Diffusion normalizing flow

A paper by Qinsheng Zhang and Yongxin Chen
Georgia Institute of Technology

A presentation by Andrew Gracyk
University of Illinois Urbana-Champaign

Abstract

Diffusion normalizing flow is a deep generative modeling method in which information, particularly images, is diffused into noise in a reversible process. The noise may be propagated back so that the original image is reconstructed. Such a noise-inducing process is done by forward and backward neural stochastic differential equations.

Diffusion normalizing flow acts as a combination between diffusion models, i.e. injecting noise into data, and normalizing flow, i.e. a compositional sequence of invertible functions.

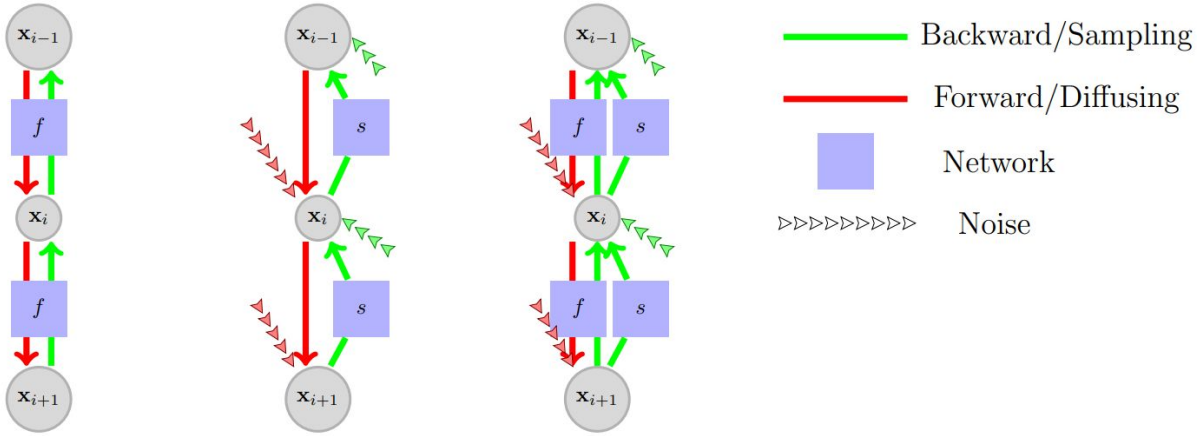
This method demonstrates competitive performance, and is especially strong for images with sharp boundaries.

Introduction

A class of deep generative modeling is done by injecting noise into images with a stochastic differential equation. A novel mathematical result implies this noise can be reversed to produce the image. These generative models can be classified as diffusion models.

This paper presents *DiffFlow* (diffusion normalizing flow), in which noise is added with a discretized sequence through trainable stochastic differential equations. It can be shown that normalizing flow is a special case of DiffFlow.

Introduction



Normalizing Flows Diffusion Models DiffFlow

Background

A ***normalizing flow*** is a solution to the differential equation

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t, \theta),$$

that evolves according to the equation

$$\frac{\partial \log p(\mathbf{x}(t))}{\partial t} = -\mathrm{tr}(\frac{\partial \mathbf{f}}{\partial \mathbf{x}})$$

(proof discussed later). Such an evolution can be made discrete, in which such a discrete trajectory is described by the equations

$$\mathbf{x}_i = F_i(\mathbf{x}_{i-1}, \theta), \quad \mathbf{x}_{i-1} = F_i^{-1}(\mathbf{x}_i, \theta)$$

Where we denote such a discrete trajectory as $\tau = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_N\}$. The flow follows the composition of functions $F = F_N \circ F_{N-1} \cdots F_2 \circ F_1$. Furthermore, the following equation is obeyed, using the change-of-variables formula and following the path:

$$\log p(\mathbf{x}_0) = \log p(\mathbf{x}_N) - \sum_{i=1}^{N} \log |\det(\frac{\partial F_i^{-1}(\mathbf{x}_i)}{\partial \mathbf{x}_i})|.$$

Background

A ***diffusion model*** is a stochastic differential equation (SDE) of the form

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w},$$

where $\mathbf{f} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ is a drift function (frequently, but not necessarily, linear), and $g : \mathbb{R} \to \mathbb{R}$ is a diffusion function (frequently constant).

