp p_0, \forall t \geq 0$, where $X_t(x)$ is defined by the characteristic system of ODEs $\frac{\mathrm{d}}{\mathrm{d}t}X_t(x) = v_t(X_t(x))$, $X_0(x) = x, \forall x \in \mathbb{R}^d$ (Bers et al., 1964). The PDE formulation and the characteristic system of ODEs provide us with an important perspective to analyze and simulate the Wasserstein gradient flows.

3.4. Sliced-Wasserstein Flows

The tractability of the one-dimensional OT and Wasserstein distance motivates the definition of the sliced-Wasserstein distance (Rabin et al., 2011). For any $\theta \in \mathbb{S}^{d-1}$ (the unit sphere in $\mathbb{R}^d$), we denote by $\theta^* : \mathbb{R}^d \to \mathbb{R}$ the orthogonal projection, defined as $\theta^*(x) \triangleq \theta^\top x$. Then, counting all the Wasserstein distance between the projected distributions $\theta^*_\sharp p, \theta^*_\sharp q \in \mathcal{P}(\mathbb{R})$ leads to the sliced-Wasserstein distance:

$$SW_2(p, q) \triangleq \left(\int_{\mathbb{S}^{d-1}} W_2^2(\theta^*_\sharp p, \theta^*_\sharp q)\mathrm{d}\lambda(\theta)\right)^{1/2}, \quad (2)$$

where $\lambda(\theta)$ is the uniform distribution on the sphere $\mathbb{S}^{d-1}$. The SW_2 has many similar properties as the W_2 (Bonnotte, 2013) and can be estimated with Monte Carlo methods. It is therefore often used as an alternative to the W_2 in practical problems (Kolouri et al., 2018; Deshpande et al., 2018).

In this paper, we will focus on the Wasserstein gradient flows of functionals $\mathcal{F}(\cdot) = \frac{1}{2}SW_2^2(\cdot, q)$, where q is a target distribution. Bonnotte (2013) proves that (under regularity conditions on p_0 and q) such Wasserstein gradient flows $(p_t)_{t\geq 0}$ satisfy the continuity equation (in a weak sense):

$$\begin{aligned} &\frac{\partial p_t(x)}{\partial t} + \nabla \cdot (p_t(x)v_t(x)) = 0, \\ &v_t(x) \triangleq -\int_{\mathbb{S}^{d-1}} \psi'_{t,\theta}(\theta^\top x)\cdot\theta \,\mathrm{d}\lambda(\theta), \end{aligned} \quad (3)$$

where $\psi_{t,\theta}$ denotes the Kantorovich potential between the (one-dimensional) projected distributions $\theta^*_\sharp p_t$ and $\theta^*_\sharp q$. Moreover, according to Sec. 3.1 the optimal transport map from $\theta^*_\sharp p_t$ to $\theta^*_\sharp q$ is given by $T_{t,\theta} = F^{-1}_{\theta^*_\sharp q} \circ F_{\theta^*_\sharp p_t}$, which gives $\psi'_{t,\theta}(z) = z - T_{t,\theta}(z) = z - F^{-1}_{\theta^*_\sharp q} \circ F_{\theta^*_\sharp p_t}(z)$.

Liutkus et al. (2019) refers to it as the *sliced-Wasserstein flow* (SWF) and adapts it into a practical algorithm for building generative models of an unknown target distribution q (assuming access to i.i.d. samples). By simulating a similar PDE to Eq. (3),[2] it transforms a bunch of particles sampled from p_0 (e.g., a Gaussian distribution) to match the target distribution q. We recap more details in Appendix B.

[1] $L^1(p)$ is the set of absolutely integrable functions under p and $\psi^c(x') \triangleq \inf_x c(x, x') - \psi(x)$ is called the c-transform of ψ.

[2] The authors originally consider the entropy-regularized SWFs, leading to a similar PDE, with an extra Laplacian term that is often ignored when modeling real data. See more details in Appendix B.

4. Conditional Sliced-Wasserstein Flows

In this section, we present an extended framework of SWFs for *conditional* probability distributions and accordingly propose a practical nonparametric method for *conditional* generative modeling.

Formally, given a dataset $\mathcal{D} \triangleq \{(x_i, y_i)\}_{i=1}^N$ representing N i.i.d. samples from the target distribution $q \in \mathcal{P}_2(\mathcal{X} \times \mathcal{Y})$, where $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^l$ are two related domains (e.g., $\mathcal{X}$ the image space and $\mathcal{Y}$ the set of labels), we aim to model the conditional distributions $q_y \triangleq q(\cdot|y) \in \mathcal{P}_2(\mathcal{X})$ for all $y \in \mathcal{Y}$. We assume the marginal distribution $q(y)$ is known.

4.1. Conditional Sliced-Wasserstein Flows

A straightforward idea is to consider a SWF in $\mathcal{P}_2(\mathcal{X})$ with the target distribution q_y for each $y \in \mathcal{Y}$ separately, which we denote by $(p_{y,t})_{t\geq 0}$ and refer to as the *conditional SWF given* y. Then with a suitable initial $p_{y,0} \in \mathcal{P}_2(\mathcal{X})$, it satisfies:

$$\begin{aligned} &\frac{\partial p_{y,t}(x)}{\partial t} + \nabla \cdot (p_{y,t}(x) v_{y,t}(x)) = 0, \\ &v_{y,t}(x) \triangleq -\int_{\mathbb{S}^{d-1}} \psi'_{y,t,\theta}\left(\theta^\top x\right) \cdot \theta \, \mathrm{d}\lambda(\theta), \end{aligned} \tag{4}$$

where $\psi_{y,t,\theta}$ denotes the Kantorovich potential between the projected conditional distributions $\theta^*_\sharp p_{y,t}$ and $\theta^*_\sharp q_y$. Samples from q_y can be drawn if we can simulate the PDE (4).

However, modeling the conditional SWFs for all $y \in \mathcal{Y}$ separately with Liutkus et al.'s (2019) algorithm can be impractical for at least two reasons. First, it is only feasible for the cases where $\mathcal{Y}$ is a finite set, and every y appears in $\mathcal{D}$ often enough since a different split of the dataset $\mathcal{D}_y \triangleq \mathcal{D} \cap (\mathcal{X} \times \{y\})$ is required for each y. Second, even for a finite $\mathcal{Y}$, the knowledge in different $\mathcal{D}_y$ cannot be shared, making it inefficient and unscalable when $|\mathcal{Y}|$ is large or the distribution over $\mathcal{Y}$ is highly imbalanced.

To overcome these difficulties, it is crucial to enable knowledge sharing and generalization abilities among $\mathcal{Y}$ by exploiting the global information from the joint distribution q, rather than solely the conditional information from q_y.

4.2. Conditional SWFs via the Joint SWF

We instead consider the SWF $(p_t)_{t\geq 0}$ in $\mathcal{P}_2(\mathcal{X} \times \mathcal{Y})$ with the target being the joint distribution q. We refer to it as the *joint SWF*, and write its corresponding PDE below:

$$\begin{aligned} &\frac{\partial p_t(x,y)}{\partial t} + \nabla \cdot (p_t(x,y) v_t(x,y)) = 0, \\ &v_t(x,y) = -\int_{\mathbb{S}^{d+l-1}} \psi'_{t,\theta}\left(\theta_x^\top x + \theta_y^\top y\right) \cdot \begin{bmatrix}\theta_x \\ \theta_y\end{bmatrix} \mathrm{d}\lambda(\theta), \end{aligned} \tag{5}$$

where $\theta = [\theta_x^\top, \theta_y^\top]^\top \in \mathbb{S}^{d+l-1}$ is $(d+l)$-dimensional, $\theta_x \in \mathbb{R}^d$, $\theta_y \in \mathbb{R}^l$, and $\psi_{t,\theta}$ is the Kantorovich potential between $\theta^*_\sharp p_t$ and $\theta^*_\sharp q$. Note that here $v_t : \mathbb{R}^{d+l} \to \mathbb{R}^{d+l}$ is a vector field on $\mathcal{X} \times \mathcal{Y}$. We denote the $\mathcal{X}$- and $\mathcal{Y}$-components of $v_t(x,y)$ by $v_t^{\mathcal{X}}(x,y)$ and $v_t^{\mathcal{Y}}(x,y)$, respectively.

