Path-Gradient Estimators for Continuous Normalizing Flows

Lorenz Vaitl [1] Kim Nicoli [1 2] Shinichi Nakajima [1 2 3] Pan Kessel [1 2]

Abstract

Recent work has established a path-gradient estimator for simple variational Gaussian distributions and has argued that the path-gradient is particularly beneficial in the regime in which the variational distribution approaches the exact target distribution. In many applications, this regime can however not be reached by a simple Gaussian variational distribution. In this work, we overcome this crucial limitation by proposing a path-gradient estimator for the considerably more expressive variational family of continuous normalizing flows. We outline an efficient algorithm to calculate this estimator and establish its superior performance empirically.

1. Introduction

Variational Inference is increasingly applied to very high-dimensional systems in various application domains. For example, recent developments in Variational Autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014; Kingma et al., 2016; van den Berg et al., 2018; Chen et al., 2018) have led to models that can generate high-resolution natural images. Similarly promising examples are applications in computer graphics (Müller et al., 2019), theoretical physics (Wu et al., 2019; Nicoli et al., 2020; Albergo et al., 2021; Nicoli et al., 2021) and computational chemistry (Noé et al., 2019).

A recurrent theme of most of these examples is that a highly expressive normalizing flow model is used as the variational density which can be trained by maximizing the evidence lower bound (ELBO) with the reparameterization trick (Kingma & Welling, 2014; Rezende et al., 2014). These models are considerably more expressive than traditional models, such as (mixtures of) Gaussians, and thereby enable the application of Variational Inference to these highly complex problems.

A promising subclass of these models are continuous normalizing flows (Chen et al., 2018) because they allow for a free-form Jacobian (Grathwohl et al., 2019) and are particularly suited for incorporating known symmetries of the target density (Köhler et al., 2020) as inductive bias in their architecture. The latter point is of crucial importance in many applications in the natural sciences, (see e.g. Köhler et al. (2020); Kanwar et al. (2020); de Haan et al. (2021)).

In a seminal paper, Roeder et al. (2017) proposed a promising alternative training method for Variational Inference. Specifically, they derived a new unbiased estimator for the gradient of the ELBO with respect to the variational parameters. Unlike the standard estimator, their proposed estimator is not based on the total derivative but rather on the path gradient, i.e. it only takes into account the implicit dependency on the variational parameters through reparameterized sampling but is insensitive to any explicit parameter dependency. From theoretical arguments, which we will discuss in Section 3, it may be expected that this estimator has lower variance than the standard estimator. In their experiments, Roeder et al. (2017) demonstrated that, for comparatively simple variational models, this estimator indeed leads to faster convergences and better approximation results.

Unfortunately, their method cannot be efficiently implemented for normalizing flows for reasons that we will describe in detail in Section 4. As a result, the path-gradient estimator cannot efficiently be applied to the modern normalizing-flow-based applications of Variational Inference discussed above.

The main contribution of this work is to propose a method to estimate the path-gradient for continuous normalizing flow models. Our method has only slightly larger runtime and the same memory costs per iteration as the standard total derivative training. This mild increase in computational cost per iteration is however more than compensated by faster convergence often leading to a substantial speed-up in training time and better overall training results.

We demonstrate for two application domains, i.e. VAEs and lattice field theories in theoretical physics, that a simple re-

[1]Machine Learning Group, Department of Electrical Engineering & Computer Science, Technische Universität Berlin, Germany [2]BIFOLD - Berlin Institute for the Foundations of Learning and Data, Technische Universität Berlin, Berlin, Germany [3]RIKEN Center for AIP, 103-0027 Tokyo, Chuo City, Japan. Correspondence to: Pan Kessel <pan.kessel@tu-berlin.de>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

placement of the standard total gradient by the path-gradient estimator improves the performance across different architectures of continuous normalizing flows, datasets as well as for fixed and adaptive step-size ODE solvers.

2. Related Work

Path gradients were introduced by Roeder et al. (2017). In this reference, the authors propose an algorithm to calculate path-gradients for certain simple variational families and demonstrate its superior performance on various VAE tasks. In Tucker et al. (2019), the authors extended the results of Roeder et al. (2017) to other VAE losses, such as Importance Weighted Autoencoder (Burda et al., 2016), Reweighted Wake Sleep (Bornschein & Bengio, 2015), and Jackknife Variational Autoencoder (Nowozin, 2018). Later work (Finke & Thiery, 2019; Geffner & Domke, 2020; 2021; Bauer & Mnih, 2021) extended these results to further VAE loss functions, for example based on α-divergences, and clarified theoretical aspects of the original references.

We extend on these reference by proposing a method to estimate the path-gradient which can efficiently be implemented for Normalizing Flows. To the best of our knowledge, Agrawal et al. (2020) is the only reference studying path-wise gradient estimators of normalizing flows as part of a broader ablation study for comparatively simple models from the STAN library. In comparison to our method, their proposal has twice the memory and substantially larger runtime costs. This severely restricts their applicability in more complex problem settings. We refer to Section 4 for a more detailed discussion.

3. Variational Inference with Path-Gradients

In Variational Inference, we aim to approximate a target density p by a variational density q_θ with parameters θ. We denote the sampling space of these densities by $\mathcal{X}$. Typically, the target density p is only known in its unnormalized form $\hat{p}$ with

$$p(x) = \frac{1}{Z}\hat{p}(x)\,,$$

while the partition function Z is computationally intractable.

This specific type of Variational Inference arises in many application domains such as Variational Autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014), Boltzmann generators in Quantum Chemistry (Noé et al., 2019; Köhler et al., 2020), as well as Generalized Neural Samplers in Statistical Physics (Wu et al., 2019; Nicoli et al., 2020) and Lattice Field Theories (Kanwar et al., 2020; Albergo et al., 2021; Nicoli et al., 2021; de Haan et al., 2021).

In the following, we will assume that the variational density $q_\theta(x)$ is obtained by reparameterization $g_\theta : \mathcal{Z} \to \mathcal{X}$ of a base density q_Z with sampling space $\mathcal{Z}$, i.e. any sample $x \in \mathcal{X}$ can be written as

$$x = g_\theta(z)$$

for some sample $z \in \mathcal{Z}$ from the base density q_Z. An example is the variational family of Gaussians $\mathcal{N}(\mu, \sigma^2)$ parameterized by the mean μ and the variance σ^2 with base density $q_Z = \mathcal{N}(0,1)$ and

$$g_\theta(z) = \sigma z + \mu\,, \qquad \text{with} \qquad \theta \in \{\mu, \sigma^2\}\,.$$

Another example are normalizing flows for which g_θ is given by an invertible neural network. In particular, reparameterized densities allow for the reparameterization trick

$$\mathbb{E}_{x\sim q_\theta}\left[f(x)\right] = \mathbb{E}_{z\sim q_z}\left[f(g_\theta(z))\right] \tag{1}$$

where $f : \mathcal{X} \to \mathbb{R}$ is an arbitrary function.

The most widely used optimization criterion in Variational Inference is obtained by minimization of the reverse Kullback–Leibler divergence $\mathrm{KL}(q_\theta, p) = \mathbb{E}_{x\sim q_\theta}\left[\ln \frac{q_\theta(x)}{p(x)}\right]$. Crucially, its derivative with respect to the variational parameters does not depend on the partition function Z, i.e.

$$\frac{d}{d\theta}\mathrm{KL}(q_\theta, p) = \frac{d}{d\theta}\mathcal{F}(\theta)\,,$$

where we introduced

$$\begin{aligned}\mathcal{F}(\theta) &= \mathbb{E}_{z\sim q_z}\left[\ln q_\theta(g_\theta(z)) + E(g_\theta(z))\right] \\ &= \mathbb{E}_{z\sim q_z}\left[f(\theta, g_\theta(z))\right]\,,\end{aligned}$$

with energy $E(x) = -\ln\hat{p}(x)$ and $f(\theta, g_\theta(z)) = \ln q_\theta(g_\theta(z)) + E(g_\theta(z))$. One commonly refers to the quantity $\mathcal{F}$ as the variational free energy and to its negative $-\mathcal{F}$ as the evidence lower bound (ELBO). In the physics community, the former terminology is preferred while the machine learning literature tends to use the latter.