It is a known result such an SDE has a backward process

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\mathbf{s}(\mathbf{x}, t, \theta)]dt + g(t)d\mathbf{w}.$$

In particular, we can perform the forward SDE on a segment of data, i.e., an image, in a noise-injecting process. The noise can be reversed with such a backwards SDE to restore the original image. This occurs when the neural network $\mathbf{s}(\mathbf{x}, t, \theta)$ coincides with score function (log density gradient w.r.t. space) of the forward probability flow.

We write the trajectory distributions as

$$p_F(\tau) = p_F(\mathbf{x}_0) \prod_{i=1}^{N} p_F(\mathbf{x}_i|\mathbf{x}_{i-1}), \quad p_B(\tau) = p_B(\mathbf{x}_T) \prod_{i=1}^{N} p_B(\mathbf{x}_{i-1}|\mathbf{x}_i).$$

Diffusion normalizing flow

We present ***diffusion normalizing flow*** (DiffFlow). This is essentially a diffusion model in which the drift term is learnable as a neural network. The forward SDE becomes

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t, \theta)dt + g(t)d\mathbf{w},$$

with a corresponding backwards SDE

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t, \theta) - g^2(t)\mathbf{s}(\mathbf{x}, t, \theta)]dt + g(t)d\mathbf{w}.$$

It is the goal of the final result at zero time of the backwards process to match the true distribution in the forward process.

To train DiffFlow, we minimize the KL divergence over the trajectory space τ.

Diffusion normalizing flow

We use the following numerical scheme as a means of implementation:

$$\begin{aligned} \mathbf{x}_{i+1} &= \mathbf{x}_i + \mathbf{f}_i(\mathbf{x}_i)\Delta t_i + g_i \delta_i^F \sqrt{\Delta t_i} \\ \mathbf{x}_i &= \mathbf{x}_{i+1} - [\mathbf{f}_{i+1}(\mathbf{x}_{i+1}) - g_{i+1}^2 \mathbf{s}_{i+1}(\mathbf{x}_{i+1})]\Delta t_i + g_{i+1}\delta_i^B \sqrt{\Delta t_i}, \end{aligned}$$

for Gaussian noise $\delta_i^F, \delta_i^B \sim \mathcal{N}(0, \mathbf{I})$. Neural networks are evaluated at discrete time locations given by $\{t_i\}_{i=0}^N$.

It is clear, using log properties and the forward and backward trajectory distributions, the KL loss to be minimized is given by

$$KL(p_F(\tau)|p_B(\tau)) = \underbrace{\mathbb{E}_{\tau\sim p_F}[\log p_F(\mathbf{x}_0)]}_{L_0} + \underbrace{\mathbb{E}_{\tau\sim p_F}[-\log p_B(\mathbf{x}_N)]}_{L_N} + \sum_{i=1}^{N-1} \underbrace{\mathbb{E}_{\tau\sim p_F}[\log \frac{p_F(\mathbf{x}_i|\mathbf{x}_{i-1})}{p_B(\mathbf{x}_{i-1}|\mathbf{x}_i)}]}_{L_i}.$$

Diffusion normalizing flow

Furthermore, it can be shown the backward Gaussian noise satisfies

$$\delta_i^B(\tau) = \frac{1}{g_{i+1}\sqrt{\Delta t}}\left[\mathbf{x}_i - \mathbf{x}_{i+1} + [\mathbf{f}_{i+1}(\mathbf{x}_{i+1}) - g_{i+1}^2 \mathbf{s}_{i+1}(\mathbf{x}_{i+1})]\Delta t\right].$$

In addition, we will show later that the final loss function in the training algorithm is given by

$$L := \mathbb{E}_{\tau\sim p_F}[-\log p_B(\mathbf{x}_N) + \sum_i \frac{1}{2}(\delta_i^B(\tau))^2] = \mathbb{E}_{\delta^F;\mathbf{x}_0\sim p_0}[-\log p_B(\mathbf{x}_N) + \sum_i \frac{1}{2}(\delta_i^B(\tau))^2].$$

