Continuous-Time Flows for Efficient Inference and Density Estimation

Changyou Chen [1] **Chunyuan Li** [2] **Liqun Chen** [2] **Wenlin Wang** [2] **Yunchen Pu** [2] **Lawrence Carin** [2]

Abstract

Two fundamental problems in unsupervised learning are efficient inference for latent-variable models and robust density estimation based on large amounts of unlabeled data. Algorithms for the two tasks, such as normalizing flows and generative adversarial networks (GANs), are often developed independently. In this paper, we propose the concept of *continuous-time flows* (CTFs), a family of diffusion-based methods that are able to asymptotically approach a target distribution. Distinct from normalizing flows and GANs, CTFs can be adopted to achieve the above two goals in one framework, with theoretical guarantees. Our framework includes distilling knowledge from a CTF for efficient inference, and learning an explicit energy-based distribution with CTFs for density estimation. Both tasks rely on a new technique for distribution matching within amortized learning. Experiments on various tasks demonstrate promising performance of the proposed CTF framework, compared to related techniques.

1. Introduction

Efficient inference and robust density estimation are two important goals in unsupervised learning. In fact, they can be unified from the perspective of learning desired target distributions. In inference problems, one targets to learn a tractable distribution for a *latent variable* that is close to a given unnormalized distribution (*e.g.*, a posterior distribution in a Bayesian model). In density estimation, one tries to learn an unknown *data distribution* only based on the samples from it. It is also helpful to make a distinction between two types of representations for learning distributions: explicit and implicit methods (Mohamed & Lakshminarayanan, 2017). Explicit methods provide a prescribed parametric form for the distribution, while implicit methods learn a stochastic procedure to directly generate samples from the unknown distribution.

Existing deep generative models can easily be identified from this taxonomy. For example, the standard variational autoencoder (VAE) (Kingma & Welling, 2014; Rezende et al., 2014) is an important example of an *explicit inference* method. Within the inference arm (encoder) of a VAE, recent research has focused on improving the accuracy of the approximation to the posterior distribution on latent variables (codes) using normalizing flow (NF) (Rezende & Mohamed, 2015). NF is particularly interesting due to its ability to approximate the posterior distribution arbitrarily well, while maintaining explicit parametric forms. On the other hand, Stein VAE (Feng et al., 2017; Pu et al., 2017) is an *implicit inference* method, as it only learns to draw samples to approximate posteriors, without assuming an explicit form for the distribution.. For density estimation on observed data, the generative adversarial network (GAN) can be regarded as an *implicit density estimation* method (Ranganath et al., 2016; Huszár, 2017; Mohamed & Lakshminarayanan, 2017), in the sense that one may sample from the distribution (regarded as a representation of the unknown distribution), but an explicit form for the distribution is not estimated. GAN has recently been augmented by Flow-GAN (Grover et al., 2017) to incorporate a likelihood term for *explicit density estimation*. Further, the real-valued non-volume preserving (real NVP) transformations algorithm (Dinh et al., 2017) was proposed to perform inference within the implicit density estimation framework.

Some aforementioned methods rely on the concept of *flows*. A flow defines a series of transformations for a random variable (RV), such that the distribution of the RV evolves from a simple distribution to a more complex distribution. When the sequence of transformations are indexed on a discrete-time domain (*e.g.*, indexed with integers) with a finite number of transformations, this method is referred to as a normalizing flow (Rezende & Mohamed, 2015). Various efficient implementations of NFs have been proposed, such as the planar, radial (Rezende & Mohamed, 2015), Householder (Tomczak & Welling, 2016), and inverse autoregressive flows (Kingma et al., 2016). One theoretical limitation of existing normalizing flows is that there is no guarantee on the approximation accuracy due to the finite number of transformations.

[1]SUNY at Buffalo [2]Duke University. Correspondence to: Changyou Chen <cchangyou@gmail.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

By contrast, little work has explored the applicability of continuous-time flows (CTFs) in deep generative models, where a sequence of transformations are indexed on a continuous-time domain (*e.g.*, indexed with real numbers). There are at least two reasons encouraging research in this direction: *i*) CTFs are more general than traditional normalizing flows in terms of modeling flexibility, due to the intrinsic infinite number of transformations; *ii*) CTFs are more theoretically grounded, in the sense that they are guaranteed to approach a target distribution asymptotically (details provided in Section 2.2).

In this paper, we propose efficient ways to apply CTFs for the two motivating tasks. Based on the CTF, our framework learns to drawn samples directly from desired distributions (*e.g.*, the unknown posterior and data distributions) for both inference and density estimation tasks via amortization. In addition, we are able to learn an explicit form of the unknown data distribution for density estimation*. The core idea of our framework is the amortized learning, where knowledge in a CTF is distilled sequentially into another neural network (called *inference network* in the inference task, and *generator* in density estimation). The distillation relies on the distribution matching technique proposed recently via adversarial lerning (Li et al., 2017a). We conduct various experiments on both synthetic ad real datasets, demonstrating excellent performance of the proposed framework, relative to representative approaches.

2. Preliminaries

2.1. Efficient inference and density estimation

Efficient inference with normalizing flows Consider a probabilistic generative model with observation $\mathbf{x} \in \mathbb{R}^D$ and latent variable $\mathbf{z} \in \mathbb{R}^L$ such that $\mathbf{x} \,|\, \mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{z})$ with $\mathbf{z} \sim p(\mathbf{z})$. For efficient inference of $\mathbf{z}$, the VAE (Kingma & Welling, 2014) introduces the concept of an inference network (recognition model or encoder), $q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})$, as a variational distribution in the VB framework. An inference network is typically a stochastic (nonlinear) mapping from the input $\mathbf{x}$ to the latent $\mathbf{z}$, with associated parameters $\boldsymbol{\phi}$. For example, one of the simplest inference networks is defined as $q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x}), \text{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\mathbf{x})))$, where the mean function $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{x})$ and the standard-derivation function $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\mathbf{x})$ are specified via deep neural networks parameterized by $\boldsymbol{\phi}$. Parameters are learned by minimizing the negative evidence lower bound (ELBO), *i.e.*, the KL divergence between $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ and $q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})$: $\text{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right) \triangleq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})}\left[\log q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right]$, via stochastic gradient descent (Bottou, 2012).

One limitation of the VAE framework is that $q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})$ is often restricted to simple distributions for feasibility, *e.g.*, the normal distribution discussed above, and thus the gap between $q_{\boldsymbol{\phi}}(\mathbf{z} \,|\, \mathbf{x})$ and $p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, \mathbf{x})$ is typically large for complicated posterior distributions. NF is a recently proposed VB-based technique designed to mitigate this problem (Rezende & Mohamed, 2015). The idea is to augment $\mathbf{z}$ via a sequence of deterministic invertible transformations $\{\mathcal{T}_k : \mathbb{R}^L \rightarrow \mathbb{R}^L\}_{k=1}^K$, such that: $\mathbf{z}_0 \sim q_{\boldsymbol{\phi}}(\cdot \,|\, \mathbf{x}), \mathbf{z}_1 = \mathcal{T}_1(\mathbf{z}_0), \cdots, \mathbf{z}_K = \mathcal{T}_K(\mathbf{z}_{K-1})$.