Using the reparameterization trick (1), the total derivative of the reverse KL divergence can be written as

$$\tfrac{d}{d\theta}\mathrm{KL}(q, p) = \tfrac{d}{d\theta}\mathcal{F}(\theta) = \mathbb{E}_{z\sim q_Z}\left[\tfrac{d}{d\theta}f(\theta, g_\theta(z))\right]. \tag{2}$$

The total derivative of the function f with respect to the variational parameters can be decomposed as follows

$$\frac{d}{d\theta}f(g_\theta(z), \theta) = \blacktriangledown_\theta f(g_\theta(z), \theta) + \left.\frac{\partial}{\partial\theta}f(x, \theta)\right|_{x=g_\theta(z)}\,,$$

where we have introduced the *path-gradient*

$$\blacktriangledown_\theta f(g_\theta(z), \theta) = \frac{\partial f(g_\theta(z), \theta)}{\partial g_\theta(z)}^{\intercal}\frac{\partial g_\theta(z)}{\partial\theta}\,,$$

i.e. the path-gradient only takes into account the implicit dependency on θ through the reparameterization g_θ and is insensitive to any explicit dependency. The total derivative of the reverse KL divergences therefore splits in two terms

$$\frac{d}{d\theta}\mathrm{KL}(q,p) \\ = \mathbb{E}_{z\sim q_Z}[\blacktriangledown_\theta f(g_\theta(z),\theta)] + \mathbb{E}_{z\sim q_Z}[\frac{\partial}{\partial\theta} f(x,\theta)\Big|_{x=g_\theta(z)}].$$

The latter is known as the score term and vanishes because the reparameterization trick (1) implies that

$$\mathbb{E}_{z\sim q_Z}[\frac{\partial}{\partial\theta} f(x,\theta)\Big|_{x=g_\theta(z)}] = \mathbb{E}_{z\sim q_Z}[\frac{\partial}{\partial\theta}\ln q_\theta(x)\Big|_{x=g_\theta(z)}] \\ = \mathbb{E}_{x\sim q_\theta}[\frac{\partial}{\partial\theta}\ln q_\theta(x)] = \frac{\partial}{\partial\theta}\int \mathrm{d}x\, q_\theta(x) = 0\,.$$

The total derivative of the reverse KL divergence (2) is then estimated by Monte-Carlo

$$\frac{d}{d\theta}\mathrm{KL}(q_\theta,p) \approx \mathcal{G}_{\text{total}} = \mathcal{G}_{\text{score}} + \mathcal{G}_{\text{path}} \tag{3}$$

where we have defined

$$\mathcal{G}_{\text{path}} = \frac{1}{N}\sum_{i=1}^{N} \blacktriangledown_\theta \left(\ln q_\theta(g_\theta(z_i)) + E(g_\theta(z_i))\right)\,, \tag{4}$$

$$\mathcal{G}_{\text{score}} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial}{\partial\theta}\ln q_\theta(g_\theta(z_i))\,, \tag{5}$$

with $z_i \sim q_Z$. We will refer to $\mathcal{G}_{\text{path}}$ as the path-gradient estimator and to $\mathcal{G}_{\text{score}}$ as the score estimator.

Both the total estimator $\mathcal{G}_{\text{total}}$ and the path-gradient estimator $\mathcal{G}_{\text{path}}$ are unbiased estimators of the gradient of the reverse KL divergence (2). However, the estimators $\mathcal{G}_{\text{path}}$ and $\mathcal{G}_{\text{total}}$ have generically different variances because the variance of the score $\mathcal{G}_{\text{score}}$ is given by the Fisher information $\mathcal{I}(\theta)$ of the variational density q_θ, i.e.

$$\mathrm{Cov}_{z_1,\dots,z_N\sim q_Z}\left[\mathcal{G}_{\text{score}}\right] = \mathcal{I}(\theta)\,.$$

We refer to the Supplement A for a more detailed discussion.

This difference in variance of the two estimators is particularly illustrative for a variational density q_{θ^*} which perfectly approximates the target p, i.e.

$$\forall x \in \mathcal{X}: \qquad q_{\theta^*}(x) = p(x)\,. \tag{6}$$

For this case, the path-gradient estimator $\mathcal{G}_{\text{path}}$ vanishes identically because

$$\mathcal{G}_{\text{path}} = \frac{1}{N}\sum_{i=1}^{N}\blacktriangledown_\theta \ln\frac{q_\theta(g_\theta(z_i))}{p(g_\theta(z_i))}\Big|_{\theta^*} \\ = \frac{1}{N}\sum_{i=1}^{N}\underbrace{\frac{\partial}{\partial x_i}\ln\frac{q_{\theta^*}(x_i)}{p(x_i)}}_{=0}\frac{\partial g_\theta}{\partial\theta}\Big|_{\theta^*} = 0\,,$$

where we have used (6) in the last step. As a result, the total gradient estimator $\mathcal{G}_{\text{total}}$ has non-vanishing variance $\mathcal{I}(\theta^*)$ even if the target p is perfectly modelled by the variational density q_{θ^*}. The path-gradient estimator $\mathcal{G}_{\text{path}}$ on the other hand does not have this problem.

By continuity, one may therefore expect that the path-gradient estimator $\mathcal{G}_{\text{path}}$ has lower variance than the total gradient estimator $\mathcal{G}_{\text{total}}$ if the variational density q_θ is close to the target density p, i.e. in the final phase of training.

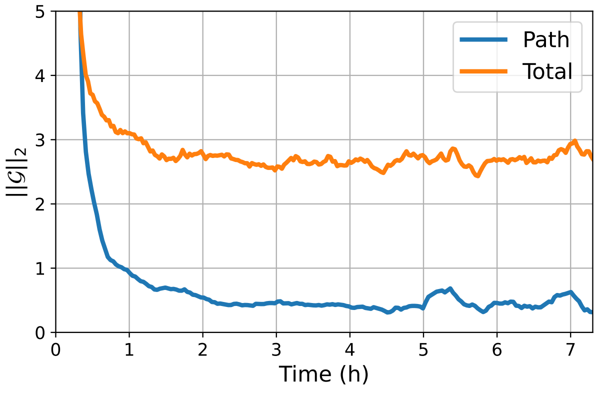

Figure 1. Norm of gradient estimators averaged over three runs for the ϕ^4 lattice field theory with a rolling sum over 10 epochs for a lattice size of $L = 12$.

3.1. Path-Gradients for Simple Densities

For the path-gradient estimator (4), we need to calculate

$$\blacktriangledown_\theta \ln q_\theta(g_\theta(z)) + \blacktriangledown_\theta E(g_\theta(z))\,.$$

The second term can trivially be obtained by automatic differentiation because the total derivative leads to the same result as the path gradient, i.e. $\frac{d}{d\theta}E(g_\theta(z)) = \blacktriangledown_\theta E(g_\theta(z))$. The first term is however non-trivial to calculate.

Roeder et al. (2017) proposed an algorithm for estimating path-gradients which proceeds in two steps:

1. Sample from q_θ by

$$x = g_\theta(z)\,.$$

   One can differentiate through this operation due to the reparameterization property of the density.

2. Evaluate the probability density of the sample by formally using different parameters θ'. The path-gradient is obtained by differentiating the result with respect to the original parameters θ

$$\blacktriangledown_\theta \ln q(x,\theta) = \frac{d}{d\theta}\ln q(g_\theta(z),\theta')\,.$$

This approach can be efficiently implemented if the sampling process can be effectively disentangled from the evaluation of the log density, as is the case for (mixtures of) Gaussians and for other density known in closed-form.