Training

Algorithm 1 Training

repeat

$\mathbf{x}_0 \sim$ Real data distribution

$\delta^F_{1:N} \sim \mathcal{N}(0, \mathbf{I})$

Discrete timestamps: $t^N_{i=0}$

Sample: $\tau = \{\mathbf{x}_i\}^N_{i=0}$ based on $\delta^F_{1:N}$

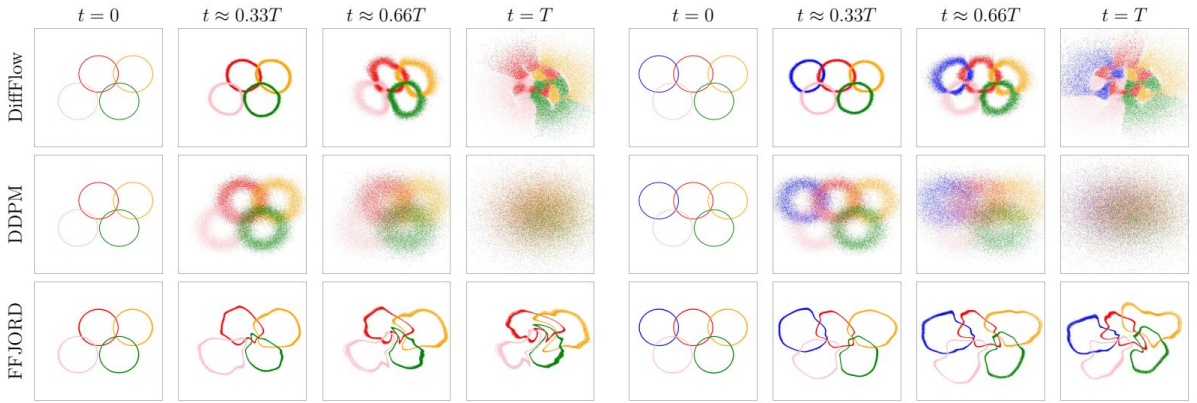
Gradient descent step $\nabla_\theta[-\log p_B(\mathbf{x}_N) + \sum_i \frac{1}{2}(\delta^B_i(\tau))^2]$

until converged

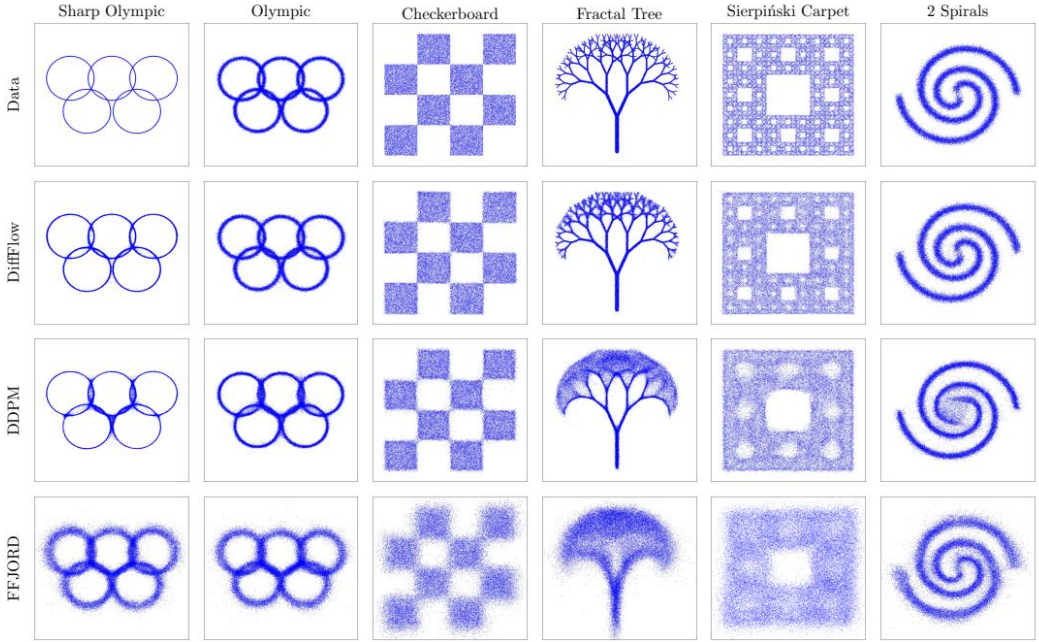
Theorem

Theorem 1. *As diffusion coefficients* $g_i \to 0$, *DiffFlow reduces to Normalizing Flow. Moreover, minimizing the objective function in Equation* (13) *is equivalent to minimizing the negative log-likelihood as in Equation* (4).