At first glance, the joint SWF $(p_t)_{t\geq 0}$ may only provide us with a possibility to sample from q, but is not obviously helpful for modeling conditional distributions. Interestingly, our empirical observation shows that under the assumption that (i) $p_0(y) = q(y)$ and (ii) the target conditional distribution q_y changes slowly enough w.r.t. y, then for all $t \geq 0$, we have $v_t^{\mathcal{X}}(x,y) \approx v_{y,t}(x)$ and $v_t^{\mathcal{Y}}(x,y) \approx 0$. We are unable to provide a rigorous theoretical justification for the time being. We instead include an illustration in Appendix A.

Intuitively, this means that if the assumptions are met, the evolution of distributions $(p_t)_{t\geq 0}$ characterized by the joint SWF can be factorized into two levels. First, the marginal distributions $p_t(y)$ remain unchanged for all $t \geq 0$ since the velocity has zero $\mathcal{Y}$-component. Second, for each y, the evolution of the conditional distributions $(p_t(x|y))_{t\geq 0}$ coincides with the evolution of $(p_{y,t})_{t\geq 0}$ in the conditional SWF given y described in Sec. 4.1.

Moreover, if we simulate the continuity equation (5) of the joint SWF with particle-based methods (e.g., using the characteristic system), then $v_t^{\mathcal{Y}}(x,y) \approx 0$ implies that the $\mathcal{Y}$-component of each particle will almost stand still, and $v_t^{\mathcal{X}}(x,y) \approx v_{y,t}(x)$ implies that the $\mathcal{X}$-component of each particle will move just as if we are simulating the conditional SWF given y. This provides us with an elegant way to practically model conditional distributions through the joint SWF, as described in the next section.

4.3. Practical Algorithm

Based on our observation, we propose a practical algorithm for conditional probabilistic modeling, dubbed the conditional sliced-Wasserstein flows (CSWF). The basic idea is to first adjust the target distribution q and initialize p_0 properly so that the assumptions are (approximately) met, and then to simulate the joint SWF with a particle-based method.

Initialization To satisfy the assumption made in Sec. 4.2, we can always define a new target distribution $q' \triangleq L_\sharp q$ with $L(x,y) \triangleq (x, \xi y)$, that is, q' is obtained using a simple change of variable that scales the $\mathcal{Y}$-component with a number $\xi > 1$, which we call the *amplifier*. Intuitively, scaling the $\mathcal{Y}$-component by ξ will make the change of $q(x|y)$ (w.r.t. y) ξ times slower. The conditions required can thus be approximately satisfied with a large enough ξ. In practice, given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, this simply means that all we need is to use $\{(x_i, \xi y_i)\}_{i=1}^N$ for subsequent processing. We thus assume below w.l.o.g. that q has met the condition, to keep the notations consistent. Then, we set the initial distribution as $p_0(x,y) = q(y)p_0(x)$, where $p_0(x) = \mathcal{N}(x; 0, I_d)$ is Gaussian. This ensures the initial

marginal distribution over $\mathcal{Y}$ is aligned with the target q.

Particle System The solution $(p_t)_{t\geq 0}$ to the continuity equation (5) can be represented by $(Z_t)_\sharp p_0, \forall t \geq 0$, where $Z_t\colon \mathbb{R}^{d+l} \to \mathbb{R}^{d+l}$ denotes the mapping from (x, y) to $Z_t(x, y)$ characterized by the ODE $\mathrm{d}Z_t(x, y) = v_t(Z_t(x, y))\mathrm{d}t$ with initial condition $Z_0(x, y) \triangleq (x, y)$ (see Sec. 3.3). This intuitively means that we can sample from p_t by simulating the ODE with initial point sampled from p_0.

To estimate the velocity field v_t, we follow Liutkus et al. (2019) to consider a particle system so that p_t (and thus v_t) can be estimated within the system. More precisely, we consider M particles $\{\bar{Z}_t^j = (\bar{X}_t^j, \bar{Y}_t^j) \in \mathbb{R}^{d+l}\}_{j=1}^M$, described by a collection of characteristic ODEs:

$$\mathrm{d}\bar{Z}_t^j = \hat{v}_t(\bar{Z}_t^j)\mathrm{d}t, \quad j = 1, \dots, M, \tag{6}$$

where $\hat{v}_t$ is the velocity field estimated with the empirical distributions $\hat{p}_t \triangleq \frac{1}{M}\sum_{j=1}^M \delta_{\bar{Z}_t^j}$ and $\hat{q} \triangleq \frac{1}{N}\sum_{i=1}^N \delta_{(x_i,y_i)}$. Specifically, given a projection θ, the projected empirical distributions $\theta_\sharp^*\hat{p}_t$ and $\theta_\sharp^*\hat{q}$ become two sets of scalar values. Then, estimating $\psi'_{t,\theta}$ is essentially the problem of fitting one-dimensional distributions (i.e., $F_{\theta_\sharp^*\hat{q}}$ and $F_{\theta_\sharp^*\hat{p}_t}$). We simply estimate the CDFs with linear interpolations between the empirical distribution functions. Finally, the velocity is approximated using a Monte Carlo estimate of the integral:

$$\hat{v}_t(x, y) \triangleq -\frac{1}{H}\sum_h^H \hat{\psi}'_{t,\theta_h}\left(\theta_{h,x}^\top x + \theta_{h,y}^\top y\right) \cdot \begin{bmatrix}\theta_{h,x}\\ \theta_{h,y}\end{bmatrix}, \tag{7}$$

where $\{\theta_h\}_{h=1}^H$ are H i.i.d. samples from the unit sphere $\mathbb{S}^{d+l-1}$ and $\hat{\psi}'_{t,\theta_h}$ is the derivative of the Kantorovich potential computed with the estimated CDFs:

$$\hat{\psi}'_{t,\theta_h}(z) = z - F^{-1}_{\theta_{h\sharp}^*\hat{q}} \circ F_{\theta_{h\sharp}^*\hat{p}_t}(z). \tag{8}$$

Velocity Masking Although the initialization has provided a good approximation of the required conditions, the velocity may still has a small $\mathcal{Y}$-component, which can be accumulated over time t. We correct this by manually setting $\hat{v}_t^{\mathcal{Y}}(x, y) = 0$, which means that only the $\mathcal{X}$-component of each particle is updated during the simulation.

Finally, we adopt the Euler method with step size η to iteratively simulate the particle system (i.e., the characteristic ODEs) for K steps. The particles $\{\bar{Z}_0^j = (\bar{X}_0^j, \bar{Y}_0^j)\}_{j=1}^M$ are initialized by independently sampling $\{\bar{Y}_0^j\}_{j=1}^M$ from $q(y)$ and sampling $\{\bar{X}_0^j\}_{j=1}^M$ from $\mathcal{N}(0, I_d)$. In cases where we do not have access to the true marginal $q(y)$, we can alternatively sample $\{\bar{Y}_0^j\}_{j=1}^M$ from the dataset (with replacement). We describe the overall CSWF algorithm in Algorithm 1. Note that by simulating the particle system, we end up with M conditional samples $\{\bar{X}_K^j\}_{j=1}^M$, which we refer to as the *batched samples*. Once we have simulated a particle system, we can opt to save all the θ and the CDFs and reuse them as a model. Then, we can generate new samples conditioned on any input $y \in \mathcal{Y}$, by following the same pipeline of Algorithm 1 but with the shaded lines skipped. We refer to such samples as the *offline samples*.