For normalizing flows, this is however not the case: the density contains the determinant of the Jacobian $|\frac{dg_\theta}{dz}|$ which is calculated during the sampling process. To the best of our knowledge, the only reference using path-gradients for invertible normalizing flow is Agrawal et al. (2020). In this reference, the authors calculate the path-gradient for a normalizing flow by considering two separate copies of the model for sampling and density evaluation. This doubles the runtime as well as the memory footprint and thereby reduces the possible mini-batch sizes in training. Large mini-batch sizes are however of central importance for successful training in many relevant application domains (see e.g. Del Debbio et al. (2021); Kanwar (2021)).

For the simple class of variational densities discussed above, Roeder et al. (2017); Tucker et al. (2019); Finke & Thiery (2019); Geffner & Domke (2020) have convincingly established that path-gradients lead to better optimization behaviour in practice.

4. Path-Gradients for Continuous Normalizing Flows

In the following, we will introduce an efficient algorithm to calculate path-gradients for continuous normalizing flows.

Continuous normalizing flows (Chen et al., 2018) transform a sample $z_0 \sim q_Z$ from a base-density q_Z using

$$\begin{aligned} x \equiv z_T &= g_\theta(z_0) \\ &= z_0 + \int_0^T \mathrm{d}t\, f_\theta(z_t, t)\,. \end{aligned} \tag{7}$$

Under the mild assumptions of the Picard–Lindelöf theorem, the map g_θ is bijective. In practice, the map g_θ is implemented using a Neural Ordinary Differential Equation (NODE) (Chen et al., 2018) of the form

$$\frac{dz_t}{dt} = f_\theta(z_t, t)\,, \tag{8}$$

where f_θ is a (not necessarily invertible) neural network with parameters θ and z_t is the intermediate state at time t. The determinant of the Jacobian of the reparameterization g_θ is given by

$$\ln\left|\det\frac{dg_\theta}{dz_0}\right| = \int_0^T \mathrm{tr}\left(\frac{df_\theta(z_t,t)}{dz_t}\right)\mathrm{d}t\,.$$

The logarithm of the variational density is thus given by

$$\begin{aligned} \ln q_\theta(x) &= \ln q_Z(g_\theta^{-1}(x)) - \ln\left|\det\frac{dg_\theta}{dz_0}\right|_{z_0=g_\theta^{-1}(x)} \\ &= \ln q_Z(z_0) - \int_0^T \mathrm{tr}\left(\frac{\partial f_\theta(z_t,t)}{\partial z_t}\right)\mathrm{d}t\,. \end{aligned}$$

In practice, the trace is often estimated using the Hutchinson trace estimator (Grathwohl et al., 2019).

4.1. Total Derivatives

One major insight of Chen et al. (2018) was the fact that we can use the adjoint state method (Pontryagin, 1987) for computing the gradients of NODE model instead of using standard backpropagation through the ODE solver. Specifically, for any loss function of the form

$$L(x) = L(z_T) = L\left(z_0 + \int_0^T \mathrm{d}t\, f_\theta(z_t,t)\right)\,,$$

its derivative with respect to the parameters θ can be obtained by

$$\frac{dL}{d\theta} = -\int_0^T \mathrm{d}t\, \left(\frac{\partial L}{\partial z_t}\right)^\intercal \frac{\partial f_\theta(z_t,t)}{\partial\theta}\,,$$

where the adjoint state $a_t = \frac{dL}{dz_t}$ is obtained by solving the terminal value problem

$$\begin{aligned} \frac{da_t}{dt} &= -a_t^\intercal \frac{\partial f_\theta(z_t,t)}{\partial z_t}\,, \\ a_T &= \frac{dL}{dz_T}\,. \end{aligned} \tag{9}$$

This approach allows to compute the gradients of the NODE with constant memory requirements at the price of additional computational costs. This proceeds in two steps:

1. Calculation of z_T by forward integration of (8). The intermediate states z_t are *not* stored in this forward pass to ensure constant memory costs.

2. The adjoint terminal value problem (9) is then integrated starting from the terminal condition $a_T = \frac{dL}{dz_T}$ to obtain the adjoint states $a_t = \frac{dL}{dz_t}$. Since these adjoint states have the same dimensionality as the states z_t, this can be done for about the same computational cost as the forward pass. However, we also need to recompute the intermediate states z_t by an additional reverse evolution of (8) as they were not stored in the forward pass but enter the adjoint terminal value problem (9). So in total, solving (9) has the computational cost of about two forward passes for constant memory requirements.

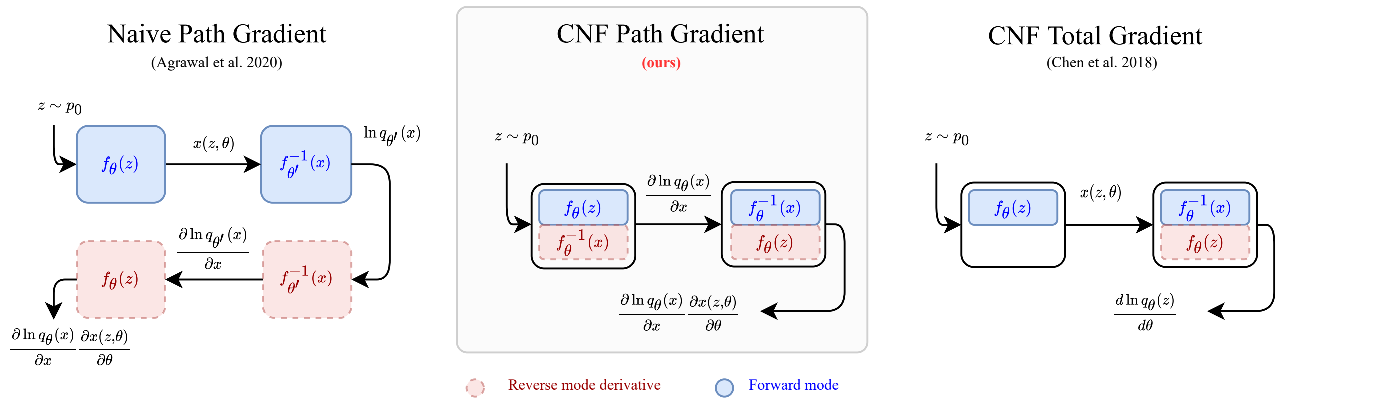

Figure 2. **Left:** method of Agrawal et al. (2020). Top row: two cloned models q_θ and $q_{\theta'}$ are used for sampling and density evaluation. This doubles the memory requirements. Lower row: reverse-mode differentiation through these two cloned models. Each NODE backward pass has the cost of two forward passes. Total cost of (Agrawal et al., 2020): about six forward passes; double memory due to the cloning of the models. **Middle:** our method. Forward pass to calculate $x = z_T$ and the derivative $\frac{\partial \ln q_\theta(z_T)}{\partial z_T}$ at cost of about two forward passes and constant memory. By reverse-mode differentiation, the vector Jacobian product (10) is obtained at costs of about two forward passes and constant memory requirements. Total cost of our method: about four forward passes; half the memory requirements compared to Agrawal et al. (2020). **Right:** standard total gradient approach. Forward is followed by backward pass for which intermediate states are recomputed for memory efficiency. Total costs: about three forward passes. Our method has the same memory costs and $4/3$ times the computational costs but nevertheless converges more quickly.

In summary, the total derivative $\frac{dL}{d\theta}$ of a loss L can be calculated at constant memory with about the computational cost of three forward passes.