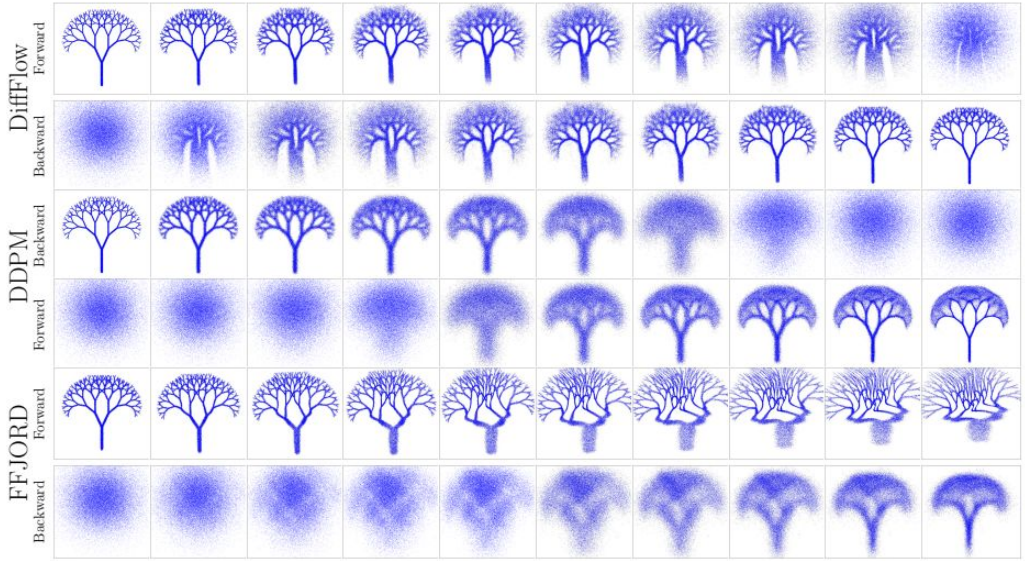
Results


$t = 0$
$t \approx 0.33T$
$t \approx 0.66T$
$t = T$
$t = 0$
$t \approx 0.33T$
$t \approx 0.66T$
$t = T$
DiffFlow
DDPM
FFJORD

Results


Sharp Olympic
Olympic
Checkerboard
Fractal Tree
Sierpiński Carpet
2 Spirals
Data
DiffFlow
DDPM
FFJORD

Results



DiffFlow is particularly strong for capturing images with sharp boundaries.

Proofs and derivations

First, we prove $\frac{\partial \log p(\mathbf{x}(t))}{\partial t} = -\text{tr}(\frac{\partial \mathbf{f}}{\partial \mathbf{x}})$.

The proof follows from the ***Neural Ordinary Differential Equations*** (Chen., R.T.Q., et al.) paper.

Theorem (Instantaneous Change of Variables). *Let $\mathbf{z}(t)$ be a finite continuous random variable with probability $p(\mathbf{z}(t))$ dependent on time. Let $\frac{d\mathbf{z}}{dt} = f(\mathbf{z}(t), t)$ be a differential equation describing a continuous-in-time transformation of $\mathbf{z}(t)$. Assuming that f is uniformly Lipschitz continuous in $\mathbf{z}$ and continuous in t, then the change in log probability also follows a differential equation:*

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\text{tr}\left(\frac{df}{d\mathbf{z}}(t)\right)$$

Proofs and derivations

Proof: We begin by denoting the solution over an infinitesimal change in time through a transformation

$$\mathbf{z}(t+\varepsilon) = T_\varepsilon(\mathbf{z}(t)).$$

Furthermore, we assume sufficiently nice conditions. Using the definition of the derivative, and the change of variables property (multiplying by Jacobian determinant)