Algorithm 1: Conditional Sliced-Wasserstein Flow

Input: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, $\{\bar{Y}_0^j\}_{j=1}^M$, ξ, H, η, K
Output: $\{\bar{X}_K^j\}_{j=1}^M$

// Initialize the $\mathcal{X}$-component
$\{\bar{X}_0^j\}_{j=1}^M \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d)$
// Discretize the ODEs
for $k = 0, \dots, K-1$ **do**
  for $h = 1, \dots, H$ **do**
    // Generate random projections
    $\theta_h \sim \mathrm{Uniform}(\mathbb{S}^{d+l-1})$
    // Estimate the CDFs
    $F_{\theta_{h\sharp}^*\hat{q}} = \mathrm{CDF}(\{\theta_{h,x}^\top x_i + \xi \cdot \theta_{h,y}^\top y_i\}_{i=1}^N)$
    $F_{\theta_{h\sharp}^*\hat{p}_k} = \mathrm{CDF}(\{\theta_{h,x}^\top \bar{X}_k^j + \xi \cdot \theta_{h,y}^\top \bar{Y}_0^j\}_{j=1}^M)$
  // Update the $\mathcal{X}$-component with (7)&(8)
  $\bar{X}_{k+1}^j = \bar{X}_k^j - \eta \cdot \hat{v}_k^{\mathcal{X}}(\bar{X}_k^j, \xi \cdot Y_0^j) \qquad j = 1, \dots, M$

4.4. Discussions

The advantages of considering the joint SWF instead of separate conditional SWFs as described in Sec. 4.1 become more clear now. As one can see, the generalization ability comes from interpolating the empirical CDFs. In our method, we always interpolate the CDFs of the projected joint distributions $\theta_\sharp^*\hat{q}$, suggesting that we are indeed generalizing across both $\mathcal{X}$ and $\mathcal{Y}$. Hence, the estimated velocity field applies to all $y \in \mathcal{Y}$ even if it does not exist in $\mathcal{D}$. Moreover, the CDFs are always estimated using the entire dataset, which means the knowledge is shared for all $y \in \mathcal{Y}$.

The time complexity is discussed here. In each step of the simulation, estimating the empirical CDFs for each projection requires sorting two sets of scalar values, with time complexity $\mathcal{O}(M \log M + N \log N)$. Therefore, the overall time complexity is $\mathcal{O}(KH(M \log M + N \log N))$ and the per-sample complexity is $\mathcal{O}(KH(\log M + \frac{N}{M} \log N))$. For the offline samples, since querying $F_{\theta_{h\sharp}^*\hat{p}_t}$ and $F^{-1}_{\theta_{h\sharp}^*\hat{q}}$ are indeed binary search and indexing operations, the per-sample time complexity is $\mathcal{O}(KH \log M)$. Note that the constant KH can possibly be further reduced by sharing projection between steps, which is left for future work.

Similar to Liutkus et al. (2019), the nonparametric nature of CSWF stems from expressing the CDFs directly with empirical data (e.g., sorted arrays of projections). This makes it fundamentally different from parametric generative models that are typically learned via (stochastic) gradient descent

training and thus implies many potential advantages over them. Notably, when new data samples are observed, the empirical CDFs of the projected data distributions $\theta_\sharp^* \hat{q}$ can be updated perfectly by only insertion operations (in $\mathcal{O}(\log N)$ time), which suggests that CSWF has great potential to be adapted to online methods and bypass the challenges associated with parametric online learning, such as catastrophic forgetting (Kirkpatrick et al., 2017; French, 1999). This is also an exciting direction for follow-up research.

Finally, it is worth noting that by setting $\xi = 0$ the effect of conditions is completely removed and our method falls back to an unconditional variant similar to Liutkus et al. (2019).

5. SWFs with Visual Inductive Biases

In this section, we propose to introduce appropriate inductive biases for image tasks into SWF-based methods via *locally-connected projections* and *pyramidal schedules*. The key idea is to use domain-specific projection distributions instead of the uniform distribution, thus focusing more on the OT problems in critical directions. We shall show below how we adapt our CSWF with these techniques. Adapting the SWF algorithm should then be straightforward.

5.1. Locally-Connected Projections

For an image domain $\mathcal{X} \subseteq \mathbb{R}^d$, We assume that $d = \mathtt{C} \times \mathtt{H} \times \mathtt{W}$, where $\mathtt{H}$ and $\mathtt{W}$ denote the height and width of the image in pixels and $\mathtt{C}$ denotes the number of channels.

We observe that projecting an image $x \in \mathcal{X}$ with uniformly sampled θ is analogous to using a fully-connected layer on the flattened vector of the image in neural networks, in the sense that all pixels contribute to the projected values (or the neurons). On the other hand, it has been widely recognized that local connectivity is one of the key features making CNNs effective for image tasks (Ngiam et al., 2010). This motivates us to use *locally-connected projections*, where θ_x is made sparse so that it only projects a small patch of the image x. Specifically, given a patch size $\mathtt{S}$, we first sample a projection θ_{patch} at the patch level from the $(\mathtt{C} \times \mathtt{S} \times \mathtt{S})$-dimensional sphere. Then, we sample a spatial position $(\mathtt{r}, \mathtt{c})$ (i.e., the row and column indices), which stands for the center of the patch. Finally, we obtain θ_x by embedding θ_{patch} into a $(\mathtt{C} \times \mathtt{H} \times \mathtt{W})$-dimensional zero vector such that $\theta_x^\top x$ is equivalent to the projection of the $\mathtt{S} \times \mathtt{S}$ patch of x centered at $(\mathtt{r}, \mathtt{c})$ using θ_{patch},[3] as illustrated in Fig. 1.

The domain knowledge of $\mathcal{Y}$ can be incorporated into θ_y in a similar way. To obtain the final projection θ, we simply concatenate θ_x and θ_y and normalize it to ensure $\theta \in \mathbb{S}^{d+l-1}$.

[3]Note that such choices of projections result in a non-uniform distribution over $\mathbb{S}^{d-1}$ and thus do not necessarily induce a well-defined metric in $\mathcal{P}_2(\mathbb{R}^d)$. However, it can be practically effective for image data. See more discussion in Nguyen & Ho (2022b).

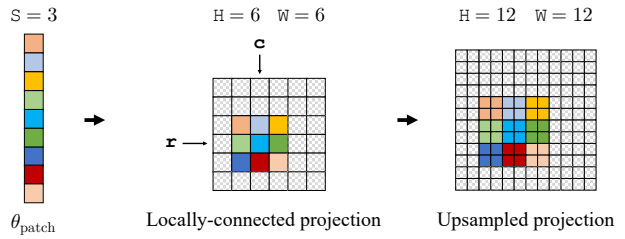


Figure 1. An illustration of locally-connected projection and the upsampled projection used in pyramidal schedules. Left panel shows an example of how we generate a projection θ_x for 6×6 image with patch size 3×3 and spatial position $(\mathtt{r} = 4, \mathtt{c} = 3)$. Right panel shows how we upsample the projection to size 12×12.

5.2. Pyramidal Schedules

Pyramid (or multiscale) representation of images has been widely used in computer vision (Adelson et al., 1984; Lin et al., 2017). In image classification (Krizhevsky et al., 2012), neural networks start from a high-resolution input with details and gradually apply subsampling to obtain low-resolution feature maps that contain high-level information. Image generation usually follows the reverse order, i.e., starts by sketching the high-level structure and completes the details gradually (Ramesh et al., 2022; Jing et al., 2022).

We thus adapt our CSWF to image tasks by introducing *pyramidal schedules*, where we apply locally-connected projections at different resolutions from low to high sequentially. However, due to the dimension-preserving constraint of SWF, instead of working directly on a low-resolution image, we translate the operation to the full-sized image by upsampling the projection filter and modifying the stride parameter accordingly. See Fig. 1 for an illustration and more details in Sec. 6 and Appendix C.

In the following, we refer to our CSWF combined with locally-connected projections and pyramidal schedules as the *locally-connected CSWF* and denote it as ℓ-CSWF. We refer to unconditional SWF (Liutkus et al., 2019) combined with the same techniques as ℓ-SWF.

6. Experiments

In this section, we first examine the efficacy of the proposed techniques of locally-connected projections and pyramidal schedules. We then demonstrate that with these techniques our ℓ-CSWF further enables superior performances on conditional modeling tasks, including class-conditional generation and image inpainting.

We use MNIST, Fashion-MNIST (Xiao et al., 2017), CIFAR-10 (Krizhevsky et al., 2009) and CelebA (Liu et al., 2015) datasets in our experiments. For CelebA, we first center-crop the images to 140×140 according to Song et al. (2020)

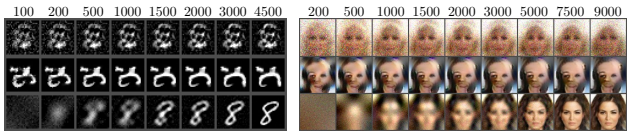


Figure 2. Ablation study of the proposed locally-connected projections and pyramidal schedules. Initially, uniformly sampled projections leads to slow convergence (top rows). Using locally-connected projections, the samples converge rapidly but lose semantic information (middle rows). Further combined with the pyramidal schedules, it is possible to generate high-quality samples quickly (bottom rows). Numbers indicate the simulation steps.