4.2. Path Gradients

We now discuss how to calculate the path-gradient for continuous normalizing flows. We recall from our discussion in Section 3.1 that the only non-trivial term of the loss involves

$$\blacktriangledown_\theta \ln q_\theta(g_\theta(z_0)) = \frac{\partial \ln q_\theta(g_\theta(z_0))}{\partial g_\theta(z_0)}^\intercal \frac{\partial g_\theta(z_0)}{\partial \theta} = \frac{\partial \ln q_\theta(z_T)}{\partial z_T}^\intercal \frac{\partial z_T}{\partial \theta}, \qquad (10)$$

where we have used the relation $z_T = g_\theta(z_0)$ as defined in (7).

We prove in Appendix C that the derivative of the log density $\frac{\partial \ln q_\theta(z_T)}{\partial z_T}$ with respect to the final state z_T can be calculated using the following ODE:

Theorem 4.1. *The derivative* $\frac{\partial \ln q_\theta(z_T)}{\partial z_T}$ *can be obtained by solving the initial value problem*

$$\frac{d}{dt}\frac{\partial \ln q_\theta(z_t)}{\partial z_t} = -\frac{\partial \ln q_\theta(z_t)}{\partial z_t}^\intercal \frac{\partial f_\theta(z_t,t)}{\partial z_t} - \partial_{z_t} tr\left(\frac{\partial f_\theta(z_t,t)}{\partial z_t}\right), \qquad (11)$$

with initial condition

$$\frac{\partial \ln q_\theta(z_0)}{\partial z_0} = \frac{\partial \ln q_Z(z_0)}{\partial z_0}.$$

We note that the derivative $\frac{\partial \ln q_\theta(z_t)}{\partial z_t}$ is of the same dimension as the intermediate state z_t. Thus, solving its initial value problem of Theorem 4.1 has about the same computational cost as the forward pass which calculates z_T. As a result, the joint calculation of $\frac{\partial \ln q_\theta(z_t)}{\partial z_t}$ and z_T has about the cost of two forward passes. We then need to calculate the vector Jacobian product (10). This can be done with reverse-mode differentiation using the adjoint state method discussed in Section 4.1 for the costs of about two forward passes (one for the evolution of the adjoint state and the other to re-compute the intermediate states z_t).

In summary, our proposed method to calculate the path-gradient for continuous normalizing flows has the computational cost of about four forward passes as compared to three for the total derivative. Crucially, it also has constant memory requirements.

The following comments are in order:

- The initial value problem (11) of Theorem 4.1 is solved forward in time. As such the intermediate states z_t do not need to be re-computed for this in order to satisfy the constant memory constraint. This is in contrast to the reverse-mode differentiation with the adjoint state method discussed in Section 4.1.

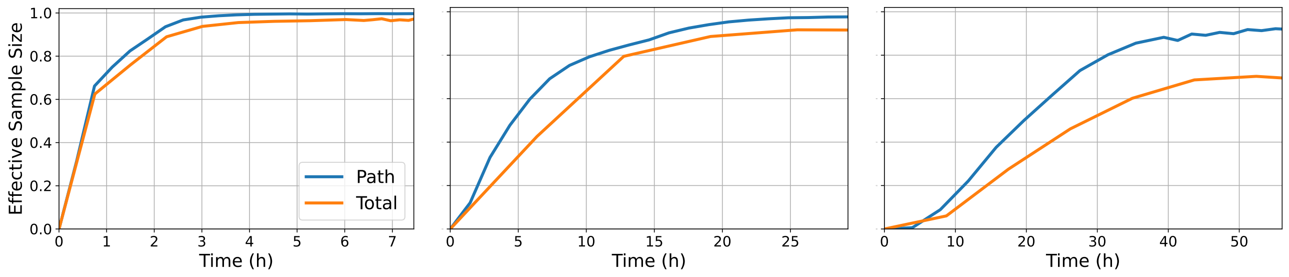

Figure 3. Evolution of effective sampling size (ESS) during training; **Larger is better**. Training for LxL-lattices with $L = 12$ (**left**) and $L = 20$ (**middle**), as well as $L = 32$ (**right**). The path-gradient estimator leads to faster convergence and overall better ESS values. Training was performed on a single $A100$ GPU.

- Our estimate of the runtime ratio of about $4/3$ for the path-gradient compared to the total derivative does not take account synergies between the various ODEs. For example, the derivative $\frac{\partial f_\theta(z_t,t)}{\partial z_t}$ appears both in the adjoint terminal value problem (9) and in the initial value problem (11) of Theorem 4.1. This will result in an effectively smaller difference in runtime as we will demonstrate in our numerical experiments in Section 5.3.

- In our numerical experiments, we will demonstrate that this runtime overhead of the path-gradient estimator will be more than compensated by better convergence properties of the path-gradient estimator, see Section 5. This may be expected due to the smaller variance of the path-gradient estimator, as discussed in Section 3.

- Our method compares very favorably to the approach taken in Agrawal et al. (2020) which has the cost of about six forward passes and twice the memory footprint (compared to both the standard total- and our path-gradient estimator). See Figure 2 for an illustration.

5. Numerical Experiments

We consider two different application domains, i.e. Variational Autoencoders and normalizing flows for lattice field theories. We demonstrate that a simple replacement of the total derivative estimator by the path-gradient estimator improves the performance on both of these tasks. We also discuss the runtime costs per iteration and for the entire training process. We provide code to reproduce the experiments using VAEs[1].

[1]https://github.com/lenz3000/ffjord-path

5.1. Lattice Field Theory

Recently, there has been substantial interest in applying normalizing flows to Lattice Field Theories which can be used to describe the strong interactions of elementary particles such as electrons and quarks. From the machine learning perspective, this application is exciting as incorporating the symmetries of the field theories in the model is of the utmost importance in this domain. Continuous normalizing flows are particularly suitable for ensuring such an inductive bias (see Köhler et al. (2020)).

We repeat the state-of-the-art experiments in de Haan et al. (2021) for a two-dimensional scalar lattice field theory with a quartic interaction.

ϕ^4-theory: this lattice field theory is described by a random vector ϕ with components ϕ_x where the index $x \in \Lambda$ takes values on an $L \times L$ square lattice Λ. The random vector ϕ has the density $p(\phi) = \frac{1}{Z}\exp(-S(\phi))$ with action

$$S(\phi) = \sum_{x,y\in\Lambda} \phi_x \triangle_{xy} \phi_y + \sum_{x\in\Lambda} m^2 \phi_x^2 + \lambda \phi_x^4 ,$$

where $\triangle_{xy}$ is the Laplacian matrix on the lattice Λ and we use periodic boundary conditions. The density p has a $\mathbb{Z}_2$-symmetry, i.e. $p(\phi) = p(-\phi)$. This density is also invariant under the spatial symmetry group $G = C_L^2 \rtimes D_4$ of the lattice Λ, i.e. the semi-direct product of the squared cyclic group C_L and the Dihedral group D_4. We use the state-of-the-art model of de Haan et al. (2021) which is manifestly invariant under these symmetries. We also use the same choices for bare mass m^2 and coupling λ as in de Haan et al. (2021) as well as their three largest lattices.

Training: the reverse KL divergence $\mathrm{KL}(q_\theta, p)$ from the model q_θ to the target density p is minimized using either the standard total derivative estimator $\mathcal{G}_{\text{total}}$ or the path-gradient estimator $\mathcal{G}_{\text{path}}$. Then the performance of the respectively trained models is compared. Both the path-gradient and total gradient training use the same training hyperparameters as in de Haan et al. (2021) except for a larger batch-size of 5000 and learning rate which we find to improve the performance of the baseline. Each model was trained on a single A100 GPU. We refer to the Supplement D for more details on the training procedure.

