SDEs are all you need for latent variable models

Ole Winther

Department of Biology
University of Copenhagen (KU)

Dept for Applied Mathematics and Computer Science
Technical University of Denmark (DTU)

June 19, 2024

Motivation and outline

- Hierarchical VAEs are great, but today $T \to \infty$ layers are preferred.
- Formally we can start from finite T and take the limit but
- I want to understand the continuous limit modelling.

Motivation and outline

- Hierarchical VAEs are great, but today $T \to \infty$ layers are preferred.
- Formally we can start from finite T and take the limit but
- I want to understand the continuous limit modelling.

Roadmap

- The VAE and the Ladder VAE
- DDPM, VDM and DiffEnc (=diffusion with an encoder)

Motivation and outline

- Hierarchical VAEs are great, but today $T \to \infty$ layers are preferred.
- Formally we can start from finite T and take the limit but
- I want to understand the continuous limit modelling.

Roadmap

- The VAE and the Ladder VAE
- DDPM, VDM and DiffEnc (=diffusion with an encoder)
- Fokker-Planck - the master equation for the marginal distribution $p(\mathbf{z}(t);t)$

Motivation and outline

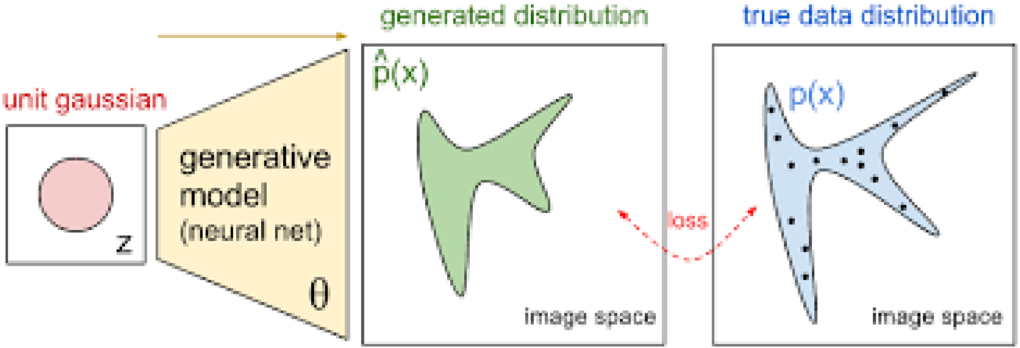
- Hierarchical VAEs are great, but today $T \to \infty$ layers are preferred.
- Formally we can start from finite T and take the limit but
- I want to understand the continuous limit modelling.

Roadmap

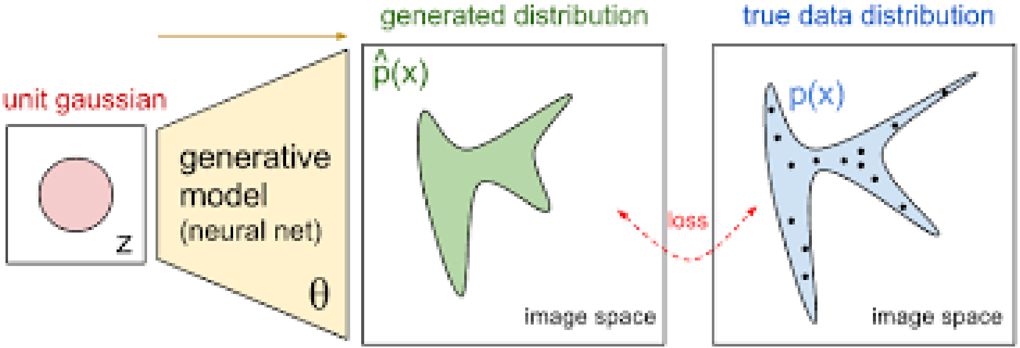
- The VAE and the Ladder VAE
- DDPM, VDM and DiffEnc (=diffusion with an encoder)
- Fokker-Planck - the master equation for the marginal distribution $p(\mathbf{z}(t); t)$
- SDE, ODE and reverse time SDE
- ODE and SDE for hierarchical generative models