Note the transformations $\{\mathcal{T}_k\}$ are typically endowed with different parameters, and we absorb them into $\boldsymbol{\phi}$. Because the transformations are deterministic, the distribution of $\mathbf{z}_K$ can be written as $q(\mathbf{z}_K) = q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x}) \prod_{k=1}^K \left|\det \frac{\partial \mathcal{T}_k}{\partial \mathbf{z}_k}\right|^{-1}$ via the change of variable formula. As a result, the negative ELBO for normalizing flows becomes:

$$\text{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}_K \,|\, \mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x})}\left[\log q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x})\right] \quad (1)$$

$$- \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}_K)\right] - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x})}[\sum_{k=1}^K \log \left|\det \frac{\partial \mathcal{T}_k}{\partial \mathbf{z}_k}\right|].$$

Typically, transformations $\mathcal{T}_k$ of a simple parametric form are employed to make the computations tractable (Rezende & Mohamed, 2015). Our method generalizes these discrete-time transformations to continuous-time ones, ensuring convergence of the transformations to a target distribution.

Related density-estimation methods There exist implicit and explicit density-estimation methods. Implicit density models such as GAN provide a flexible way to draw samples directly from unknown data distributions (via a deep neural network (DNN) called a generator with stochastic inputs) without explicitly modeling their density forms; whereas explicit models such as the pixel RNN/CNN (van den Oord et al., 2016) define and learn explicit forms of the unknown data distributions. This gives the advantage that the likelihood for a test data point can be explicitly evaluated. However, the generation of samples is typically time-consuming due to the sequential generation nature.

Similar to Wang & Liu (2017), our CTF-based approach in Section 4 provides an alternative way for this problem, by simultaneously learning an explicit energy-based data distribution (estimated density) and a generator whose generated samples match the learned data distribution. This not only gives us the advantage of explicit density modeling but also provides an efficient way to generate samples. Note that our technique differs from that of Wang & Liu (2017) in that distribution matching is adopted to learn an accurate generator, which is a key component in our framework.

2.2. Continuous-time flows

We notice two potential limitations with traditional normalizing flows: *i*) given specified transformations $\{\mathcal{T}_k\}$, there is no guarantee that the distribution of $\mathbf{z}_K$ could exactly match $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$; *ii*) the randomness is only introduced in $\mathbf{z}_0$ (from the inference network), limiting the representation

*Although the density is represented as an energy-based distribution with an intractable normalizer.

power. We specify CTFs where transformations are indexed by real numbers, thus they could be considered as consisting of infinite transformations. Further, we consider stochastic flows where randomness is injected in a continuous-time manner. In fact, the concept of CTFs (such as the Hamiltonian flow) has been introduced by Rezende & Mohamed (2015), without further development on efficient inference.

We consider a flow on $\mathbb{R}^L$, defined as the mapping[†] $\mathcal{T} : \mathbb{R}^L \times \mathbb{R} \to \mathbb{R}^L$ such that[‡] we have $\mathcal{T}(\mathbf{Z}, 0) = \mathbf{z}$ and $\mathcal{T}(\mathcal{T}(\mathbf{Z}, t), s) = \mathcal{T}(\mathbf{Z}, s + t)$, for all $\mathbf{Z} \in \mathbb{R}^L$ and $s, t \in \mathbb{R}$. A specific form consider here is defined as $\mathcal{T}(\mathbf{Z}, t) = \mathbf{Z}_t$, where $\mathbf{Z}_t$ is driven by a diffusion of the form:

$$\mathrm{d}\,\mathbf{Z}_t = F(\mathbf{Z}_t)\mathrm{d}t + V(\mathbf{Z}_t)\mathrm{d}\mathcal{W}\,. \tag{2}$$

Here $F : \mathbb{R}^L \to \mathbb{R}^L$, $V : \mathbb{R}^{L\times L} \to \mathbb{R}^L$ are called the drift term and diffusion term, respectively; $\mathcal{W}$ is the standard L-dimensional Brownian motion. In the context of inference, we seek to make the stationary distribution of $\mathbf{Z}_t$ approach $p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, \mathbf{x})$. One solution for this is to set $F(\mathbf{Z}_t) = \frac{1}{2}\nabla_{\mathbf{z}} \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z} = \mathbf{Z}_t)$ and $V(\mathbf{Z}_t) = \mathbf{I}_L$ with $\mathbf{I}_L$ the $L \times L$ identity matrix. The resulting diffusion is called Langevin dynamics (Welling & Teh, 2011). Denoting the distribution of $\mathbf{Z}_t$ as ρ_t, it is well known (Risken, 1989) that ρ_t is characterized by the Fokker-Planck (FP) equation:

$$\partial_t \rho_t = -\nabla_{\mathbf{z}} \cdot (\rho_t F(\mathbf{Z}_t) + \nabla_{\mathbf{z}} \cdot (\rho_t V(\mathbf{Z}_t) V^\top(\mathbf{Z}_t)))\,, \tag{3}$$

where $\mathbf{a} \cdot \mathbf{b} \triangleq \mathbf{a}^\top \mathbf{b}$ for vectors $\mathbf{a}$ and $\mathbf{b}$.

For simplicity, we consider the flow defined by the Langevin dynamics specified above, though our results generalize to other stochastic flows (Dorogovtsev & Nishchenko, 2014). In fact, CTF has been applied for scalable Bayesian sampling (Ding et al., 2014; Li et al., 2016a; Chen et al., 2016; Li et al., 2016b; Zhang et al., 2017). In this paper, we generalize it by specifying an ELBO under a CTF, which can then be readily solved by a discretized numerical scheme, based on the results from Jordan et al. (1998). An approximation error bound for the scheme is also derived. We defer proofs of our theoretical results to the Supplementary Material (SM) for conciseness.

3. Continuous-Time Flows for Inference

For this task, we adopt the VAE/normalizing-flow framework with an encoder-decoder structure. An important difference is that instead of feeding data to an encoder and sampling a latent representation in the output as in VAE, we concatenate the data with independent noise as input and directly generate output samples[§], constituting an *implicit* model. These outputs are then driven by the CTF to approach the true posterior distribution. In the following, we first show that directly optimizing the ELBO is infeasible. We then propose an amortized-learning process that sequentially distills the implicit transformations from the CTF an inference network by distribution matching in Section 3.2.

3.1. The ELBO and discretized approximation

We first incorporate CTF into the NF framework by writing out the corresponding ELBO. Note that there are two steps in the inference process. First, an initial $\mathbf{z}_0$ is drawn from the inference network $q_{\boldsymbol{\phi}}(\cdot|\,\mathbf{x})$; second, $\mathbf{z}_0$ is evolved via a diffusion such as (2) for time T (via the transformation $\mathbf{Z}_T = \mathcal{T}(\mathbf{z}_0, T)$). Consequently, the negative ELBO for CTF can be written as

$$\begin{aligned}\mathcal{F}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0\,|\,\mathbf{x})}\mathbb{E}_{\rho_T}\left[\log \rho_T - \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{Z}_T)\right.\\ \left. + \log\left|\det \frac{\partial\, \mathbf{Z}_T}{\partial\, \mathbf{z}_0}\right|\right] \triangleq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_0\,|\,\mathbf{x})}\left[\mathcal{F}_1(\mathbf{x}, \mathbf{z}_0)\right]\,.\end{aligned} \tag{4}$$

Note the term $\mathcal{F}_1(\mathbf{x}, \mathbf{z}_0)$ is intractable to calculate, in that *i*) ρ_T does not have an explicit form; *ii*) the Jacobian $\frac{\partial\, \mathbf{Z}_T}{\partial\, \mathbf{z}_0}$ is generally infeasible. In the following, we propose an approximate solution for problem *i*). Learning by avoiding problem *ii*) is presented in Section 3.2 via amortization.