Effective Sampling Size: one can then estimate expectation values with respect to the *target density* p, i.e. physical observables, using self-normalized importance sampling (Müller et al., 2019; Noé et al., 2019; Nicoli et al., 2020):

$$\mathbb{E}_{\phi\sim p}\left[F(\phi)\right] \approx \hat{F} \equiv \sum_{i=1}^{N} \frac{\tilde{w}_i}{\sum_{j=1}^{N}\tilde{w}_j} F(\phi_i)\,, \quad \phi_i \sim p\,,$$

where $\tilde{w}_i = \frac{\exp(-S(\phi_i))}{q(\phi_i)}$ and F is an arbitrary function. For a variational density q_θ which perfectly approximates the target density p, the variance of the estimator $\hat{F}$ scales as $1/N$. Due to the approximation error of q_θ, in practice a scaling with $1/N_{\text{eff}}$ is observed with $N_{\text{eff}} \leq N$. The quality of the approximation of the target density p by the variational density q_θ is therefore commonly measured in terms of the effective sampling size

$$\text{ESS} = \frac{N_{\text{eff}}}{N} \in [0,1]\,, \tag{12}$$

where values close to the limits of 1 and 0 indicate perfect or very poor approximation of the target, respectively.

In practice, the ESS can be estimated by

$$\text{ESS} \approx \frac{\left(\frac{1}{N}\sum_{i=1}^{N}\tilde{w}_i\right)^2}{\frac{1}{N}\sum_{i=1}^{N}\tilde{w}_i^2}\,.$$

We refer to Nicoli et al. (2020) for more details.

Results: a simple replacement of the total derivative estimator $\mathcal{G}_{\text{total}}$ by the path-gradient estimator $\mathcal{G}_{\text{path}}$ leads to a significant faster learning process as demonstrated in Figure 3. We stress that this speed-up is in terms of runtime and not only number of iterations. Furthermore, Table 1 demonstrates that the final effective sampling size of the models trained by path-gradients is consistently larger as compared to models trained by the total derivative estimator. This effect is more pronounced as the lattice size increases.

Figure 1 shows that the norm of the path-gradient estimator is significantly lower than the norm of the standard estimator, as expected from the discussion in Section 3.

5.2. Variational Autoencoder

We repeat the VAE experiments in Grathwohl et al. (2019) which train a VAE for four datasets using a FFJORD flow.

Table 1. Effective Sample Size estimated over 500k samples after 24k epochs for total derivative estimator $\mathcal{G}_{\text{total}}$ and 20k for path-gradient estimator $\mathcal{G}_{\text{path}}$; **larger is better**. [†] For $L = 32$, we trained for 15.4k/13k epochs.

LATTICE SIZE	PATH	TOTAL
12x12	**99.66** $\pm$ 0.07	98.01 $\pm$ 0.44
20x20	**97.65** $\pm$ 0.14	91.56 $\pm$ 1.13
32x32[†]	**91.81** $\pm$ 1.32	69.53 $\pm$ 5.59

Table 2. Negative ELBO on test data for VAE models using FFJORD flows; **lower is better**. In nats for all datasets except Frey Faces which is presented in bits per dimension. Mean/stdev are estimated over 3 runs. Baseline total derivative estimator results are as in Grathwohl et al. (2019).

DATASET	PATH	TOTAL
MNIST	**82.09** $\pm$ .04	82.82 $\pm$ .01
OMNIGLOT	**96.61** $\pm$ .17	98.33 $\pm$ .09
CALTECH SILHOUETTES	**101.93** $\pm$ .63	104.03 $\pm$ .43
FREY FACES	**4.35** $\pm$ .00	4.39 $\pm$ .01

We use the path-gradient $\mathcal{G}_{\text{path}}$ and the total derivative estimator $\mathcal{G}_{\text{total}}$ for training and compare their respective performances. In contrast to the lattice field theory application above, these experiments use early stopping and an adaptive step size ODE solver. We refer to the Supplement D for more details.

Table 2 summarizes the results for the ELBO. On all considered datasets, the path-gradient estimator outperforms the total gradient estimator.

5.3. Runtime Comparison

We empirically evaluate the runtime costs of our path-gradient estimator compared to the standard total gradient estimator on the two considered tasks.

Lattice Field Theory: we stress that we performed both the path-gradient and total gradient training for the same overall runtime to ensure fair comparison. As discussed in Section 4, we expect the path-gradient estimator to have slightly higher costs *per iteration*. We find that this effect is more than compensated by better convergence properties of the path-gradient.

As expected, we furthermore find in Table 3 that the runtime overhead per iteration is between 14% and 5% depending on the lattice size and thereby substantially lower than the estimate of 33% which neglected synergies between the various ODEs, as discussed in Section 4.

The difference in iteration costs decreases as the size of the lattice increases. This effect can be attributed to fact that the continuous normalizing flow architecture of de Haan

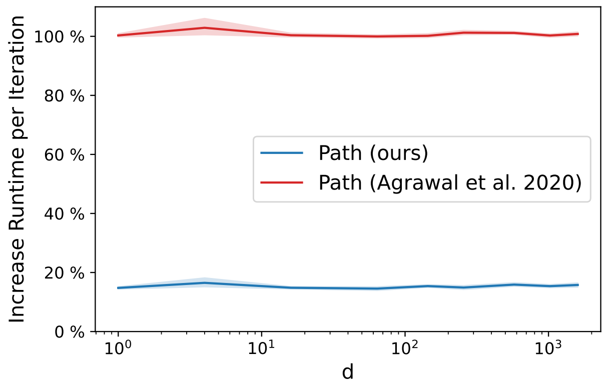

Figure 4. Increase in runtime per iteration for computing the path gradient estimated by our method and the method by Agrawal et al. (2020) as compared to the standard reparametrized gradient estimator measured on an A100 GPU.

et al. (2021) has a quadratic scaling of model parameters with the lattice size. Even if we use a best-case scenario for the total gradient baseline, i.e. a model whose number of parameters is independent of the lattice size, this overhead is mild and stays constant with the problem size, see Figure 4. Furthermore, our method is significantly faster than the method to estimate the path-gradient proposed by Agrawal et al. (2020) which furthermore has twice the memory costs.

In summary, we find empirically that the overhead of our path-gradient estimation method per iteration is mild, decreasing as the dimension increases for the relevant architecture, and significantly outperforms the proposal by Agrawal et al. (2020). Crucially, the path-gradient leads to faster convergence and thus achieves better results using the same walltime.

VAE: The VAE experiments use an adaptive step-size solver for evolving the ODEs. As such, the time per iteration cannot be meaningfully compared. The training also used early-stopping which necessarily implies that training has no fixed runtime.

Nevertheless, we find that the path-gradient training has comparable runtime to the standard total gradient training (largest runtime of our method were 111% for the Omniglot and 98% for the MNIST dataset compared to standard estimator). Reassuringly, we did not see any noticeable effect of the path gradient estimator on the number of function evaluations (see Figure 5).

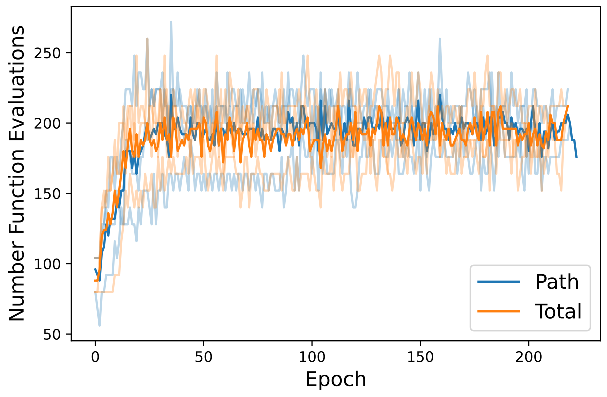

Figure 5. Number of function evaluations for both gradient estimators on the Frey Faces dataset using the same hyperparameters. No trend is recognizable

Table 3. Time increase *per iteration* of the path-gradient over the standard total derivative estimator on an A100 GPU for the lattice field theory experiments of Table 1.

LATTICE SIZE	TIME PER ITERATION
12x12	$+14\% \pm 0$
20x20	$+12\% \pm 1$
32x32	$+5\% \pm 1$

6. Conclusion

We have proposed a path-gradient estimator which can be efficiently implemented for continuous normalizing flows.