Table 1. FID↓ scores obtained by ℓ-SWF on CIFAR-10 and CelebA. $\diamond$ Use 160×160 center-cropping. $*$ Use 128×128 center-cropping. † Use 140×140 center-cropping.

Method	CIFAR-10	CelebA
Auto-encoder based		
VAE (Kingma & Welling, 2013)	155.7	$85.7^{\diamond}$
SWAE (Wu et al., 2019)	107.9	48.9^{*}
WAE (Tolstikhin et al., 2017)	–	$42^{\dagger}$
CWAE (Knop et al., 2020)	120.0	$49.7^{\dagger}$
Autoregressive & Energy based		
PixelCNN (Van den Oord et al., 2016)	65.9	–
EBM (Du & Mordatch, 2019)	37.9	–
Adversarial		
WGAN (Arjovsky et al., 2017)	55.2	$41.3^{\diamond}$
WGAN-GP (Gulrajani et al., 2017)	55.8	$30.0^{\diamond}$
CSW (Nguyen & Ho, 2022b)	36.8	–
SWGAN (Wu et al., 2019)	17.0	13.2^{*}
Score based		
NCSN (Song & Ermon, 2019)	25.3	–
Nonparametric		
SWF (Liutkus et al., 2019)	> 200	$> 150^{\dagger}$
SINF (Dai & Seljak, 2021)	66.5	37.3^{*}
ℓ-SWF (Ours)	59.7	$38.3^{\dagger}$

and then resize them to 64×64. For all experiments, we set $H = 10000$ for the number of projections in each step and set the step size $\eta = d$. The number of simulation steps K varies from 10000 to 20000 for different datasets, due to different resolutions and pyramidal schedules. For MNIST and Fashion-MNIST, we set $M = 2.5 \times 10^5$. For CIFAR-10 and CelebA, we set $M = 7 \times 10^5$ and $M = 4.5 \times 10^5$, respectively. Additional experimental details and ablation studies are provided in Appendix C & D.1. Code is available at https://github.com/duchao0726/Conditionial-SWF.

6.1. Unconditional Generation

To assess the effectiveness of the inductive biases introduced by the locally-connected projections and the pyramidal schedules, we opt to first evaluate ℓ-SWF on standard unconditional image generation tasks. We do so because this makes more existing generative models comparable since most of them are designed for unconditional generation.

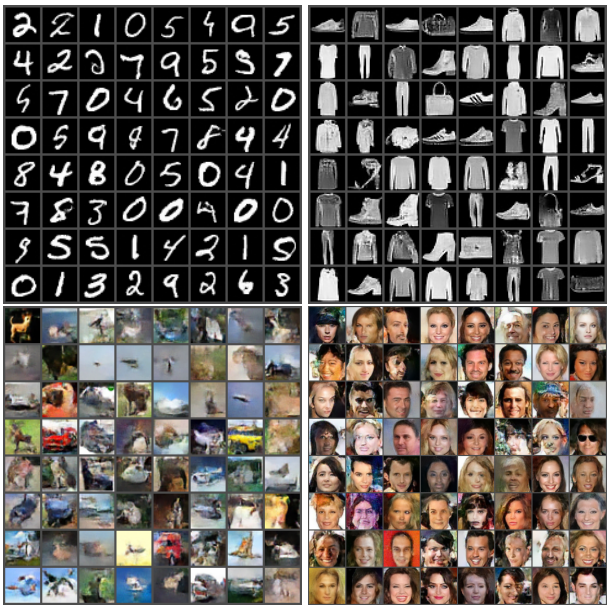


Figure 3. Uncurated batched samples from ℓ-SWF on MNIST, Fashion MNIST, CIFAR-10 and CelebA.

Fig. 3 shows uncurated batched samples (see Sec. 4.3) from ℓ-SWF on MNIST, Fashion MNIST, CIFAR-10 and CelebA. More samples, including their nearest neighbors in the datasets and the offline samples, are shown in Appendix D.2. We observe that the generated images are of high quality. To intuit the effectiveness of locally-connected projections and pyramidal schedules, we show in Fig. 2 an ablation study. It can be observed that with the introduced inductive biases, the number of simulation steps can be greatly reduced, and the generative quality is significantly improved. For comparison, Liutkus et al. (2019) report that uniformly sampled projections (i.e. without the inductive biases) fail to produce satisfactory samples on high-dimensional image data.

We report the FID scores (Heusel et al., 2017) on CIFAR-10 and CelebA in Table 1 for quantitative evaluation. We compare with SWF (Liutkus et al., 2019) and SINF (Dai & Seljak, 2021), which are also iterative methods based on the SW_2 distance. We also include the results of other generative models based on optimal transport for comparison, including SWAE & SWGAN (Wu et al., 2019), WAE (Tolstikhin et al., 2017), CWAE (Knop et al., 2020), CSW (Nguyen & Ho, 2022b), WGAN (Arjovsky et al., 2017) and WGAN-GP (Gulrajani et al., 2017). For better positioning our method in the literature of generative models, we also list results of some representative works, including VAE (Kingma & Welling, 2013), PixelCNN (Van den Oord et al., 2016), EBM (Du & Mordatch, 2019) and NCSN (Song & Ermon, 2019). Our ℓ-SWF significantly outperforms

Figure 4. Class-conditional samples from ℓ-CSWF ($\xi = 10$) on MNIST, Fashion MNIST and CIFAR-10.

Figure 5. Image inpainting results of ℓ-CSWF ($\xi = 1$) on MNIST, Fashion MNIST, CIFAR-10 and CelebA. In each figure, the leftmost column shows the occluded images and the rightmost column shows the original images.

SWF due to the appropriately introduced inductive biases via the techniques described in Sec. 5. On CIFAR-10, it also outperforms SINF (which is layer-wise optimized), probably because SINF requires the projections to be orthogonal, which limits its capability. We include results on CelebA for reference, while we note that different preprocessing make the scores not directly comparable. It is worth noting that, as a nonparametric method that does not require any optimization (e.g. backpropagation), ℓ-SWF achieves comparable results to many elaborate parametric methods such as WGAN and PixelCNN, showing great promise.

6.2. Conditional Modeling

We now demonstrate that, with the help of the introduced inductive biases, our ℓ-CSWF is capable of handling commonly concerned conditional distribution modeling tasks such as class-conditional generation and image inpainting.

6.2.1. Class-Conditional Image Generation

For class-conditional generation tasks, we let $\mathcal{Y}$ be the set of one-hot vectors representing the class labels. The initial $\{\bar{Y}_0^j\}_{j=1}^M$ are sampled according to the label distribution in the dataset (which is categorically uniform for all three datasets tested here). For each projection θ_x, we additionally sample a θ_y uniformly from $\mathbb{S}^{l-1}$ with l being the number of classes and then normalize it together with θ_x, ensuring that $\theta = [\theta_x^\top, \theta_y^\top]^\top$ has unit length. We set the amplifier $\xi = 10$ for all datasets. Other experimental settings, including the hyperparameters and the pyramidal schedules, are the same as in Sec. 6.1. The generated images are shown in Fig. 4. We observe that the samples are of good visual quality and consistent with the class labels. Interestingly, by varying the amplifier ξ, ℓ-CSWF can smoothly transit between class-conditional and unconditional generation, as shown in Appendix D.1.

6.2.2. Image Inpainting

For inpainting tasks, we let $\mathcal{X}$ and $\mathcal{Y}$ represent the pixel spaces of the occluded and observed portions of images, respectively. Since the true marginal $q(y)$ is not available in this setting, we set the initial $\{\bar{Y}_0^j\}_{j=1}^M$ to the partially-observed images created from the dataset. We directly sam-

ple θ (using locally-connected projections) rather than dealing with θ_x and θ_y separately, as both $\mathcal{X}$ and $\mathcal{Y}$ are in the image domain. The amplifier is set to $\xi = 1$ for all datasets. In Fig. 5, we show inpainting results (offline samples) for the occluded images created from the test split of each dataset. We observe that the inpaintings are semantically meaningful and consistent with the given pixels.

7. Conclusions

In this work, we make two major improvements to SWF, a promising type of nonparametric generative model. We first extend SWF in a natural way to support conditional distribution modeling, which opens up the possibility to applications that rely on conditional generation, e.g. text-to-image, image inpainting. On the other hand, we introduce domain-specific inductive biases for image generation, which significantly improves the efficiency and generative quality. Despite being a nonparametric model that does not rely on backpropagation training, our method performs comparably to many parametric models. This promising performance could inspire more further research in this field.