$$\begin{aligned}
\frac{\partial \log p(\mathbf{z}(t))}{\partial t} &= \lim_{\varepsilon \to 0^+} \frac{\log p(\mathbf{z}(t)) - \log \left|\det \frac{\partial}{\partial \mathbf{z}} T_\varepsilon(\mathbf{z}(t))\right| - \log p(\mathbf{z}(t))}{\varepsilon} \\
&= - \lim_{\varepsilon \to 0^+} \frac{\log \left|\det \frac{\partial}{\partial \mathbf{z}} T_\varepsilon(\mathbf{z}(t))\right|}{\varepsilon} \\
&= - \lim_{\varepsilon \to 0^+} \frac{\frac{\partial}{\partial \varepsilon} \log \left|\det \frac{\partial}{\partial \mathbf{z}} T_\varepsilon(\mathbf{z}(t))\right|}{\frac{\partial}{\partial \varepsilon}\varepsilon} && \text{(by L'Hôpital's rule)} \\
&= - \lim_{\varepsilon \to 0^+} \frac{\frac{\partial}{\partial \varepsilon} \left|\det \frac{\partial}{\partial \mathbf{z}} T_\varepsilon(\mathbf{z}(t))\right|}{\left|\det \frac{\partial}{\partial \mathbf{z}} T_\varepsilon(\mathbf{z}(t))\right|} && \left(\left.\frac{\partial \log(\mathbf{z})}{\partial \mathbf{z}}\right|_{\mathbf{z}=1} = 1\right) \\
&= - \underbrace{\left(\lim_{\varepsilon \to 0^+} \frac{\partial}{\partial \varepsilon} \left|\det \frac{\partial}{\partial \mathbf{z}} T_\varepsilon(\mathbf{z}(t))\right|\right)}_{\text{bounded}} \underbrace{\left(\lim_{\varepsilon \to 0^+} \frac{1}{\left|\det \frac{\partial}{\partial \mathbf{z}} T_\varepsilon(\mathbf{z}(t))\right|}\right)}_{=1} = - \lim_{\varepsilon \to 0^+} \frac{\partial}{\partial \varepsilon} \left|\det \frac{\partial}{\partial \mathbf{z}} T_\varepsilon(\mathbf{z}(t))\right|
\end{aligned}$$

Proofs and derivations

It follows using the derivative of the determinant

$$\begin{aligned}
\frac{\partial \log p(\mathbf{z}(t))}{\partial t} &= -\lim_{\varepsilon \to 0^+} \operatorname{tr}\left(\operatorname{adj}\left(\frac{\partial}{\partial \mathbf{z}} T_\varepsilon(\mathbf{z}(t))\right) \frac{\partial}{\partial \varepsilon} \frac{\partial}{\partial \mathbf{z}} T_\varepsilon(\mathbf{z}(t))\right) \\
&= -\operatorname{tr}\left(\underbrace{\left(\lim_{\varepsilon \to 0^+} \operatorname{adj}\left(\frac{\partial}{\partial \mathbf{z}} T_\varepsilon(\mathbf{z}(t))\right)\right)}_{=I} \left(\lim_{\varepsilon \to 0^+} \frac{\partial}{\partial \varepsilon} \frac{\partial}{\partial \mathbf{z}} T_\varepsilon(\mathbf{z}(t))\right)\right) \\
&= -\operatorname{tr}\left(\lim_{\varepsilon \to 0^+} \frac{\partial}{\partial \varepsilon} \frac{\partial}{\partial \mathbf{z}} T_\varepsilon(\mathbf{z}(t))\right)
\end{aligned}$$

Proofs and derivations

and using Taylor expansions,

$$
\begin{aligned}
\frac{\partial \log p(\mathbf{z}(t))}{\partial t} &= -\mathrm{tr}\left(\lim_{\varepsilon \to 0^+} \frac{\partial}{\partial \varepsilon} \frac{\partial}{\partial \mathbf{z}} \left(\mathbf{z} + \varepsilon f(\mathbf{z}(t), t) + \mathcal{O}(\varepsilon^2) + \mathcal{O}(\varepsilon^3) + \dots\right)\right) \\
&= -\mathrm{tr}\left(\lim_{\varepsilon \to 0^+} \frac{\partial}{\partial \varepsilon} \left(I + \frac{\partial}{\partial \mathbf{z}} \varepsilon f(\mathbf{z}(t), t) + \mathcal{O}(\varepsilon^2) + \mathcal{O}(\varepsilon^3) + \dots\right)\right) \\
&= -\mathrm{tr}\left(\lim_{\varepsilon \to 0^+} \left(\frac{\partial}{\partial \mathbf{z}} f(\mathbf{z}(t), t) + \mathcal{O}(\varepsilon) + \mathcal{O}(\varepsilon^2) + \dots\right)\right) \\
&= -\mathrm{tr}\left(\frac{\partial}{\partial \mathbf{z}} f(\mathbf{z}(t), t)\right)
\end{aligned}
$$

□

Proofs and derivations

Next, we derive the DiffFlow loss function.