For this, we derived an ODE (11) which can be solved forwards in time to obtain the gradient of the variational log density with respect to the output of the flow. From this, we can then straightforwardly obtain the path-gradient.

This approach has same constant memory requirements and only slightly increased runtime per iteration as compared to the standard total gradient estimator. It also is significantly faster per iteration and has half the memory requirements compared to the method proposed by Agrawal et al. (2020). In particular, our method allows for training with sufficiently large mini-batches for modern Variational Inference tasks in the natural sciences, as demonstrated by the lattice field theory application considered in our experiments.

We show empirically that a straightforward plug-and-play replacement of the total gradient by the path-gradient estimator leads to a surprisingly pronounced increase in performance in both the VAE and lattice field theory tasks. As an example, we observe a larger than 30% increase in ESS for highest dimensional lattice field theory task by switching from the standard total derivative to the path-gradient estimator.

A limitation of our work is that so far there is only a theoretical explanation for the lower variance of the path-gradient estimator in the final training phase, as was discussed in Section 3. However, our experimental results suggest that the entire training process benefits from the path-gradient estimator. A theoretical understanding of this would be desirable but seems mathematically very challenging.

In light of our results, we expect that the proposed path-gradient estimator will become a standard item in the toolbox for training continuous normalizing flows on modern Variational Inference tasks.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. K.N., S.N. and P.K. are supported by the German Ministry for Education and Research (BMBF) as BIFOLD - Berlin Institute for the Foundations of Learning and Data under grants 01IS18025A and 01IS18037A.

References

Agrawal, A., Sheldon, D. R., and Domke, J. Advances in black-box VI: normalizing flows, importance weighting, and optimization. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Albergo, M. S., Boyda, D., Hackett, D. C., Kanwar, G., Cranmer, K., Racanière, S., Rezende, D. J., and Shanahan, P. E. Introduction to normalizing flows for lattice field theory. *arXiv preprint arXiv:2101.08176*, 2021.

Bauer, M. and Mnih, A. Generalized doubly-reparameterized gradient estimators. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2021.

Bornschein, J. and Bengio, Y. Reweighted wake-sleep. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Burda, Y., Grosse, R. B., and Salakhutdinov, R. Importance weighted autoencoders. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 6572–6583, 2018.

de Haan, P., Rainone, C., Cheng, M. C. N., and Bondesan, R. Scaling up machine learning for quantum field theory with equivariant continuous flows. *CoRR*, abs/2110.02673, 2021.

Del Debbio, L., Rossney, J. M., and Wilson, M. Machine learning trivializing maps: A first step towards understanding how flow-based samplers scale up. *arXiv preprint arXiv:2112.15532*, 2021.

Finke, A. and Thiery, A. H. On importance-weighted autoencoders, 2019.

Geffner, T. and Domke, J. On the difficulty of unbiased alpha divergence minimization. *arXiv preprint arXiv:2010.09541*, 2020.

Geffner, T. and Domke, J. Empirical evaluation of biased methods for alpha divergence minimization. *arXiv preprint arXiv:2105.06587*, 2021.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., and Duvenaud, D. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019.

Kanwar, G. Machine learning and variational algorithms for lattice field theory. *arXiv preprint arXiv:2106.01975*, 2021.

Kanwar, G., Albergo, M. S., Boyda, D., Cranmer, K., Hackett, D. C., Racaniere, S., Rezende, D. J., and Shanahan, P. E. Equivariant flow-based sampling for lattice gauge theory. *Physical Review Letters*, 125(12):121601, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pp. 4743–4751, 2016.

Köhler, J., Klein, L., and Noé, F. Equivariant flows: Exact likelihood generative learning for symmetric densities. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5361–5370. PMLR, 2020.

Müller, T., McWilliams, B., Rousselle, F., Gross, M., and Novák, J. Neural importance sampling. *ACM Transactions on Graphics (TOG)*, 38(5):1–19, 2019.

Nicoli, K. A., Nakajima, S., Strodthoff, N., Samek, W., Müller, K.-R., and Kessel, P. Asymptotically unbiased estimation of physical observables with neural samplers. *Physical Review E*, 101(2):023304, 2020.

Nicoli, K. A., Anders, C. J., Funcke, L., Hartung, T., Jansen, K., Kessel, P., Nakajima, S., and Stornati, P. Estimation of thermodynamic observables in lattice field theories with deep generative models. *Phys. Rev. Lett.*, 126:032001, 2021.

Noé, F., Olsson, S., Köhler, J., and Wu, H. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457), 2019.

Nowozin, S. Debiasing evidence approximations: On importance-weighted autoencoders and jackknife variational inference. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Pontryagin, L. S. *Mathematical theory of optimal processes*. CRC press, 1987.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 1278–1286. JMLR.org, 2014.

Roeder, G., Wu, Y., and Duvenaud, D. Sticking the landing: Simple, lower-variance gradient estimators for variational inference. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6925–6934, 2017.

Tucker, G., Lawson, D., Gu, S., and Maddison, C. J. Doubly reparameterized gradient estimators for monte carlo objectives. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

van den Berg, R., Hasenclever, L., Tomczak, J. M., and Welling, M. Sylvester normalizing flows for variational inference. In Globerson, A. and Silva, R. (eds.), *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pp. 393–402. AUAI Press, 2018.

Wu, D., Wang, L., and Zhang, P. Solving statistical mechanics using variational autoregressive networks. *Physical review letters*, 122(8):080602, 2019.

A. Path-Gradients and Information Geometry

The Fisher information is defined by

$$\mathcal{I}(\theta)_{ij} = \mathbb{E}_{x\sim q_\theta}\left[s(\theta,x)_i\, s(\theta,x)_j\right],$$

where the score function is defined by

$$s(\theta,x)_i = \frac{\partial}{\partial\theta_i}\ln q_\theta(x)\,.$$

Using

$$\mathbb{E}_{z_1,\dots,z_N\sim q_Z}\left[\mathcal{G}_{\text{score}}\right] = 0\,,$$

the covariance of the score term $\mathcal{G}_{\text{score}}$ is given by

$$\mathrm{Cov}_{z^{(1)},\dots,z^{(N)}\sim q_Z}\left[\mathcal{G}_{\text{score}}\right]_{ij} = \mathbb{E}_{z^{(1)},\dots,z^{(N)}\sim q_Z}\left[\frac{1}{N}\sum_{l=1}^{N}\partial_{\theta_i}\ln q_\theta(x(z^{(l)}))\frac{1}{N}\sum_{k=1}^{N}\partial_{\theta_j}\ln q_\theta(x(z^{(k)}))\right]$$

We now use the fact that the samples z_i are independent and identically distributed, e.g. $\mathbb{E}_{z^{(i)},z^{(j)}\sim q_Z}[\bullet] = \mathbb{E}_{z^{(i)}\sim q_Z}[\bullet]\mathbb{E}_{z^{(j)}\sim q_Z}[\bullet]$ for $i\neq j$, to arrive at the final result

$$\begin{aligned}\mathrm{Cov}_{z^{(1)},\dots,z^{(N)}\sim q_Z}\left[\mathcal{G}_{\text{score}}\right]_{ij} &= \frac{1}{N}\sum_{l=1}^{N}\mathbb{E}_{z^{(l)}\sim q_Z}\left[\partial_{\theta_i}\ln q_\theta(x(z^{(l)}))\,\partial_{\theta_j}\ln q_\theta(x(z^{(l)}))\right]\\ &= \mathbb{E}_{z\sim q_Z}\left[\partial_{\theta_i}\ln q_\theta(x(z))\,\partial_{\theta_j}\ln q_\theta(x(z))\right]\\ &= \mathbb{E}_{x\sim q_\theta}\left[\partial_{\theta_i}\ln q_\theta(x)\,\partial_{\theta_j}\ln q_\theta(x)\right]\\ &= \mathbb{E}_{x\sim q_\theta}\left[s(\theta,x)_i\, s(\theta,x)_j)\right]\\ &= \mathcal{I}(\theta)_{ij}\,.\end{aligned}$$