References

Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., and Ogden, J. M. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.

Ambrosio, L., Gigli, N., and Savaré, G. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2005.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

Benamou, J.-D. and Brenier, Y. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.

Bers, L., John, F., and Schechter, M. *Partial differential equations*. American Mathematical Soc., 1964.

Bonet, C., Courty, N., Septier, F., and Drumetz, L. Sliced-wasserstein gradient flows. *arXiv preprint arXiv:2110.10972*, 2021.

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. *arXiv preprint arXiv:1809.11096*, 2018.

Chen, S. and Gopinath, R. Gaussianization. In Leen, T., Dietterich, T., and Tresp, V. (eds.), *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

Child, R. Very deep vaes generalize autoregressive models and can outperform them on images. *arXiv preprint arXiv:2011.10650*, 2020.

Dai, B. and Seljak, U. Sliced iterative normalizing flows. In *International Conference on Machine Learning*, pp. 2352–2364. PMLR, 2021.

Deshpande, I., Zhang, Z., and Schwing, A. G. Generative modeling using the sliced wasserstein distance. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3483–3491, 2018.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10648–10656, 2019.

Du, Y. and Mordatch, I. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Neural spline flows. *Advances in neural information processing systems*, 32, 2019.

Feng, Z., Zhang, Z., Yu, X., Fang, Y., Li, L., Chen, X., Lu, Y., Liu, J., Yin, W., Feng, S., et al. Ernie-vilg 2.0: Improving text-to-image diffusion model with knowledge-enhanced mixture-of-denoising-experts. *arXiv preprint arXiv:2210.15257*, 2022.

French, R. M. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

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*, volume 27, 2014.

Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., and Cohen-Or, D. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

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. *Advances in neural information processing systems*, 30, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Jing, B., Corso, G., Berlinghieri, R., and Jaakkola, T. Subspace diffusion generative models. *arXiv preprint arXiv:2205.01490*, 2022.

Jordan, R., Kinderlehrer, D., and Otto, F. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

Kantorovich, L. V. On the translocation of masses. *Journal of mathematical sciences*, 133(4):1381–1382, 2006.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Knop, S., Spurek, P., Tabor, J., Podolak, I., Mazur, M., and Jastrzebski, S. Cramer-wold auto-encoder. *The Journal of Machine Learning Research*, 21(1):6594–6621, 2020.

Kolouri, S., Pope, P. E., Martin, C. E., and Rohde, G. K. Sliced wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

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

Laparra, V., Camps-Valls, G., and Malo, J. Iterative gaussianization: from ica to random rotations. *IEEE transactions on neural networks*, 22(4):537–549, 2011.

Laparra, V., Hepburn, A., Johnson, J. E., and Malo, J. Orthonormal convolutions for the rotation based iterative gaussianization. *arXiv preprint arXiv:2206.03860*, 2022.

Li, Y. and Turner, R. E. Gradient estimators for implicit models. *arXiv preprint arXiv:1705.07107*, 2017.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

Liutkus, A., Simsekli, U., Majewski, S., Durmus, A., and Stöter, F.-R. Sliced-wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In *International Conference on Machine Learning*, pp. 4104–4113. PMLR, 2019.

Meng, C., Ke, Y., Zhang, J., Zhang, M., Zhong, W., and Ma, P. Large-scale optimal transport map estimation using projection pursuit. *Advances in Neural Information Processing Systems*, 32, 2019.

Meng, C., Song, Y., Song, J., and Ermon, S. Gaussianization flows. In *International Conference on Artificial Intelligence and Statistics*, pp. 4336–4345. PMLR, 2020.

Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.-Y., and Ermon, S. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Mirza, M. and Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Monge, G. *Mémoire sur la théorie des déblais et des remblais*. De l'Imprimerie Royale, 1781.

Ngiam, J., Chen, Z., Chia, D., Koh, P., Le, Q., and Ng, A. Tiled convolutional neural networks. *Advances in neural information processing systems*, 23, 2010.

Nguyen, K. and Ho, N. Amortized projection optimization for sliced wasserstein generative models. *arXiv preprint arXiv:2203.13417*, 2022a.

Nguyen, K. and Ho, N. Revisiting sliced wasserstein on images: From vectorization to convolution. *arXiv preprint arXiv:2204.01188*, 2022b.

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

Nguyen, K., Nguyen, S., Ho, N., Pham, T., and Bui, H. Improving relational regularized autoencoders with spherical sliced fused gromov wasserstein. *arXiv preprint arXiv:2010.01787*, 2020b.

Nguyen, K., Ren, T., Nguyen, H., Rout, L., Nguyen, T., and Ho, N. Hierarchical sliced wasserstein distance. *arXiv preprint arXiv:2209.13570*, 2022.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

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.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

Santambrogio, F. {Euclidean, metric, and Wasserstein} gradient flows: an overview. *Bulletin of Mathematical Sciences*, 7(1):87–154, 2017.

Shi, J., Sun, S., and Zhu, J. A spectral approach to gradient estimation for implicit distributions. In *International Conference on Machine Learning*, pp. 4644–4653. PMLR, 2018.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

Villani, C. *Optimal Transport: Old and New*. Springer Berlin Heidelberg, 2008.

Wu, J., Huang, Z., Acharya, D., Li, W., Thoma, J., Paudel, D. P., and Gool, L. V. Sliced wasserstein generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3713–3722, 2019.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Yu, J., Xu, Y., Koh, J. Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B. K., et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

Zhou, Y., Shi, J., and Zhu, J. Nonparametric score estimators. In *International Conference on Machine Learning*, pp. 11513–11522. PMLR, 2020.

A. Illustrations of Joint SWFs, Conditional SWFs and CSWF

In this section, we show several 2-dimensional toy examples (i.e. $\mathcal{X} = \mathcal{Y} = \mathbb{R}$) to motivate our CSWF method.

The Joint SWF Suppose we have a target distribution $q(x, y) \in \mathcal{P}_2(\mathbb{R}^2)$, which is a mixture of two Gaussian distributions (outlined in blue shaded contours in Fig. 6a) and an initial distribution $p_0(x, y) \in \mathcal{P}_2(\mathbb{R}^2)$, which is a mixture of another two Gaussian distributions with different modes (outlined in red contours in Fig. 6a). The joint SWF $(p_t)_{t\geq 0}$ starting from p_0 and targeting q is demonstrated in Fig. 6a, chronologically from left to right. We observe that each mixture component of p_0 is "split" into two parts and moves to two different components of the target distribution q.

The Ideal Conditional SWFs In the setting of conditional modeling described in Sec. 4, we aim to fit $q(x|y)$ for all $y \in \mathcal{Y}$. Ideally, we can achieve this with a conditional SWF starting from $p_0(x|y)$ and targeting $q(x|y)$ for each y, as described in Sec. 4.1. Fig. 6b illustrates the effect of the ideal conditional SWFs $(p_{y,t}(x))_{t\geq 0}, \forall y \in \mathcal{Y}$.

A Difference Joint SWF We now alter the initial distribution $p_0(x, y)$ and the target $q(x, y)$ by simply shifting their mixture components farther apart in the y-direction (i.e., the vertical direction), and then show the new joint SWF in Fig. 6c. The mixture components now move (roughly) horizontally, resulting in a significantly different trajectory than in Fig. 6a.

Motivation of CSWF While the above example (Fig. 6c) remains a joint SWF, it bears considerable resemblance to the desired conditional SWFs (Fig. 6b). This motivates us to approximate conditional SWFs with a joint SWF of scaled initial and target distributions (Algorithm 1). Specifically, we first stretch the initial and target distributions along the $\mathcal{Y}$-component (using a factor ξ, which we call an amplifier), then simulate the joint SWF of the streched initial and target distributions, and finally compress the distributions to the original scale. We show the results of our CSWF in Fig. 6d.

Significance of ξ in CSWF Note that the effect of a large amplifier ξ is significant, since scaling the $\mathcal{Y}$-component is the key factor in making the joint SWF approximate the conditional SWFs. In Fig. 6e, we show the results of CSWF without amplifying (i.e., with $\xi = 1$) for comparison.