For problem *i*), a reformulation of the results from Jordan et al. (1998) leads to a nice way to approximate ρ_t in Lemma 1. Note in practice we adopt an *implicit* method which uses samples to approximate the solution in Lemma 1 for feasibility, detailed in (6).

Lemma 1 *Assume that* $\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) \le C_1$ *is infinitely differentiable, and* $\|\nabla_{\mathbf{z}} \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\| \le C_2\,(1 + C_1 - \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}))\,(\forall\, \mathbf{x}, \mathbf{z})$ *for some constants* $\{C_1, C_2\}$. *Let* $T = hK$ *(*h *is the stepsize and* K *is the number of transformations),* $\rho_0 \triangleq q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x})$, *and* $\{\tilde{\rho}_k\}_{k=1}^K$ *be the solution of the functional optimization problem:*

$$\tilde{\rho}_k = \arg\min_{\rho \in \mathcal{K}} KL\left(\rho \| p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right) + \frac{1}{2h} W_2^2\left(\tilde{\rho}_{k-1}, \rho\right)\,, \tag{5}$$

with $W_2^2\left(\mu_1, \mu_2\right) \triangleq \inf_{p \in \mathcal{P}(\mu_1, \mu_2)} \int \|\mathbf{x} - \mathbf{y}\|_2^2\, p(\mathrm{d}\,\mathbf{x}, \mathrm{d}\,\mathbf{y})$ *the square of 2nd-order Wasserstein distance, and* $\mathcal{P}(\mu_1, \mu_2)$ *the set of joint distributions on* $\{\mu_1, \mu_2\}$. $\mathcal{K}$ *is the space of distributions with finite 2nd-order moment. Then* $\tilde{\rho}_K$ *converges to* ρ_T *in the limit of* $h \to 0$, *i.e.,* $\lim_{h\to 0} \tilde{\rho}_K = \rho_T$, *where* ρ_T *is the solution of the FP equation* (3) *at time* T.

Lemma 1 reveals an interesting way to compute ρ_T via a sequence of functional optimization problems. By comparing it with the objective of the traditional NF, which minimizes the KL-divergence between ρ_K and $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$, at each sub-optimization-problem in Lemma 1, it minimizes the KL-divergence between $\tilde{\rho}_k$ and $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$, plus a regularization term as the Wasserstein distance between $\tilde{\rho}_{k-1}$

[†] We reuse the notation $\mathcal{T}$ as transformations from the discrete case above for simplicity, and use $\mathbf{Z}$ instead of $\mathbf{z}$ (reserved for the discrete-time setting) to denote the random variable in the continuous-time setting.

[‡] Note we define continuous-time flows in terms of latent variable $\mathbf{Z}$ in order to incorporate it into the setting of inference. However, the same description applies when we define the flow in data space, which is the setting of density estimation in Section 4.

[§] Such structure can represent much more complex distributions than a parametric form, useful for the follow up procedures. In contrast, we will define an explicit energy-based distribution for the density in density-estimation tasks.

and $\tilde{\rho}_k$. The extra Wasserstein-distance term arises naturally due to the fact that the Langevin diffusion can be explained as a gradient flow whose geometry is equipped with the Wasserstein distance (Otto, 1998).

The optimization problem in Lemma 1 is, however, difficult to deal with directly. In practice, we instead approximate the discretization in an equivalent way by simulation from the CTF. Starting from $\mathbf{z}_0$, each $\mathbf{z}_k$ $(k = 0, \cdots, K-1)$ is fed into a transformation $\mathcal{T}_k$ (specified below), resulting in $\mathbf{z}_{k+1}$ whose distribution coincides with $\tilde{\rho}_{k+1}$ in Lemma 1. The discretization procedure is illustrated in Figure 1. We must specify the transformations $\mathcal{T}_k$. For each k, let $t = hk$; we can conclude from Lemma 1 that $\lim_{h\to 0} \tilde{\rho}_k = \rho_t$. From FP theory, ρ_t is obtained by solving the diffusion (2) with initial condition $\mathbf{Z}_0 = \mathbf{z}_0$. It is thus reasonable to specify the transformation $\mathcal{T}_k$ as the k-th step of a numerical integrator for (2). Specifically, we specify $\mathcal{T}_k$ stochastically:

$$\mathbf{z}_k = \mathcal{T}_k(\mathbf{z}_{k-1}) \triangleq \mathbf{z}_{k-1} + F(\mathbf{z}_{k-1})h + V(\mathbf{z}_{k-1})\boldsymbol{\zeta}_k \ , \quad (6)$$

where $\boldsymbol{\zeta}_k \sim \mathcal{N}(\mathbf{0}, h\mathbf{I}_L)$ is drawn from an isotropic normal. Note the transformation defined here is stochastic, thus we only get samples from $\tilde{\rho}_K$ at the end. A natural way to approximate $\tilde{\rho}_K$ is to use the empirical sample distribution, *i.e.*, $\tilde{\rho}_K \approx \frac{1}{K}\sum_{k=1}^{K} \delta_{\mathbf{z}_k} \triangleq \bar{\rho}_T$ with $\delta_{\mathbf{z}}$ a point mass at $\mathbf{z}$. Afterwards, $\tilde{\rho}_K$ (thus $\bar{\rho}_T$) will be used to approximate the true ρ_T from (3).

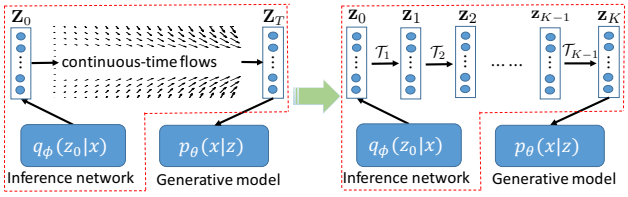

Figure 1. Discretized approximation (right) of a continuous-time flow (left). Densities $\{\tilde{\rho}_k\}$ of $\{\mathbf{z}_k\}$ evolve via transformations $\{\mathcal{T}_k\}$, with $\tilde{\rho}_k \to \rho_{hk}$ when $h \to 0$ for each k due to Lemma 1.

Better ways to approximate ρ_T might be possible, *e.g.*, by assigning more weights to the more recent samples. However, the problem becomes more challenging in theoretical analysis, an interesting point left for future work. In the following, we study how well $\bar{\rho}_T$ approximates ρ_T. Following literature on numerical approximation for Itô diffusions (Vollmer et al., 2016; Chen et al., 2015), we consider a 1-Lipschitz test function $\psi : \mathbb{R}^L \to \mathbb{R}$, and use the mean square error (MSE) bound to measure the closeness of $\bar{\rho}_T$ and ρ_T, defined as: $\text{MSE}(\bar{\rho}_T, \rho_T; \psi) \triangleq \mathbb{E}\left(\int \psi(\mathbf{z})(\tilde{\rho}_T - \rho_T)(\mathbf{z})\mathrm{d}\,\mathbf{z}\right)^2$, where the expectation is taken over all the randomness in the construction of $\tilde{\rho}_T$. Note that our goal is related but different from the standard setup as in (Vollmer et al., 2016; Chen et al., 2015), which studies the closeness of $\bar{\rho}_T$ to $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$. We need to adopt the assumptions from Vollmer et al. (2016); Chen et al. (2015), which are described in the SM. The assumptions are somewhat involved but essentially require coefficients of the diffusion (2) to be well-behaved. We derive the following bound for the MSE of the sampled approximation, $\bar{\rho}_T$, and the true distribution.