B. Algorithms

Algorithm 1 Forw-Aug: Forward-mode derivative for path-wise gradient estimators for CNFs

```
Input: dynamics parameters $\theta$, start time 0, stop time 1, initial state $z_0$, loss gradient $\partial \ln q_0(z_0)/\partial z_0$
$s_0 = [z_0, \frac{\partial \ln q_0(z_0)}{\partial z_0}]$                                   # Define initial augmented state
def aug_dynamics($[z_t, \alpha(t), \cdot], t, \theta$):                        # Define dynamics on augmented state
    return $[f(z_t,t,\theta), -\alpha(t)\frac{\partial f}{\partial z_t} - \frac{\partial \mathrm{tr}(df/dz_t)}{\partial z_t}]$        # Compute vector-Jacobian products
$[z_1, \ln q_\theta(z_T), \frac{\partial \ln q_\theta(z_T)}{\partial z_T}]$ = ODESolve($s_0$, aug_dynamics, $0, T, \theta$)   # Solve reverse-time ODE
Return: $z_T, \ln q_\theta(z_T), \frac{\partial \ln q_\theta(z_T)}{\partial z_T}$                             # Return gradients
```

Algorithm 2 Full path gradient computation

```
Input: dynamics parameters $\theta$, start time 0, stop time 1
$z_0 \sim q_0$                                                    # Sampling from base distribution
$g_0 = \frac{\partial \ln q_0(z_0)}{\partial z_0}$                                           # Getting initial gradients
$z_T, \ln q_\theta(z_T), \frac{\partial \ln q_\theta(z_T)}{\partial z_T}$ = Forw-Aug($\theta, 0, T, z_0, \frac{\partial \ln q_0(z_0)}{\partial z_0}$)     # Applying Alg. 1
$\ln p(z_T), \frac{\partial \ln p(z_T)}{\partial z_T}$ = Compute log-prob and derivative         # Compute log-prob and derivative
$\_, \blacktriangledown_\theta \ln q_\theta(z_T)$ = Rev.-mode der.($\theta, T, 0, z_T, \frac{\partial \ln q_\theta(z_T)}{\partial z_T} - \frac{\partial \ln p(z_T)}{\partial z_T}$)   # Applying Alg. 1 from Chen et al. (2018)
Return: $z_T, \ln q_\theta(z_T), \blacktriangledown_\theta \ln q_\theta(z_T)$                         # Return gradients
```

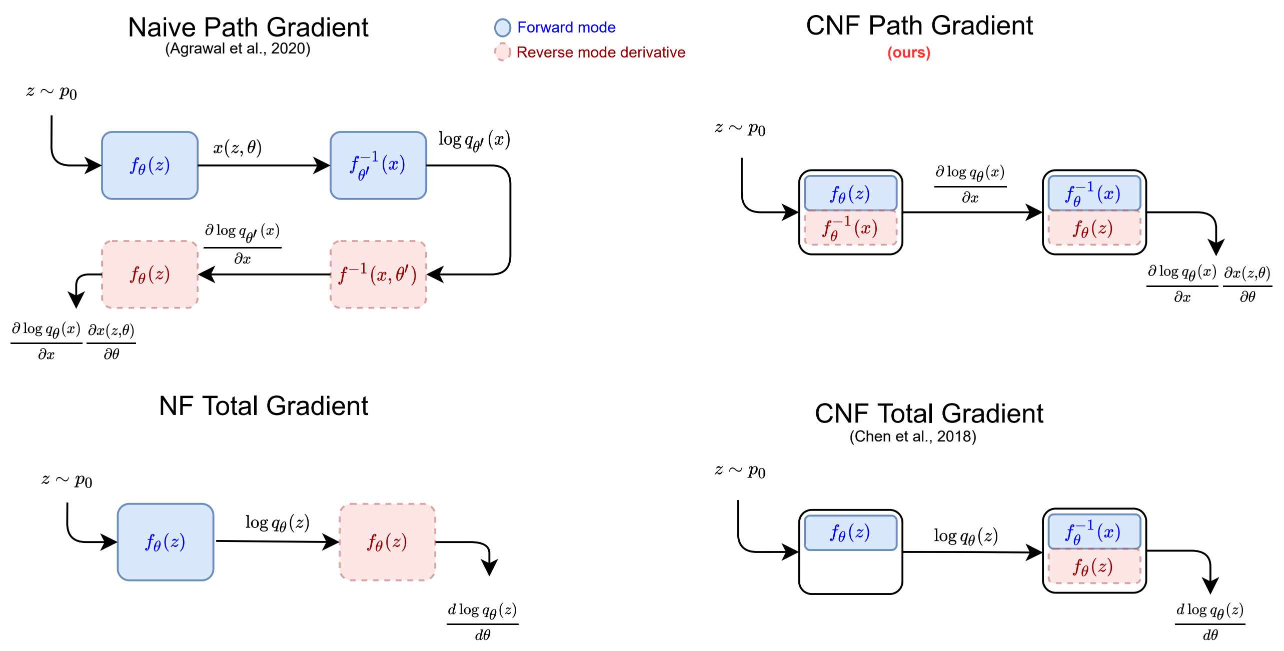

Figure 6. Comparing the gradient computation for discrete as well as continuous normalizing flows

C. Path gradient for CNFs

Theorem C.1. *The derivative* $\frac{\partial \ln q_\theta(z_T)}{\partial z_T}$ *can be obtained by solving the initial value problem*

$$\frac{d}{dt}\frac{\partial \ln q_\theta(z_t)}{\partial z_t} = -\frac{\partial \ln q_\theta(z_t)}{\partial z_t}^\intercal \frac{\partial f_\theta(z_t,t)}{\partial z_t} - \partial_{z_t} tr\left(\frac{df_\theta(z_t,t)}{dz_t}\right), \tag{13}$$

with initial condition

$$\frac{\partial \ln q_\theta(z_0)}{\partial z_0} = \frac{\partial \ln q_Z(z_0)}{\partial z_0}.$$

Proof. We define a generalized adjoint state by

$$\alpha_t \equiv \frac{d \ln q_\theta(z_t)}{dz_t} \tag{14}$$

Expanding the logarithm on the right-hand-side, we obtain

$$\alpha_t = \frac{d\left(\ln q_\theta(z_{t-\epsilon}) + \int_{t-\epsilon}^t \frac{\partial \ln q_\theta(z_t)}{\partial t}\mathrm{d}t\right)}{dz_t} = \frac{d\ln q_\theta(z_{t-\epsilon})}{dz_{t-\epsilon}}\frac{dz_{t-\epsilon}}{dz_t} + \frac{d\int_{t-\epsilon}^t \frac{\partial \ln q_\theta(z_t)}{\partial t}\mathrm{d}t}{dz_t}$$

Expanding the last summand up to linear order in ϵ, we obtain

$$\alpha_t = \frac{d\ln q_\theta(z_{t-\epsilon})}{dz_{t-\epsilon}}\frac{dz_{t-\epsilon}}{dz_t} + \frac{d\frac{\partial \ln q_\theta(z_t)}{\partial t}}{dz_t}\epsilon + \mathcal{O}(\epsilon^2) = \alpha_{t-\epsilon}\frac{dz_{t-\epsilon}}{dz_t} + \tilde{\alpha}_t\epsilon + \mathcal{O}(\epsilon^2),$$

where we have used the definition of the generalized adjoint state α_t (14) and have defined

$$\tilde{\alpha}_t = \frac{d\frac{\partial \ln q_\theta(z_t)}{\partial t}}{dz_t} = -\frac{\partial \text{tr}\left(\frac{df(t)}{dz_t}\right)}{\partial z_t} \,. \tag{15}$$