The KL divergence between forward and backward trajectories is given by

$$\begin{aligned} KL(p_F(\tau)|p_B(\tau)) &= KL(p_F(\mathbf{x}_0)p_F(\tau|\mathbf{x}_0)|p_B(\mathbf{x}_N)p_B(\tau|\mathbf{x}_N)) \\ &= \mathop{\mathbb{E}}_{p_F(\mathbf{x}_0)p_F(\tau|\mathbf{x}_0)}[\log p_F(\mathbf{x}_0) - \log p_B(\mathbf{x}_N) + \log \frac{p_F(\tau|\mathbf{x}_0)}{p_B(\tau|\mathbf{x}_N)}] \end{aligned}$$

where the first line is given by the conditional probability property, and the second by definition of KL divergence and log properties.

Proofs and derivations

Furthermore,

$$\frac{p_F(\tau|\mathbf{x}_0)}{p_B(\tau|\mathbf{x}_N)} = \frac{p_F(\mathbf{x}_N|\mathbf{x}_{N-1})\cdots p_F(\mathbf{x}_2|\mathbf{x}_1)p_F(\mathbf{x}_1|\mathbf{x}_0)}{p_B(\mathbf{x}_{N-1}|\mathbf{x}_N)\cdots p_B(\mathbf{x}_1|\mathbf{x}_2)p_B(\mathbf{x}_0|\mathbf{x}_1)}$$

which follows immediately using the trajectory distributions (discussed previously) and conditional probability property, and

$$\frac{p_F(\mathbf{x}_i|\mathbf{x}_{i-1})}{p_B(\mathbf{x}_{i-1}|\mathbf{x}_i)} = \frac{g_i\mathcal{N}(\delta_i^F|0,1)}{g_{i+1}\mathcal{N}(\delta_i^B|0,1)} \qquad \log\frac{p(\delta_i^F)}{p(\delta_i^B)} = \frac{1}{2}[(\delta_i^B)^2 - (\delta_i^F)^2],$$

(proof omitted), our loss function becomes

$$\begin{aligned} KL(p_F(\tau)|p_B(\tau)) &= -H(p_F(\mathbf{x}_0)) + \mathop{\mathbb{E}}_{\tau\sim p_F}[-\log p_B(\mathbf{x}_N) + \sum_{i=1}^{N}\log\frac{p_F(\mathbf{x}_i|\mathbf{x}_{i-1})}{p_B(\mathbf{x}_{i-1}|\mathbf{x}_i)}] \\ &= -H(p_F(\mathbf{x}_0)) + \mathop{\mathbb{E}}_{\tau\sim p_F}[-\log p_B(\mathbf{x}_N) + \log\frac{g_0}{g_N} + \sum_{i=1}^{N}\log\frac{p(\delta_i^F)}{p(\delta_i^B)}]. \end{aligned}$$

Proofs and derivations

The first and third terms are constant, so we aim to minimize empirically

$$\mathop{\mathbb{E}}_{\tau \sim p_F}[-\log p_B(\mathbf{x}_N) + \sum_{i=0}^{N} \frac{1}{2}((\delta_i^B)^2 - (\delta_i^F)^2)].$$

Expectation of the final term is constant, reducing the loss further to

$$\mathop{\mathbb{E}}_{\tau \sim p_F}[-\log p_B(\mathbf{x}_N) + \sum_{i=1}^{N} \frac{1}{2}(\delta_i^B)^2].$$

Setting the discrete processes equal to each other at a particular index, we have

$$\delta_i^B = \frac{g_i}{g_{i+1}}\delta_i^F - [\frac{f_{i+1} - f_i}{g_{i+1}} + g_{i+1}s_{i+1}]\sqrt{\Delta t_i}.$$