Theorem 2 *Under Assumption 1 in the SM, assume that* $\int \rho_T(\mathbf{z}) p_{\boldsymbol{\theta}}^{-1}(\mathbf{x}, \mathbf{z})\mathrm{d}\,\mathbf{z} < \infty$ *and there exists a constant* C *such that* $\frac{\mathrm{d}W_2^2(\rho_T, p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}))}{\mathrm{d}t} \geq CW_2^2\left(\rho_T, p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right)$*, the MSE is bounded as*

$$\textit{MSE}(\bar{\rho}_T, \rho_T; \psi) = O\left(\frac{1}{hK} + h^2 + e^{-2ChK}\right) .$$

The last assumption in Theorem 2 requires ρ_T to evolve fast through the FP equation, which is a standard assumption used to establish convergence to equilibrium for FP equations (Bolley et al., 2012). The MSE bound consists of three terms, the first two terms come from numerical approximation of the continuous-time diffusion, whereas the third term comes from the convergence bound of the FP equation in terms of the Wasserstein distance (Bolley et al., 2012). When the time $T = hK$ is large enough, the third term may be ignored due to its exponential-decay rate. Moreover, in the infinite-time limit, the bound endows a bias proportional to h; this, however, can be removed by adopting a decreasing-step-size scheme in the numerical method, as in standard stochastic gradient MCMC methods (Teh et al., 2016; Chen et al., 2015).

Remark 3 *To examine the optimal bound in Theorem 2, we drop out the term* e^{-2ChK} *in the long-time case (when* hK *is large enough) for simplicity because it is in a much lower order term than the other terms. The optimal MSE bound (over* h*) decreases at a rate of* $O\left(K^{-2/3}\right)$*, meaning that* $O\left(\epsilon^{-3/2}\right)$ *steps of transformations in Figure 1 (right) are needed to reach an* ϵ*-accurate approximation, i.e., MSE* $\leq \epsilon$*. This is computationally expensive. An efficient way for inference is thus imperative, developed in the next section.*

3.2. Efficient inference via amortization

Even though we approximate ρ_T with $\bar{\rho}_T$, it is still *infeasible* to directly apply it to the ELBO in (4) as $\bar{\rho}_T$ is discrete. To deal with this problem, we adopt the idea of "amortized learning" (Gershman & Goodman, 2014) for inference, by alternatively optimizing the two sets of parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$.

Updating $\boldsymbol{\phi}$ To explain the idea, first note that the negative ELBO can be equivalently written as

$$\mathcal{F}(\mathbf{x}) = \mathbb{E}_{\rho_0 \triangleq q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x})} \mathbb{E}_{\rho_T}\left[\log \rho_0 - \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{Z}_T)\right] \ . \quad (7)$$

When $\rho_0 = \rho_T$, it is easy to see that: $\mathcal{F}(\mathbf{x}) = \mathbb{E}_{\rho_0}\left[\log \rho_0 - \log p_{\boldsymbol{\theta}}(\mathbf{Z}_T \,|\, \mathbf{x})\right] + \log p(\mathbf{x}) = \log p(\mathbf{x})$, which essentially makes the gap between $q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x})$ and $p_{\boldsymbol{\theta}}(\mathbf{Z}_T \,|\, \mathbf{x})$ vanished. As a result, our goal is to learn $\boldsymbol{\phi}$ such that $q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x})$ approaches $p_{\boldsymbol{\theta}}(\mathbf{Z}_T \,|\, \mathbf{x})$. This is a distribution matching problem (Li et al., 2017a). As mentioned previously, we will learn an implicit distribution of $q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x})$

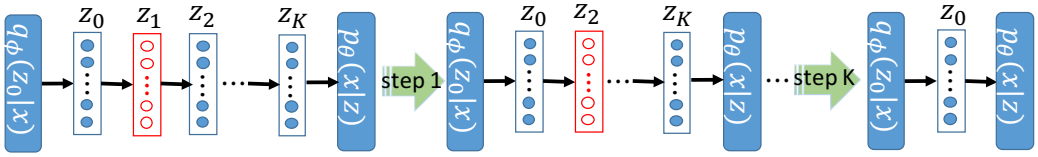

Figure 2. Amortized learning of CTFs. From left to right: the initial architecture with K-step transformations; For each step k, $q_{\boldsymbol{\phi}}(\cdot)$ is trained to match the distributin of $\mathbf{z}_k$ in CTFs; In the end, the CTF is distilled into $q_{\boldsymbol{\phi}}(\cdot)$ with distribution matching as in (8).

(*i.e.*, learn how to draw samples from $q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x})$ instead of its explicit form), as it allows us to chose a candidate distribution from a much larger distribution space, compared to explicitly defining $q_{\boldsymbol{\phi}}$[¶]. Consequently, $q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x})$ is implemented by a stochastic generator (a DNN parameterized by $\boldsymbol{\phi}$) $Q_{\boldsymbol{\phi}}(\mathbf{x}, \omega)$ with input as the concatenation of $\mathbf{x}$ and ω, where ω is a sample from an isotropic Gaussian distribution $q_0(\omega)$. Our goal is now translated to update the parameter $\boldsymbol{\phi}$ of $Q_{\boldsymbol{\phi}}(\mathbf{x}, \omega)$ to $\boldsymbol{\phi}'$ such that the distribution of $\{\mathbf{z}'_0 = Q_{\boldsymbol{\phi}'}(\mathbf{x}, \omega)\}$ with $\omega \sim q_0(\omega)$ matches that of $\mathbf{z}_1$ in the original generating process with $\boldsymbol{\phi}$ in Figure 1. In this way, the generating process of $\mathbf{z}_1$ via $\mathcal{T}_1$ is *distilled* into the parameterized generator $Q_{\boldsymbol{\phi}}(\cdot)$, eliminating the need to do a specific transformation via $\mathcal{T}_1$ in testing, and thus is very efficient. Specifically, we update $\boldsymbol{\phi}'$ such that

$$\boldsymbol{\phi}' = \arg\min_{\boldsymbol{\phi}} \mathcal{D}(\{\mathbf{z}_0'^{(i)}\}, \{\mathbf{z}_1^{(i)}\}) \,, \tag{8}$$

where $\{\mathbf{z}_0'^{(i)}\}_{i=1}^S$ are a set of samples generated from $q_{\boldsymbol{\phi}'}(\mathbf{z}'_0 \,|\, \mathbf{x})$ via $Q_\phi(\cdot)$, and $\{\mathbf{z}_1^{(i)}\}_{i=1}^S$ are samples drawn by $\omega^i \sim q_0(\omega), \tilde{\mathbf{z}}_0^i = Q_{\boldsymbol{\phi}}(\mathbf{x}, \omega^i), \mathbf{z}_1^{(i)} \sim \mathcal{T}_1(\tilde{\mathbf{z}}_0^i)$; $\mathcal{D}(\cdot, \cdot)$ is a metric between samples specified below. We call this procedure distilling knowledge from $\mathcal{T}_1$ to $Q_{\boldsymbol{\phi}}(\cdot)$. In practice, one can choose to distill knowledge for several steps (*e.g.*, $\mathcal{T}_k$) instead of one step (*e.g.*, $\mathcal{T}_1$) to $Q_{\boldsymbol{\phi}}(\cdot)$ each time. Note the distillation idea is related to Bayesian dark knowledge (Korattikara et al., 2015), but with different goal and approach.