Using $z_{t-\epsilon} = z_t - \epsilon f(z_t, t, \theta) + \mathcal{O}(\epsilon^2)$, we conclude

$$\begin{aligned} \alpha_t &= \alpha_{t-\epsilon} \frac{d}{dz_t}\Big(z_t - \epsilon f(z_t, t, \theta)\Big) + \tilde{\alpha}_t \epsilon + \mathcal{O}(\epsilon^2) \\ &= \alpha_{t-\epsilon} - \epsilon \alpha_{t-\epsilon} \frac{\partial f(z_t, t, \theta)}{\partial z_t} + \epsilon \tilde{\alpha}_t + \mathcal{O}(\epsilon^2) \,. \end{aligned}$$

The definition of the derivative then implies the following ODE

$$\begin{aligned} \frac{d\alpha_t}{dt} &= \lim_{\epsilon \to 0} \frac{\alpha_t - \alpha_{t-\epsilon}}{\epsilon} \\ &= \lim_{\epsilon \to 0} \frac{-\epsilon \alpha_{t-\epsilon} \frac{\partial f(z_t, t, \theta)}{\partial z_t} + \tilde{\alpha}_t \epsilon}{\epsilon} \\ &= -\alpha_t \frac{\partial f(z_t, t, \theta)}{\partial z_t} + \tilde{\alpha}_t \,. \end{aligned}$$

From the definition of generalized adjoint state α_t (14) and of $\tilde{\alpha}_t$ (15), we arrive at the claim of the theorem

$$\frac{d}{dt} \frac{\partial \ln q_\theta(z_t)}{\partial z_t} = -\frac{d \ln q_\theta(z_t)}{dz_t} \frac{\partial f(z_t, t, \theta)}{\partial z_t} - \frac{\partial \text{tr}\left(\frac{df(t)}{dz_t}\right)}{\partial z_t} \,.$$

□

By evolving the generalized adjoint state α_t from $t = 0$ to $t = T$, we obtain the first term of the path gradient $\frac{\partial \ln q_\theta(x)}{\partial x} \frac{\partial x(z,\theta)}{\partial \theta}$. After computing this term in the forward pass, we can compute the whole path gradient as if the CNF were a standard Neural ODE, i.e. without taking the divergence term into account. The Hutchinson trace estimator can straightforwardly be applied to the trace independently of the adjoint state propagation.

D. Experimental details

D.1. Variational Autoencoder

To reproduce the experiments of Grathwohl et al. (2019), we implemented our proposed Algorithm 2 for the path gradient estimator in their codebase.

We train a VAE with an instance of Ffjord (Grathwohl et al., 2019) between the en- and decoder. As in the original experiments, the encoder network also outputs a low-rank update $\hat{U}(x)\hat{V}(x)^T$ to the global weight matrix W of the parameters of the flow as well as a bias vector $\hat{b}(x)$. Both VAE networks are seven-layer networks with the architecture in the experiments by van den Berg et al. (2018), using gated and transposed convolutions for the encoder and decoder respectively. For the baseline total gradient estimators, we used the results as reported in Grathwohl et al. (2019). For the largest datasets MNIST and Omniglot the hyper-parameters for the path gradient estimators, we searched close to the best hyper-parameters for the total gradient, since the full grid is quite expensive to search.

- MNIST: two subsequent CNFS each with a two hidden layer neural network with softplus, width 1024 and weight matrix updates of rank 64. Patience 35
- Omniglot: Five subsequent CNFs with a two hidden layer network with softplus, 512 hidden dimensions and weight matrix update of rank 64. Patience 35

For Caltech and Freyfaces, we used gridsearch over all the configurations which Grathwohl et al. (2019) also used for finding the best hyper-parameters.

Table 4. Negative test log likelihood (nll) for path and total gradient estimator. **Lower is better.** Per image, we used 5000 importance samples (2000 for Caltech) to estimate the nll.

Dataset	Path	Total
MNIST	$79.87 \pm .14$	$80.10 \pm .13$
Omniglot	$\mathbf{92.64} \pm .13$	$93.43 \pm .06$
Caltech Silhouettes	$\mathbf{91.65} \pm .01$	$93.06 \pm .02$
Frey Faces (nll bpd)	$4.28 \pm .03$	$4.36 \pm .06$

Table 5. Reverse KL divergence estimated as described in (Nicoli et al., 2021) using 4M samples. **Lower is better.**

Lattice size	Path	Total
12x12	$\mathbf{.0016} \pm .0005$	$.0113 \pm .0027$
20x20	$\mathbf{.0123} \pm .0008$	$.0415 \pm .0087$
32x32	$\mathbf{.0498} \pm .0049$	$.1833 \pm .0643$

- Caltech: Two subsequent CNF with a two hidden layer network with tanh, width 2048 hidden dimensions and weight matrix update of rank 20. Patience was 100.

- Freyfaces: Two CNFs with two hidden layers with softplus, width 512 and weight matrix update of rank 20. Patience was 100.

Training was done with a learning rate of .001, the Adam optimizer (Kingma & Ba, 2015), batch-size 100. The ODE solver was Dopri5. Training was done with a single GPU (A100).

While a single epoch generally took longer to train when using the path gradient estimator, the faster convergence (in epochs) meant that the training generally was comparable to training with the total gradient estimator. Due to the adaptive step size of the ODE solver, the duration of one forward step of the CNF varied from run to run. During the experiments, the addition of path gradient to the augmented dynamics did not have a marked effect on the number of function evaluations (NFE). For different random seeds for the same hyper-parameters, the path gradient runs the NFE fluctuated. This sometimes led to the effect, that an epoch with the path gradient estimator was quicker than with the total gradient. Due to the increased computation time per function evaluation however, the total gradient mostly used less wall-time per epoch.

Table 4 shows the negative test log likelihood for the path and total gradient estimator.

D.2. Lattice Field Theory

For the physics application we mirrored the experiments of de Haan et al. (2021). We only looked at lattice sizes where the effective sampling size could be improved, i.e. $L \in \{12, 20, 32\}$. For training we chose a batch size of 5000 with a learning rate of 0.0005 and the ADAM optimizer (Kingma & Ba, 2015). The ODE solver was Runge-Kutta 4 with 50 steps.
Using these hyperparameters we were able to reproduce and even improve the baseline, which was optimized using the total gradient estimator. Since the usage of the path gradient estimator increases the runtime, we let the baseline run for 20% more epochs than the path gradient estimator. In practice the quadratic complexity $\mathcal{O}(d^2)$ of the equivariant flow proposed by de Haan et al. (2021), the difference in computation time decreases with increasing dimensionality of the lattice.
For $L \in \{12, 20\}$ the baseline was run for 24k batches and the path gradient for 20k batches. For $L = 32$, since the models seem to have converged by then, we only trained for 15.4k and 12k respectively. Optimization for $L = 32$ showed some instable behavior for the baseline. For the reported results for the total gradient estimator, we only took runs into account, that did not diverge. Training was done with a single GPU (A100).

Since the energy of the LQFT's target density is known in closed-form (i.e. is not learnt from data), one can also use high-precision tools of theoretical physics, see (Nicoli et al., 2021), to estimate the reverse KL divergence directly. Analogously to the effective sampling size, the reverse KL divergence is consistently lower for the models trained with path-gradients, see Table 5.

E. Runtime Analysis

In order to analyse the additional computational cost of computing the path gradients as described in Alg. 2, we tested the increase in runtime over different number of dimensions. We fixed the parameters of the hidden layers in a vanilla CNF and recorded the time that each ”ODESolve” call took. This ignores e.g. the additional runtime, that duplicating the model yields in the naive path gradient estimator. Figure 4 shows the increase in runtime analysed with 24 repetitions. We can see that it yields an increase of roughly 14%. In practice the complexity of the model increases with the number of dimensions of the sample. Furthermore there are other operations that also scale with the problem complexity, so generally we can expect a lesser impact on the runtime than shown in Figure 4.