B. Recap of the SWF Algorithm (Liutkus et al., 2019)

Liutkus et al. (2019) consider minimizing the functional $\mathcal{F}^q_\lambda(\cdot) = \frac{1}{2}SW_2^2(\cdot, q) + \lambda\mathcal{H}(\cdot)$, where $\mathcal{H}(\cdot)$ denotes the negative entropy defined by $\mathcal{H}(p) \triangleq \int_{\mathbb{R}^d} p(x)\log p(x)\mathrm{d}x$. The introduced entropic regularization term helps to have the convergence of the Wasserstein gradient flow. In specific, they prove that under certain conditions the Wasserstein gradient flow of $\mathcal{F}^q_\lambda$ admits density $(p_t)_{t\geq 0}$ that satisfies the following continuity equation:

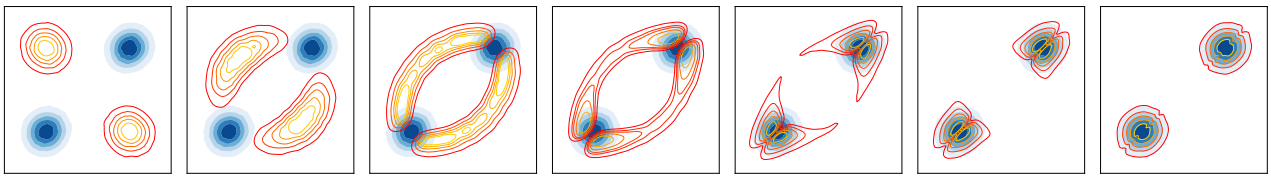
$$\frac{\partial p_t(x)}{\partial t} + \nabla \cdot (p_t(x)v_t(x)) - \lambda\Delta p_t = 0, \quad v_t(x) \triangleq -\int_{\mathbb{S}^{d-1}} \psi'_{t,\theta}(\theta^\top x)\cdot\theta \,\mathrm{d}\lambda(\theta).$$

Compared to Eq. (3), there is an extra Laplacian term which corresponds to the entropic regularization in $\mathcal{F}^q_\lambda$. By drawing a connection between this Fokker-Planck-type equation and stochastic differential equations (SDE), they propose to simulate the above equation with a stochastic process $\mathrm{d}X_t = v_t(X_t)\mathrm{d}t + \sqrt{2\lambda}\mathrm{d}W_t$, where $(W_t)_t$ denotes a standard Wiener process. Finally, they propose to approximate the SDE with a particle system and present a practical algorithm for unconditional generative modeling, which we recap in Algorithm 2 (using our notations).

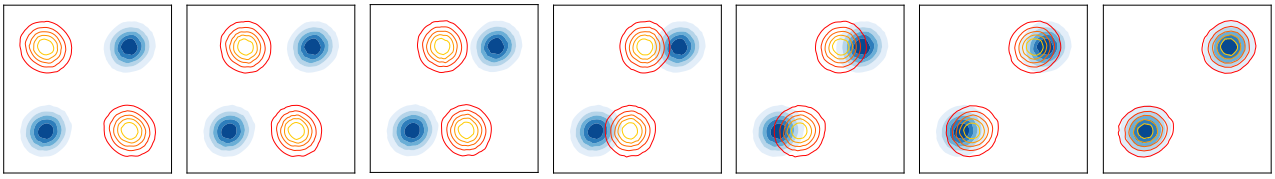
C. Additional Experimental Details

In our experiments, we augment the CIFAR-10 dataset with horizontally flipped images, resulting in a total of 100000 training images. This is analogous to the random flip data augmentation used in training neural networks. We do not employ this augmentation for the CelebA dataset due to limited computing resources. The pixel values of all images are dynamically dequantized at each step during the simulation and are rescaled to the range of $[-1, 1]$.

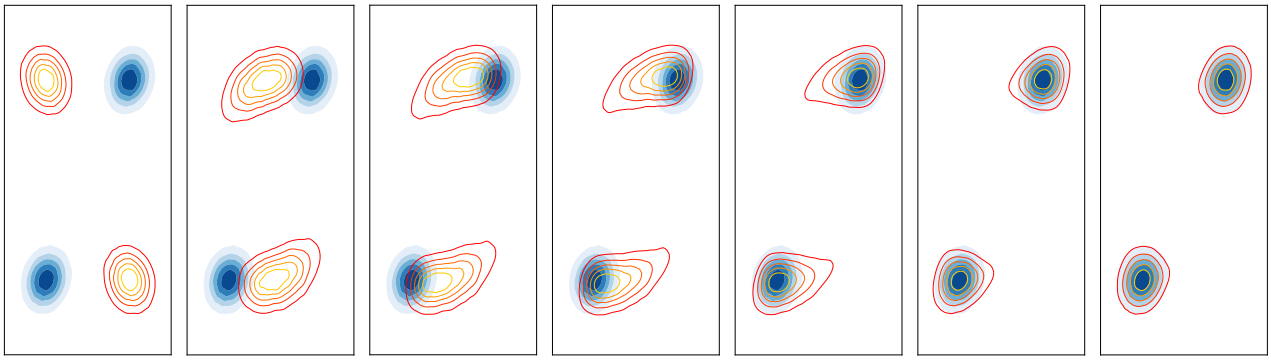
We set the simulation step size $\eta = d$ (i.e. the dimensionality of the images) due to the following reasons. In one-dimensional cases ($d = 1$), we can solve the optimal transport problem between $\theta^*_\sharp p_t$ and $\theta^*_\sharp q$ using Eq. (7) with step size $\eta = 1$, since it recovers the optimal transport map T. In d-dimensional cases, for any d orthogonal projections, the optimal transport problems between the projected distributions are independent of each other and can be solved simultaneously. In Eq. (7), however, the transport maps (i.e., the derivatives of the Kantorovich potentials) of all directions are averaged. Therefore, we set $\eta = d$ so that the step size in each direction equals to 1 in the average sense.



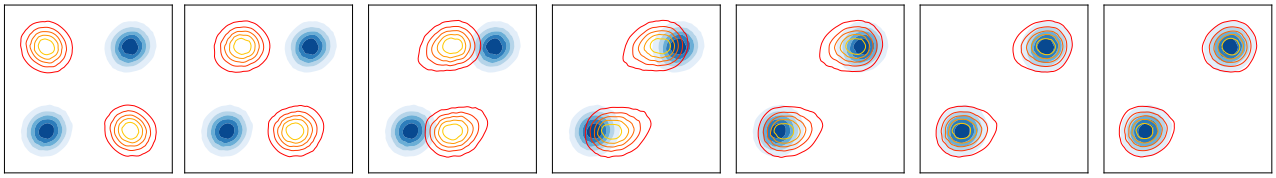
(a) The joint SWF.



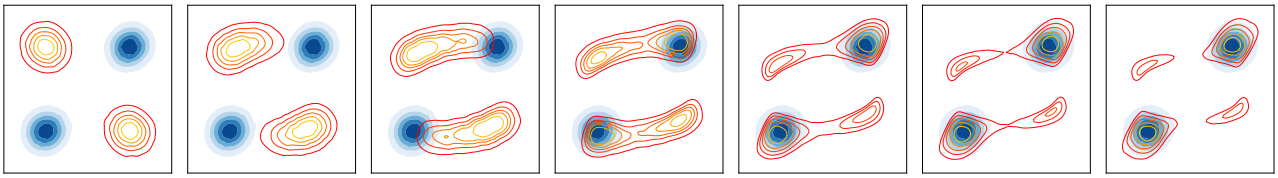
(b) The ideal conditional SWFs.



(c) The joint SWF after moving components farther apart in the y-direction.



(d) Illustration of the proposed CSWF.



(e) Illustration of the proposed CSWF without amplifying (i.e., with $\xi = 1$).

Figure 6. Illustrations of joint SWFs, conditional SWFs and the proposed CSWF algorithm. See more explanations in Appendix A.