After distilling knowledge from $\mathcal{T}_1$, we apply the same procedure for other transformations $\mathcal{T}_k (k > 1)$ sequentially. The final inference network, represented by $q_{\boldsymbol{\phi}}(\cdot \,|\, \mathbf{x})$, can then well approximate the CTF, *e.g.*, the distribution of $\mathbf{z}_0 \sim q_{\boldsymbol{\phi}}(\cdot \,|\, \mathbf{x})$ is close to ρ_T from the CTF. This concept is illustrated in Figure 2. We note choosing an appropriate $\mathcal{D}$ in (8) is important in order to make Theorem 2 applicable. Amortized SVGD (Wang & Liu, 2017) defines $\mathcal{D}$ as standard Euclidean distance between samples. We show in Proposition 4 that this would induce a large error in terms of approximation accuracy.

Proposition 4 *Fix $\boldsymbol{\theta}$. If D in (8) is defined as the summation of pairwise Euclidean distance between samples, then samples generated from $Q_{\boldsymbol{\phi}}$ converge to local modes of $\log p_{\boldsymbol{\theta}}(\mathbf{z} \,|\, \mathbf{x})$.*

[¶]This is distinct from our density-estimation framework described in the next section, where an explicit form is assumed at the beginning for practical needs.

Consequently, it is crucial to impose more general distance for $\mathcal{D}$. As GAN has been interpreted as distribution matching (Li et al., 2017a), we define $\mathcal{D}$ using the Wasserstein distance, implemented as a discriminator parameterized by a neural network. Specifically, we adopt the ALICE framework (Li et al., 2017a), and use $\{(\mathbf{x}, \mathbf{z}_0'^{(i)})\}$ as fake data and $\{(\mathbf{z}_1^{(i)}, \mathbf{x}^i \sim p_{\boldsymbol{\theta}}(\cdot \,|\, \mathbf{z}_1^{(i)}))\}$ as real data to train a discriminator. More details are discussed in Section C of the SM.

Updating $\boldsymbol{\theta}$ Given $\boldsymbol{\phi}$, $\boldsymbol{\theta}$ can be updated by simply optimizing the ELBO in (7), where ρ_T is approximated by $\bar{\rho}_T$ from the discretized CTF. Specifically, the expectation w.r.t. ρ_T in (7) is approximated by a sample average from:

$$\mathbf{z}_0 \sim q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x}), \mathbf{z}_1 \sim \mathcal{T}_1(\mathbf{z}_0), \mathbf{z}_2 \sim \mathcal{T}_2(\mathbf{z}_1), \cdots, \mathbf{z}_K \sim \mathcal{T}_K(\mathbf{z}_{K-1})$$

To sum up, there are three steps to learn a CTF-based VAE:

1. Generate $(\mathbf{z}_0, \cdots, \mathbf{z}_K)$ according to $q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x})$ and the discretized flow with transformations $\{\mathcal{T}_k\}$;
2. Update $\boldsymbol{\phi}$ according to (8);
3. Optimize $\boldsymbol{\theta}$ by minimizing the ELBO (7) with the generated sample path.

In testing, we use only the finally learned $q_{\boldsymbol{\phi}}(\mathbf{z}_0 \,|\, \mathbf{x})$ for inference (into which the CTF has been distilled), and hence testing is like the standard VAE. Since the discretized-CTF model is essentially a Markov chain, we call our model Markov-chain-based VAE (MacVAE).

4. CTFs for Energy-based Density Estimation

We assume that the density of the observation $\mathbf{x}$ is characterized by a parametric Gibbsian-style probability model $p_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{1}{\mathcal{Z}(\boldsymbol{\theta})} \tilde{p}_{\boldsymbol{\theta}}(\mathbf{x}) \triangleq \frac{1}{\mathcal{Z}(\boldsymbol{\theta})} e^{U(\mathbf{x}; \boldsymbol{\theta})}$, where $\tilde{p}_{\boldsymbol{\theta}}(\mathbf{x})$ is an unnormalized version of $p_{\boldsymbol{\theta}}(\mathbf{x})$ with parameter $\boldsymbol{\theta}$, $U(\mathbf{x}; \boldsymbol{\theta}) \triangleq \log \tilde{p}_{\boldsymbol{\theta}}(\mathbf{x})$ is called the energy function (Zhao et al., 2017), and $\mathcal{Z}(\boldsymbol{\theta}) \triangleq \int \tilde{p}_{\boldsymbol{\theta}}(\mathbf{x}) \mathrm{d}\, \mathbf{x}$ is the normalizer. Note this form of distributions constitutes a very large class of distributions as long as the capacity of the energy function is large enough. This can be easily achieved by adopting a DNN to implement $U(\mathbf{x}; \boldsymbol{\theta})$, the setting we considered in this paper. Note our model can be placed in between existing implicit and explicit density estimation methods, because we model the data density with an explicit distribution form up to an intractable normalizer. Such distributions have been proved to be useful in real applications, *e.g.*, (Haarnoja et al., 2017) used them to model policies in deep reinforcement learning.

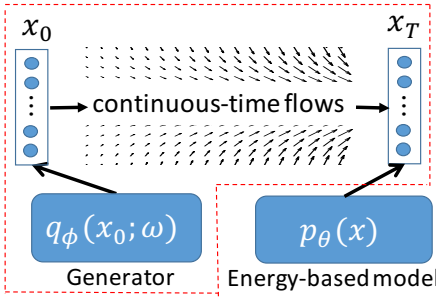

Figure 3. Learning a generator with CTF. The goal is to match the samples $\mathbf{x}_0$ from $q_{\boldsymbol{\phi}}$ to those after a CTF ($\mathbf{x}_T$).

Our goal is to learn $\boldsymbol{\theta}$ given $\{\mathbf{x}_i\}_{i=1}^N$, which can be achieved via standard maximum likelihood estimation (MLE): $\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^N \log p_{\boldsymbol{\theta}}(\mathbf{x}_i) \triangleq \arg\max_{\boldsymbol{\theta}} \mathcal{M}(\{\mathbf{x}_i\}; \boldsymbol{\theta})$.

The optimization can be achieved by standard stochastic gradient descent (SGD), with the following gradient formula:

$$\frac{\partial \mathcal{M}}{\partial \boldsymbol{\theta}} = \frac{1}{N}\sum_{i=1}^N \frac{\partial U(\mathbf{x}_i;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} - \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{x})}\left[\frac{\partial U(\mathbf{x};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right] \tag{9}$$

The above formula requires an integration over the model distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$, which can be approximated by Monte Carlo integration with samples. Here we adopt the idea of CTFs and propose to use a DNN guided by a CTF, which we call a *generator*, to generate approximate samples from the original model $p_{\boldsymbol{\theta}}(\mathbf{x})$. Specifically, we require that samples from the generator should well approximate the target $p_{\boldsymbol{\theta}}(\mathbf{x})$. This can be done by adopting the CTF idea above, *i.e.*, distilling knowledge of a CTF (which approaches $p_{\boldsymbol{\theta}}(\mathbf{x})$) to the generator. In testing, instead of generating samples from $p_{\boldsymbol{\theta}}(\mathbf{x})$ via MCMC (which is complicated and time consuming), we generate samples from the generator directly. Furthermore, when evaluating the likelihood for test data, the constant $\mathcal{Z}(\boldsymbol{\theta})$ can also be approximated by Monte Carlo integration with samples drawn from the generator.