Marginal equivalent SDEs

We provide a marginal equivalent SDE to the primary backwards SDE. We show

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t, \theta) - \frac{1+\lambda^2}{2} g^2(t)\mathbf{s}(\mathbf{x}, t, \theta)]dt + \lambda g(t) d\mathbf{w}$$

is marginally equivalent to the DiffFlow SDE

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t, \theta) - g^2(t)\mathbf{s}(\mathbf{x}, t, \theta)]dt + g(t) d\mathbf{w}.$$

According to the Fokker-Planck equation, using the corresponding diffusion and drift functions, we have

$$\frac{\partial p_1(\mathbf{x}, t)}{\partial t} = -\sum_i \frac{\partial}{\partial x_i}\{[f_i(\mathbf{x}, t) - g^2(t)s_i(\mathbf{x}, t)]p_1(\mathbf{x}, t)\} - \frac{g^2(t)}{2}\sum_i \frac{\partial^2}{\partial x_i^2}[p_1(\mathbf{x}, t)]$$

for the DiffFlow SDE.

Marginal equivalent SDEs

Considering the second SDE again,

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t, \theta) - \frac{1+\lambda^2}{2} g^2(t)\mathbf{s}(\mathbf{x}, t, \theta)]dt + \lambda g(t) d\mathbf{w}$$

and denote

$$\hat{\mathbf{f}}(\mathbf{x}, t) = \mathbf{f}(\mathbf{x}, t) - \tfrac{1+\lambda^2}{2} g^2(t)\mathbf{s}(x, t).$$

Again, with the FKP equation, the distribution evolves according to

$$\frac{\partial p_2(\mathbf{x}, t)}{\partial t} = -\sum_i \frac{\partial}{\partial x_i}[\hat{f}_i(\mathbf{x}, t) p_2(\mathbf{x}, t)] - \frac{\lambda^2 g^2(t)}{2} \sum_i \frac{\partial^2}{\partial x_i^2}[p_2(\mathbf{x}, t)].$$

Observe the fact

$$\sum_i \frac{\partial^2}{\partial x_i^2}[p(\mathbf{x}, t)] = \sum_i \frac{\partial}{\partial x_i}\frac{\partial}{\partial x_i}[p(\mathbf{x}, t)] = \sum_i \frac{\partial}{\partial x_i}[p(\mathbf{x}, t)\frac{\partial \log p(\mathbf{x}, t)}{\partial x_i}].$$

Marginal equivalent SDEs

We have

$$
\begin{aligned}
&-\sum_i \frac{\partial}{\partial x_i}[\hat{f}_i(\mathbf{x},t)p_2(\mathbf{x},t)] \; - \frac{\lambda^2 g^2(t)}{2}\sum_i \frac{\partial^2}{\partial x_i^2}[p_2(\mathbf{x},t)] \\
&= -\sum_i \frac{\partial}{\partial x_i}[\hat{f}_i(\mathbf{x},t)p_2(\mathbf{x},t)] - \frac{(\lambda^2-1)g^2(t)}{2}\sum_i \frac{\partial}{\partial x_i}[p_2(\mathbf{x},t)\frac{\partial \log p_2(\mathbf{x},t)}{\partial x_i}] - \frac{g^2(t)}{2}\sum_i \frac{\partial^2}{\partial x_i^2}[p_2(\mathbf{x},t)] \\
&= -\sum_i \frac{\partial}{\partial x_i}\{[\hat{f}_i(\mathbf{x},t) + \frac{\lambda^2-1}{2}g^2\frac{\partial \log p_2(\mathbf{x},t)}{\partial x_i}]p_2(\mathbf{x},t)\} - \frac{g^2(t)}{2}\sum_i \frac{\partial^2}{\partial x_i^2}[p_2(\mathbf{x},t)] \\
&= -\sum_i \frac{\partial}{\partial x_i}\{\left[f_i(\mathbf{x},t) - g^2\frac{\partial \log p_2(\mathbf{x},t)}{\partial x_i}\right]p_2(\mathbf{x},t)\} - \frac{g^2(t)}{2}\sum_i \frac{\partial^2}{\partial x_i^2}[p_2(\mathbf{x},t)].
\end{aligned}
$$

Since the score corresponds to the gradient, we have the result.

References

Zhang, Q., and Chen, Y. *Diffusion normalizing flow*. https://arxiv.org/abs/2110.07579.

Chen., R.T.Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. *Neural ordinary differential equations*. https://arxiv.org/abs/1806.07366.