Algorithm 2: Sliced-Wasserstein Flow (SWF) (Liutkus et al., 2019)

Input: $\mathcal{D} = \{x_i\}_{i=1}^N$, M, H, η, λ, K
Output: $\{\bar{X}_K^j\}_{j=1}^M$
// Initialize the particles
$\{\bar{X}_0^j\}_{j=1}^M \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d)$
// Generate random directions
$\{\theta_h\}_{h=1}^H \overset{\text{i.i.d.}}{\sim} \text{Uniform}(\mathbb{S}^{d-1})$
// Quantiles of projected target
for $h = 1, \dots, H$ **do**
  $F^{-1}_{\theta_{h\sharp}^*\hat{q}} = \text{QF}(\{\theta_h^\top x_i\}_{i=1}^N)$ // QF denotes the quantile function
// Iterations
for $k = 0, \dots, K-1$ **do**
  for $h = 1, \dots, H$ **do**
    // CDF of projected particles
    $F_{\theta_{h\sharp}^*\hat{p}_k} = \text{CDF}(\{\theta_h^\top \bar{X}_k^j\}_{j=1}^M)$
  for $j = 1, \dots, M$ **do**
    // Update the particles
    $\bar{X}_{k+1}^j = \bar{X}_k^j - \eta \cdot \hat{v}_k(\bar{X}_k^j) + \sqrt{2\lambda\eta} \cdot \epsilon_{k+1}^j, \qquad \epsilon_{k+1}^j \sim \mathcal{N}(0, I_d)$

C.1. Locally-Connected Projections and Pyramidal Schedules

We summarize the pyramidal schedules used for each dataset in Table 2. When upsampling projections with a lower resolution than the image, we empirically find that Lanczos upsampling works better than nearest neighbor upsampling.

C.2. CDF Estimations

We estimate the CDFs of the projected distributions $\theta_\sharp^* \hat{p}_t$ and $\theta_\sharp^* \hat{q}$ by first sorting their (scalar) projected values $\{\theta_x^\top \bar{X}^j + \xi \cdot \theta_y^\top \bar{Y}^j\}_{j=1}^M$ and $\{\theta_x^\top x_i + \xi \cdot \theta_y^\top y_i\}_{i=1}^N$, respectively. After sorting, the linear interpolation is performed as follows. Let $\{z_i\}_{i=1}^N$ denote the sorted array, i.e., $z_1 \leq \cdots \leq z_N$. When estimating the CDF of an input value z', we first find its insert position I (i.e., the index I satisfying $z_I \leq z' \leq z_{I+1}$) with binary search. Then the CDF of z' is estimated with $\frac{I-1}{N} + \frac{1}{N}\frac{z'-z_I}{z_{I+1}-z_I}$. For an input $a \in [0, 1]$, we inverse the CDF by first calculating the index $I = \lfloor a \times N \rfloor$ and then computing the inverse value as $z_I + (a \times N - I) * (z_{I+1} - z_I)$. For CIFAR-10 and CelebA, since we use a relatively large number of particles which leads to a slow sorting procedure, we choose a subset of particles for the estimation of $\theta_\sharp^* \hat{p}_t$.

D. Additional Experiments

D.1. Ablation Studies of H, M and ξ

We present the FID scores obtained by ℓ-SWF using different numbers of Monte Carlo samples H in Table 3. The results of ℓ-SWF using different numbers of particles M are shown in Table 4. We show the class-conditional generation of ℓ-CSWF using different amplifiers ξ from 0 to 10 in Fig. 7.

D.2. Additional Samples

More unconditional samples from ℓ-SWF are shown in Fig. 8. We show the nearest neighbors of the generated samples in Fig. 9, where we observe that the generated samples are not replicated training samples or combined training patches, but generalize at the semantic level. In Fig. 10, we show the offline samples, which appear to be comparable in visual quality to the batch samples (in Fig. 3). Quantitatively, the FID score of the offline samples on CIFAR-10 is 61.1, which is also close to that of the batched samples (59.7).

Fig. 11 and Fig. 12 show additional class-conditional samples and image inpainting results of ℓ-CSWF, respectively.

Table 2. Details of the pyramidal schedules for each dataset. In each entry, $(\mathtt{H} \times \mathtt{W})$ $[\mathtt{S}_1, \ldots, \mathtt{S}_k]$ denotes that we use locally-connected projections of resolution $\mathtt{H} \times \mathtt{W}$ with patch size $\mathtt{S}_1 \times \mathtt{S}_1, \ldots, \mathtt{S}_k \times \mathtt{S}_k$ sequentially. We upsample all projections to image resolution.

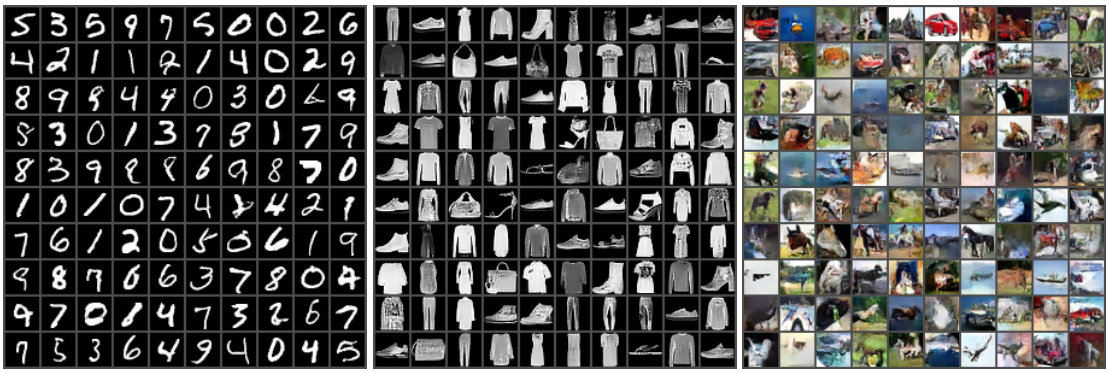
MNIST & Fashion MNIST	CIFAR-10	CelebA
(1×1) $[1]$	(1×1) $[1]$	(1×1) $[1]$
(2×2) $[2]$	(2×2) $[2]$	(2×2) $[2]$
(3×3) $[3]$	(3×3) $[3]$	(3×3) $[3]$
(4×4) $[4]$	(4×4) $[4]$	(4×4) $[4]$
(5×5) $[5]$	(5×5) $[5]$	(5×5) $[5]$
(6×6) $[6]$	(6×6) $[6]$	(6×6) $[6]$
(7×7) $[7, 5, 3]$	(7×7) $[7]$	(7×7) $[7]$
(11×11) $[11, 9, 7, 5, 3]$	(8×8) $[8, 7, 5, 3]$	(8×8) $[8, 7, 5, 3]$
(14×14) $[14, 13, 11, 9, 7, 5, 3]$	(12×12) $[12, 11, 9, 7, 5, 3]$	(12×12) $[12, 11, 9, 7, 5, 3]$
(21×21) $[15, 13, 11, 9, 7, 5, 3]$	(16×16) $[15, 13, 11, 9, 7, 5, 3]$	(16×16) $[15, 13, 11, 9, 7, 5, 3]$
(28×28) $[15, 13, 11, 9, 7, 5, 3]$	(24×24) $[15, 13, 11, 9, 7, 5, 3]$	(24×24) $[15, 13, 11, 9, 7, 5, 3]$
	(32×32) $[15, 13, 11, 9, 7, 5, 3]$	(32×32) $[15, 13, 11, 9, 7, 5, 3]$
		(64×64) $[15, 13, 11, 9, 7, 5, 3]$

Table 3. FID↓ scores of ℓ-SWF using different number of Monte Carlo samples H on CIFAR-10.

# Monte Carlo Samples	FID
$H = 1000$	90.8
$H = 5000$	68.1
$H = 10000$	59.7

Table 4. FID↓ scores of ℓ-SWF using different number of particles M on CIFAR-10.