Note the first term on the RHS of (9) is a model fit to observed data, and the second term is a model fit to synthetic data drawn from $p_{\boldsymbol{\theta}(\mathbf{x})}$; this is similar to the discriminator in GANs (Arjovsky et al., 2017), but derived directly from the MLE. More connections are discussed below.

4.1. Learning via Amortization

Our goal is to learn a generator whose generated samples match those from the original model $p_{\boldsymbol{\theta}}(\mathbf{x})$.Similar to inference setting, the generator is learned implicitly. However, we also learn an explicit density model for the data by SGD, with samples from the implicit generator to estimate gradients in (9). Note that in this case, the CTF is performed directly on the data space, instead of on latent-variable space as in previous sections. Specifically, the sampling procedure from the generator plus a CTF are written as:

$$\mathbf{x}_0 \sim q_{\boldsymbol{\phi}}(\mathbf{x}_0), \mathbf{x}_T \sim \mathcal{T}(\mathbf{x}_0, T)\ .$$

Here $\mathcal{T}(\cdot,\cdot)$ is the continuous-time flow; a sample $\mathbf{x}_0$ from $q_{\boldsymbol{\phi}}(\cdot)$ is implemented by a deep neural network (generator) $G_{\boldsymbol{\phi}}(\omega)$ with input $\omega \sim q_0(\omega)$, where q_0 is a simple distribution for a noise random variable, *e.g.*, the isotropic normal distribution. The procedure is illustrated in Figure 3.

Specifically, denote the parameters in the k-th step with subscript "(k)". For efficient sample generation, in the k-th step, we again adopt the amortization idea from Section 3.2 to update $\boldsymbol{\phi}^{(k-1)}$ of the generator network $G_{\boldsymbol{\phi}}(\cdot)$, such that samples from the updated generator match those from the current generator followed by a one-step transformation $\mathcal{T}_1(\cdot)$. After that, $\boldsymbol{\theta}$ is updated by drawing samples from $q_{\boldsymbol{\phi}}(\cdot)$ to estimate the expectation in (9). The algorithm is presented in Algorithm 1 in Section E of the SM.

4.2. Connections to WGAN and MLE

There is an interesting relation between our model and the WGAN framework (Arjovsky et al., 2017). To see this, let p_r be the data distribution. Substituting $p_{\boldsymbol{\theta}}(\mathbf{x})$ with $q_{\boldsymbol{\phi}}(\mathbf{x})$ for the expectation in the gradient formula (9) and integrating out $\boldsymbol{\theta}$, we have that our objective is

$$\max \mathbb{E}_{\mathbf{x}\sim p_r}\left[U(\mathbf{x};\boldsymbol{\theta})\right] - \mathbb{E}_{\mathbf{x}\sim q_{\boldsymbol{\phi}}}\left[U(\mathbf{x};\boldsymbol{\theta})\right] \tag{10}$$

This is an instance of the integral probability metrics (Arjovsky & Bottou, 2017). When U is chosen to be 1-Lipschitz functions, it recovers WGAN. This connection motivates us to introduce weight clipping (Arjovsky et al., 2017) or alternative regularizers (Gulrajani et al., 2017) when updating $\boldsymbol{\theta}$ for a better theoretical property. For this reason, we call our model Markov-chain-based GAN (MacGAN).

Furthermore, it can be shown by Jensen's inequality that the MLE is bounded by (detailed derivations are provided in Section F of the SM)

$$\max \frac{1}{N}\sum_{i=1}^N \log p_{\boldsymbol{\theta}}(\mathbf{x}_i) \tag{11}$$
$$\leq \max \mathbb{E}_{\mathbf{x}\sim p_r}\left[U(\mathbf{x};\boldsymbol{\theta})\right] - \mathbb{E}_{\mathbf{x}\sim q_{\boldsymbol{\phi}}}\left[U(\mathbf{x};\boldsymbol{\theta})\right] - \mathbb{E}_{\mathbf{x}\sim q_{\boldsymbol{\phi}}}\left[\log q_{\boldsymbol{\phi}}\right]\ .$$

By inspecting (10) and (11), it is clear that: *i*) when learning the energy-based model parameters $\boldsymbol{\theta}$, the objective can be interpreted as maximizing an upper bound of the MLE shown in (11); *ii*) when optimizing the parameter $\boldsymbol{\phi}$ of the inference network, we adopt the amortized learning procedure presented in Algorithm 1, whose objective is $\min_{\boldsymbol{\phi}} \mathrm{KL}\left(q_{\boldsymbol{\phi}} \| p_{\boldsymbol{\theta}}\right)$, coinciding with the last two terms in (11). In other words, both $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are optimized by maximizing the *same* upper bound of the MLE, guaranteeing convergence of the algorithm, although previous work has pointed out maximizing an upper bound is not a well-posed problem in general (Salakhutdinov & Hinton, 2012).

Proposition 5 *The optimal solution of MacGAN is the maximum likelihood estimator.*

Note another difference between MacGAN and standard GAN framework is the way of learning the generator $q_{\boldsymbol{\phi}}$.

We adopt the amortization idea, which directly guides $q_{\boldsymbol{\phi}}$ to approach $p_{\boldsymbol{\theta}}$; whereas in GAN, the generator is optimized via a min-max procedure to make it approach the empirical data distribution p_r. By explicitly learning $p_{\boldsymbol{\theta}}$, MacGAN is able to evaluate likelihood for test data up to a constant.

5. Related Work

Our framework extends the idea of normalizing flows (Rezende & Mohamed, 2015) and gradient flows (Altieri & Duvenaud, 2015) to continuous-time flows, by developing theoretical properties on the convergence behavior. Inference based on CTFs has been studied in (Sohl-Dickstein et al., 2015) based on maximum likelihood and (Salimans et al., 2015) based on the auxiliary-variable technique. However, they directly uses discrete approximations for the flow, and the approximation accuracy is unclear. Moreover, the inference network requires simulating a long Markov chain for the auxiliary model, thus is less efficient than ours. Finally, the inference network is implemented as a parametric distribution (*e.g.*, the Gaussian distribution), limiting the representation power, a common setting in existing auxiliary-variable based models (Tran et al., 2016). The idea of amortization (Gershman & Goodman, 2014) has recently been explored in various research topics for Bayesian inference such as in variational inference (Kingma & Welling, 2014; Rezende et al., 2014) and Markov chain Monte Carlo (Wang & Liu, 2017; Li et al., 2017b; Pu et al., 2017). Both (Wang & Liu, 2017) and (Pu et al., 2017) extend the idea of Stein variational gradient descent (Liu & Wang, 2016) with amortized inference for a GAN-based and a VAE-based model, respectively, which resemble our proposed MacVAE and MacGAN in concept. Li et al. (2017b) applies amortization to distill knowledge from MCMC to learn a student network. The ideas in (Li et al., 2017b) are similar to ours, but the motivation and underlying theory are different from that developed here. Remarkably, the authors endowed standard Euclidean distance in (8) for distribution matching, which is inappropriate as will be shown in our experiments.

6. Experiments

We conduct experiments to test our CTF-based framework for efficient inference and density estimation problems, and compared them with related methods. Some experiments are based on the excellent code for SteinGAN[‖] (Wang & Liu, 2017), where their default parameter setting are adopted. The discretization stepsize h is robust as long as it is set in a reasonable range, *e.g.*, we set it the same as the stepsize in SGD. More experimental results are given in the SM, including a sensitiveness experiment on model parameters in Section G.4.