Based upon Winther, Jeha and Dittadi, 2024, *in preparation*

Latent variable model - recap



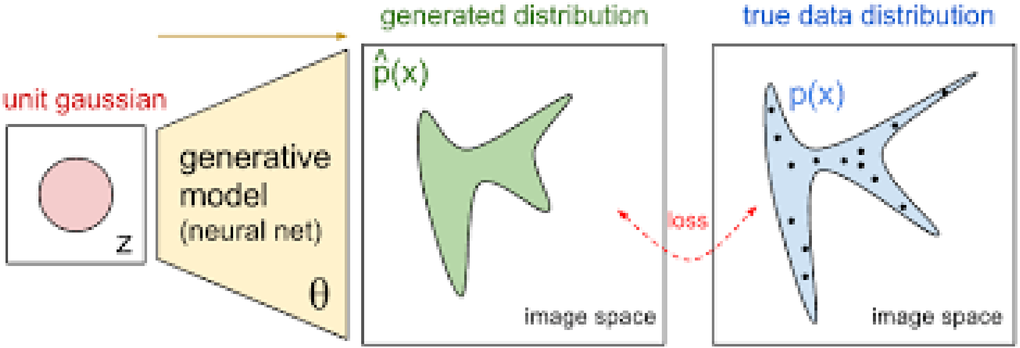
Latent variable model - recap



- Construct distribution $\hat{p}(\mathbf{x})$ from latent $\mathbf{z}$ + transformation:

$$\mathbf{z} \sim p(\mathbf{z})$$
$$\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})$$

Latent variable model - recap



- Construct distribution $\hat{p}(\mathbf{x})$ from latent $\mathbf{z}$ + transformation:

$$\mathbf{z} \sim p(\mathbf{z})$$
$$\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})$$

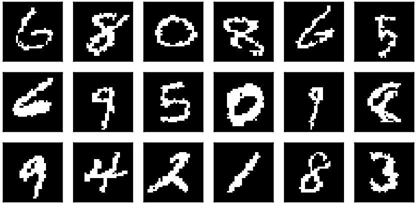
- Marginalisation:

$$\hat{p}(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

OpenAI Generative models blog

Illustrative example of a (deep) generative model - recap

Training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ binary MNIST



Generative model for $\mathbf{z} \in \mathbb{R}^2$ and $\mathbf{x} \in \{0,1\}^{28\times 28}$

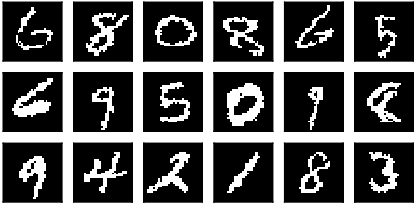
$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

Decoder network

$$f(\mathbf{z}) = \text{Sigmoid}(\mathbf{V} \tanh(\mathbf{W}\mathbf{z} + \mathbf{b}) + \boldsymbol{\beta})$$

Illustrative example of a (deep) generative model - recap

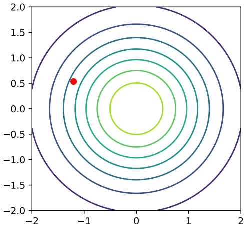
Training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ binary MNIST



Generation

$$\mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

$$\mathbf{z} = (-1.2033, 0.5340)$$



Generative model for $\mathbf{z} \in \mathbb{R}^2$ and $\mathbf{x} \in \{0,1\}^{28\times 28}$

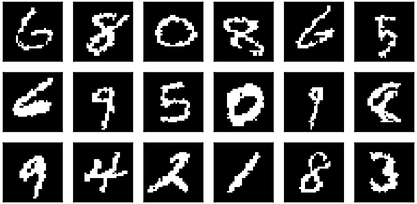
$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