# Particles	FID
$M = 1 \times 10^5$	64.0
$M = 3 \times 10^5$	61.4
$M = 7 \times 10^5$	59.7



(a) MNIST, $\xi = 0$ (b) Fashion MNIST, $\xi = 0$ (c) CIFAR-10, $\xi = 0$

(d) MNIST, $\xi = 2$ (e) Fashion MNIST, $\xi = 2$ (f) CIFAR-10, $\xi = 2$

(g) MNIST, $\xi = 5$ (h) Fashion MNIST, $\xi = 5$ (i) CIFAR-10, $\xi = 5$

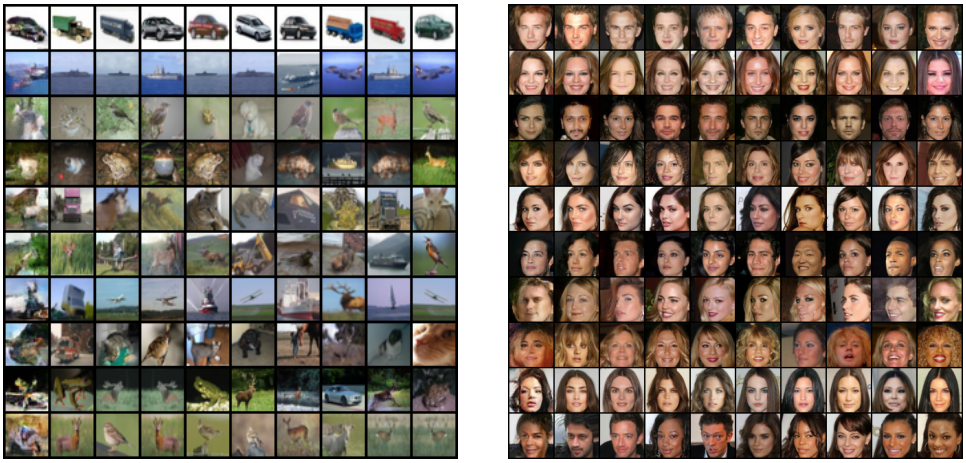
(j) MNIST, $\xi = 10$ (k) Fashion MNIST, $\xi = 10$ (l) CIFAR-10, $\xi = 10$

Figure 7. Class-conditional generation of ℓ-CSWF with different amplifiers ξ on MNIST, Fashion MNIST and CIFAR-10. In each figure, each row corresponds to a class. We observe that $\xi = 0$ recovers unconditional generation and the generated samples become more consistent with the classes as ξ grows.

(a) MNIST (b) Fashion MNIST

(c) CIFAR-10 (d) CelebA

Figure 8. Additional uncurated batched samples from ℓ-SWF on MNIST, Fashion MNIST, CIFAR-10 and CelebA.



(a) CIFAR-10 (b) CelebA

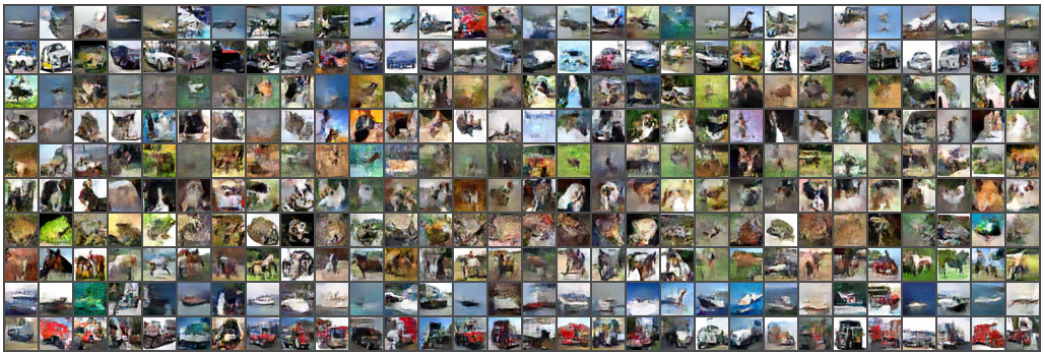
Figure 9. L_2 nearest neighbors of the generated samples from ℓ-SWF on CIFAR-10 and CelebA. The leftmost columns are the generated images. Images to the right are the nearest neighbors in the dataset.

(a) MNIST (b) Fashion MNIST (c) CIFAR-10 (d) CelebA

Figure 10. Uncurated offline samples from ℓ-SWF on MNIST, Fashion MNIST, CIFAR-10 and CelebA.

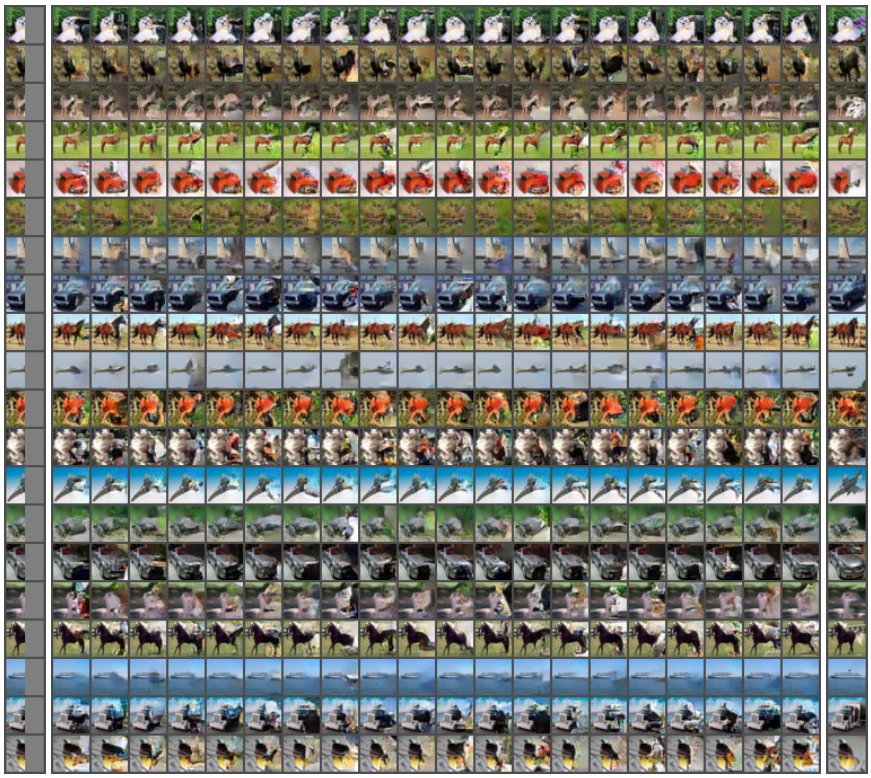
(a) MNIST

(b) Fashion MNIST

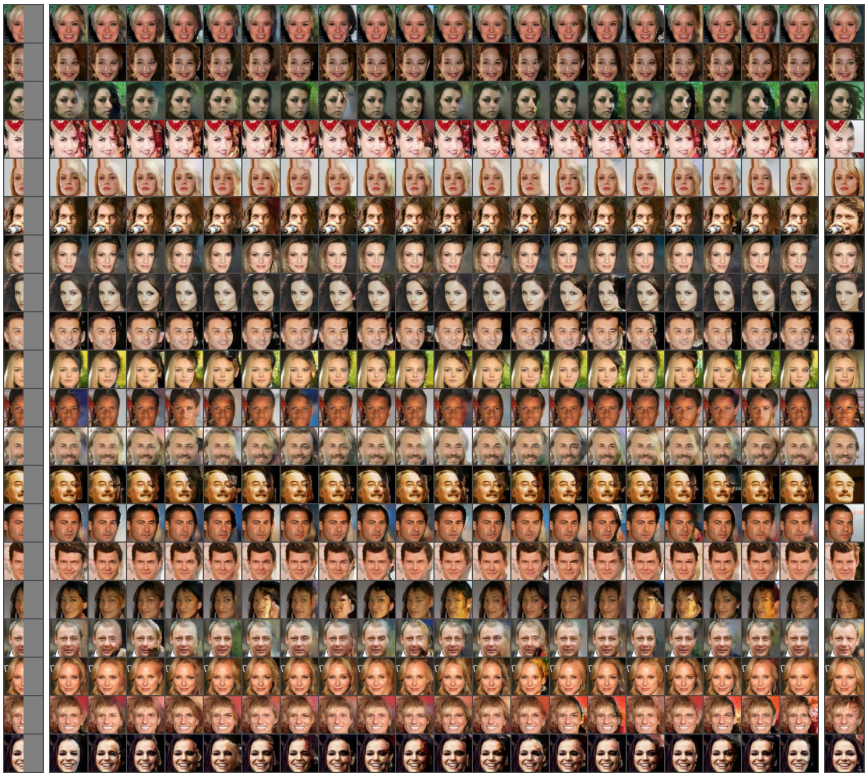


(c) CIFAR-10

Figure 11. Additional class-conditional samples from ℓ-CSWF ($\xi = 10$) on MNIST, Fashion MNIST and CIFAR-10.



(a) CIFAR-10



(b) CelebA

Figure 12. Additional image inpainting results of ℓ-CSWF ($\xi = 1$) on CIFAR-10 and CelebA.