[‖] https://github.com/DartML/SteinGAN

6.1. CTFs for inference

Synthetic experiment We examine our amortized learning framework with three toy experiments. Particularly, we want to verify the necessity of distribution matching defined in (8), *i.e.*, we test $\mathcal{D}$ implemented as a discriminator for Wasserstein distance (adversarial-CTF) against that implemented with standard Euclidean distance (ℓ_2-CTF), which can be considered as an instance of the amortized MCMC (Li et al., 2017b) with a Langevin-dynamic transition function and a Euclidean-distance-based divergence measure for samples. Two 2D distributions similar to (Rezende & Mohamed, 2015) are considered, defined in Section D of the SM. The inference network $q_{\boldsymbol{\phi}}$ is defined to be a 2-layer MLP with isotropic normal random variables as input. Figure 4 plots the densities estimated with the samples from transformations $\{\mathcal{T}_{K=100}\}$ (before optimizing $\boldsymbol{\phi}$), as well as with samples generated directly from $q_{\boldsymbol{\phi}}$ (after optimizing $\boldsymbol{\phi}$). It is clear that the amortized learning with Wasserstein distance is able to distill knowledge from the CTF to the inference network, while the algorithm fails when Euclidean distance is adopted.

Next, we test MacVAE on a VAE setting on a simple synthetic dataset containing 4 data points, each is a 4D one-hot vector, with the non-zero elements at different positions. The prior of latent code is a 2D standard Normal. Figure 5 plots the distribution of the learned latent code for VAE, adversarial-CTF and ℓ_2-CTF. Each color means the codes for one particular observation. It is observed that VAE divides the space into a mixture of 4 Gaussians (consistent with VAE theory), the adversarial-CTF learns complex posteriors, while the ℓ_2-CTF converges to the mode of each posterior (consistent with Proposition 4).

MacVAE on MNIST Following (Rezende & Mohamed, 2015; Tomczak & Welling, 2016), we define the inference network as a deep neural network with two fully connected layers of size 300 with softplus activation functions. We compare MacVAE with the standard VAE and the VAE with normalizing flow, where testing ELBOs are reported (Section G.1 of the SM describes how to calculate the ELBO). We do not compare with other state-of-the-art methods such as the inverse autoregressive flow (Kingma et al., 2016), because they typically endowed more complicated inference networks (with more parameters), unfair for comparison. We use the same inference network architecture for all the models. Figure 7 (left) plots the testing ELBO versus training epochs. MacVAE outperforms VAE and normalizing flows with a better ELBO (around -85.62).

6.2. CTFs for density estimation

We test MacGAN on three datasets: MNIST, CIFAR-10 and CelabA. Following GAN-related methods, the model is evaluated by observing its ability to draw samples from the learned data distribution. Inspiring by (Wang & Liu, 2017),

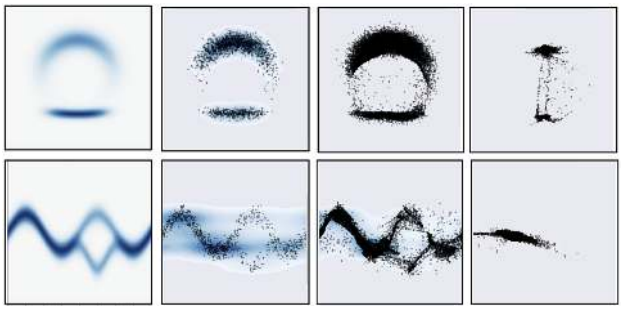

Figure 4. Illustration of CTF on toy distributions. Each row is a distribution case. 1st column: true distributions; 2nd column: MCMC results; 3rd column: approximations via adversarial-CTF; 4th column: approximations via ℓ_2-CTF.

Figure 5. Comparison of the learned latent space with standard VAE (left), adversarial-CTF (middle) and ℓ_2-CTF (right).

we define a parametric form of the energy-based model as $p_{\boldsymbol{\theta}}(\mathbf{x}) \propto \exp\{-\|\mathbf{x}-\text{DEC}_{\boldsymbol{\theta}}(\text{ENC}_{\boldsymbol{\theta}}(\mathbf{x}))\|^2\}$, where $\text{ENC}_{\boldsymbol{\theta}}(\cdot)$ and $\text{DEC}_{\boldsymbol{\theta}}(\cdot)$ are encoder and decoder defined by using deep convolutional neural networks and deconvolutional neural networks, respectively, parameterized by $\boldsymbol{\theta}$. For simplicity, we adopt the popular DCGAN architecture (Radford et al., 2016) for the encoder and decoder. The generator $G_{\boldsymbol{\phi}}$ is defined as a 3-layer CNN with the ReLU activation function (except for the top layer which uses tanh as the activation function, see SM G for details). Following (Wang & Liu, 2017), the stepsizes are set to $\frac{(m_e-e)\times l_r}{m_e-50}$, where e indexes the epoch, m_e is the total number of epochs, $l_r = $ 1e-4 when updating $\boldsymbol{\theta}$, and $l_r = $ 1e-3 when updating $\boldsymbol{\phi}$. The stepsize in $\mathcal{L}_1$ is set to 1e-3.

We compare MacGAN with DCGAN (Radford et al., 2016), the improved WGAN (WGAN-I) (Gulrajani et al., 2017) and SteinGAN (Wang & Liu, 2017). We plot images generated with MacGAN and its most related method SteinGAN in Figure 6 for CelebA and CIFAR-10 datasets. More results are provided in SM Section G. We observe that visually MacGAN is able to generate clear-looking images. Following (Wang & Liu, 2017), we also plot the images generated by a random walk in the ω space in Figure 6.

Qualitatively evaluating a GAN-like model is challenging. We follow literature and use the inception score (Salimans et al., 2016) to measure the quantity of the generated images. Figure 7 (right) plots inception scores vs epochs for different models. MacGAN obtains competitive inception scores with the popular DCGAN model. Quantitatively, we get a final inception score of 6.49 for MacGAN, compared to 6.35 for

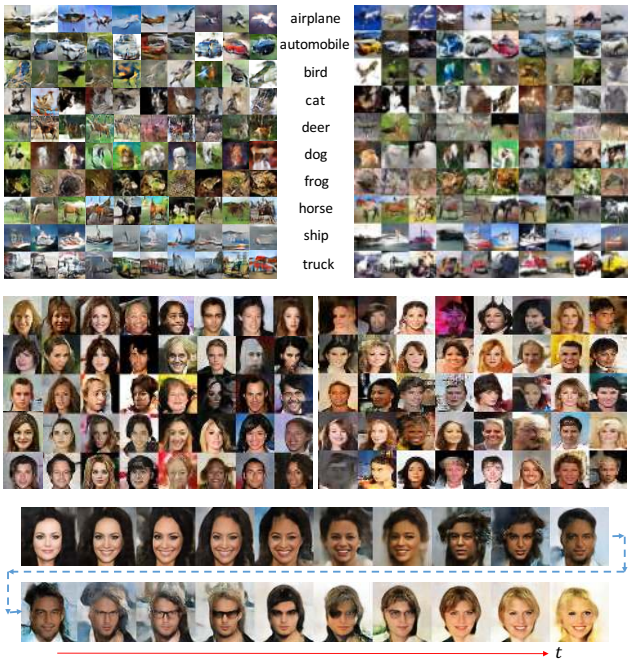

Figure 6. Generated images for CIFAR-10 (top) and CelebA (middle) datasets with MacGAN (left) and SteinGAN (right). The bottom are images generated by a random walk on the ω space for the generator of MacGAN, *i.e.*, $\omega_t = \omega_{t-1} + 0.03 \times \text{rand}([-1, 1])$.

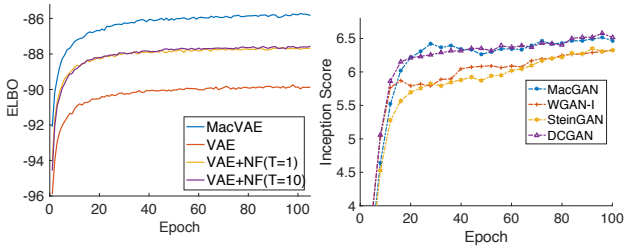

Figure 7. ELBO on MNIST vs epochs (left), and Inception score versus epochs (right) for different models. VAE with 80-layer NF is not included because it has much more parameters.

SteinGAN, 6.25 for WGAN-I and 6.58 for DCGAN.

7. Conclusion

We study the problem of applying CTFs for efficient inference and explicit density estimation in deep generative models, two important tasks in unsupervised learning. Compared to discrete-time NFs, CTFs are more general and flexible due to the fact that their stationary distributions can be controlled without extra flow parameters. We develop theory on the approximation accuracy when adopting a CTF to approximate a target distribution. We apply CTFs on two classes of deep generative models, a variational autoencoder for efficient inference, and a GAN-like density estimator for explicit density estimation and efficient data generation. Experiments show encouraging results of our framework in both models compared to existing techniques. One interesting direction of future work is to explore more efficient learning algorithms for the proposed CTF-based framework.

Acknowledgements

We thank the anonymous reviewers for their useful comments. This research was supported in part by DARPA, DOE, NIH, ONR and NSF.

References

Altieri, N. and Duvenaud, D. Variational inference with gradient flows. In *NIPS workshop on Advances in Approximate Bayesian Inference*, 2015.

Arjovsky, M. and Bottou, L. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein GAN. In *ICML*, 2017.

Bolley, F., Gentil, I., and Guillin, A. Convergence to equilibrium in wasserstein distance for fokker–planck equations. *Journal of Functional Analysis*, 263(8):2430–2457, 2012.

Bottou, L. Stochastic gradient descent tricks. Technical report, Microsoft Research, Redmond, WA, 2012.

Chen, C., Ding, N., and Carin, L. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *NIPS*, 2015.

Chen, C., Carlson, D., Gan, Z., Li, C., and Carin, L. Bridging the gap between stochastic gradient MCMC and stochastic optimization. In *AISTATS*, 2016.

Ding, N., Fang, Y., Babbush, R., Chen, C., Skeel, R. D., and Neven, H. Bayesian sampling using stochastic gradient thermostats. In *NIPS*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *ICLR*, 2017.

Dorogovtsev, A. A. and Nishchenko, I. I. An analysis of stochastic flows. *Communications on Stochastic Analysis*, 8(3):331–342, 2014.

Feng, Y., Wang, D., and Liu, Q. Learning to draw samples with amortized Stein variational gradient descent. In *UAI*, 2017.

Gershman, S. J. and Goodman, N. D. Amortized inference in probabilistic reasoning. In *Annual Conference of the Cognitive Science Society*, 2014.

Givens, C. R. and Shortt, R. M. A class of wasserstein metrics for probability distributions. *Michigan Math. J.*, 31, 1984.

Grover, A., Dhar, M., and Ermon, S. Flow-GAN: Bridging implicit and prescribed learning in generative models. Technical Report arXiv:1705.08868, 2017.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. Improved training of Wasserstein GAN. In *NIPS*, 2017.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *ICML*, 2017.

Huszár, F. Variational inference using implicit distributions. Technical Report arXiv:1702.08235, 2017.

Jordan, R., Kinderlehrer, D., and Otto, F. The variational formulation of the Fokker-Planck equation. *SIAM J. MATH. ANAL.*, 29(1):1–17, 1998.

Kingma, D., Salimans, T. P., and Welling, M. Improving variational inference with inverse autoregressive flow. In *NIPS*, 2016.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *ICLR*, 2014.

Korattikara, A., Rathod, V., Murphy, K., and Welling, M. Bayesian dark knowledge. In *NIPS*, 2015.

Li, C., Chen, C., Carlson, D., and Carin, L. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *AAAI*, 2016a.

Li, C., Steven, A., Chen, C., and Carin, L. Learning weight uncertainty with stochastic gradient MCMC for shape classification. In *CVPR*, 2016b.

Li, C., Liu, H., Chen, C., Pu, Y., Chen, L., Henao, R., and Carin, L. ALICE: Towards understanding adversarial learning for joint distribution matching. In *NIPS*, 2017a.

Li, Y., Turner, R. E., and Liu, Q. Approximate inference with amortised MCMC. Technical Report arXiv:1702.08343, 2017b.

Liu, Q. and Wang, D. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *NIPS*, 2016.

Mattingly, J. C., Stuart, A. M., and Tretyakov, M. V. Construction of numerical time-average and stationary measures via Poisson equations. *SIAM Journal on Numerical Analysis*, 48(2):552–577, 2010.

Mohamed, S. and Lakshminarayanan, B. Learning in implicit generative models. Technical Report arXiv:1610.03483, 2017.

Otto, F. Dynamics of Labyrinthine pattern formation in magnetic fluids: A mean-field theory. *Arch. Rational Mech. Anal.*, pp. 63–103, 1998.

Pu, Y., Gan, Z., Henao, R., Li, C., Han, S., and Carin, L. VAE learning via Stein variational gradient descent. In *NIPS*, 2017.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. Technical Report arXiv:1511.06434, January 2016.

Ranganath, R., Altosaar, J., Tran, D., and Blei, D. M. Operator variational inference. In *NIPS*, 2016.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In *ICML*, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Risken, H. *The Fokker-Planck equation*. Springer-Verlag, New York, 1989.

Salakhutdinov, R. and Hinton, G. An efficient learning procedure for deep machines. *Neural Computation*, 24 (8):1967–2006, 2012.

Salimans, T., Kingma, D. P., and Welling, M. Markov chain Monte Carlo and variational inference: Bridging the gap. In *ICML*, 2015.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training GANs. In *NIPS*, 2016.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

Teh, Y. W., Thiery, A. H., and Vollmer, S. J. Consistency and fluctuations for stochastic gradient Langevin dynamics. *JMLR*, 17(1):193–225, 2016.

Tomczak, J. M. and Welling, M. Improving variational auto-encoders using Householder flow. Technical Report arXiv:1611.09630, November 2016.

Tran, D., Ranganath, R., and Blei, D. M. The variational Gaussian process. In *ICLR*, 2016.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In *ICML*, 2016.

Vollmer, S. J., Zygalakis, K. C., and Teh, Y. W. Exploration of the (Non-)asymptotic bias and variance of stochastic gradient Langevin dynamics. *JMLR*, 1:1–48, 2016.

Wang, D. and Liu, Q. Learning to draw samples: With application to amortized MLE for generative adversarial learning. In *ICLR workshop*, 2017.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient Langevin dynamics. In *ICML*, 2011.

Zhang, Y., Chen, C., Gan, Z., Henao, R., and Carin, L. Stochastic gradient monomial Gamma sampler. In *ICML*, 2017.

Zhao, J., Mathieu, M., and LeCun, Y. Energy-based generative adversarial networks. In *ICLR*, 2017.