Decoder network

$$f(\mathbf{z}) = \text{Sigmoid}(\mathbf{V} \tanh(\mathbf{W}\mathbf{z} + \mathbf{b}) + \boldsymbol{\beta})$$

Illustrative example of a (deep) generative model - recap

Training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ binary MNIST



Generative model for $\mathbf{z} \in \mathbb{R}^2$ and $\mathbf{x} \in \{0,1\}^{28\times 28}$

$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

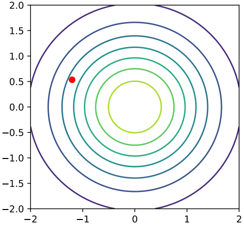
Decoder network

$$f(\mathbf{z}) = \text{Sigmoid}(\mathbf{V}\tanh(\mathbf{W}\mathbf{z} + \mathbf{b}) + \boldsymbol{\beta})$$

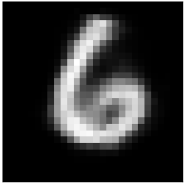
Generation

$$\mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

$$\mathbf{z} = (-1.2033, 0.5340)$$

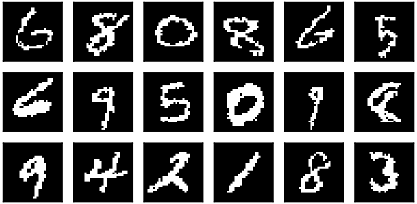


$f(\mathbf{z})$



Illustrative example of a (deep) generative model - recap

Training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ binary MNIST



Generative model for $\mathbf{z} \in \mathbb{R}^2$ and $\mathbf{x} \in \{0,1\}^{28\times 28}$

$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

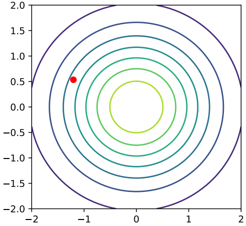
Decoder network

$$f(\mathbf{z}) = \text{Sigmoid}(\mathbf{V} \tanh(\mathbf{W}\mathbf{z} + \mathbf{b}) + \boldsymbol{\beta})$$

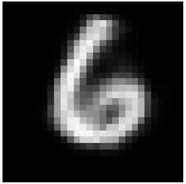
Generation

$$\mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

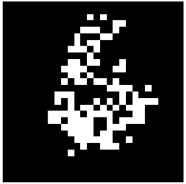
$$\mathbf{z} = (-1.2033, 0.5340)$$



$f(\mathbf{z})$

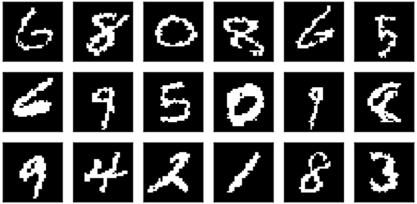


$x^{j,k} \sim \text{Bern}(f^{j,k}(\mathbf{z}))$



Illustrative example of a (deep) generative model - recap

Training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ binary MNIST



Generative model for $\mathbf{z} \in \mathbb{R}^2$ and $\mathbf{x} \in \{0,1\}^{28\times 28}$

$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

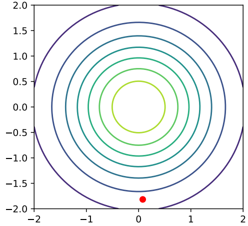
Decoder network

$$f(\mathbf{z}) = \text{Sigmoid}(\mathbf{V} \tanh(\mathbf{W}\mathbf{z} + \mathbf{b}) + \boldsymbol{\beta})$$

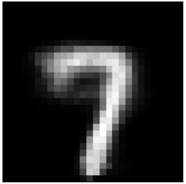
Generation

$$\mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

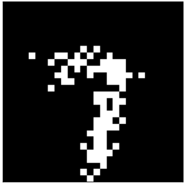
$$\mathbf{z} = (0.0791, -1.8165)$$



$f(z)$

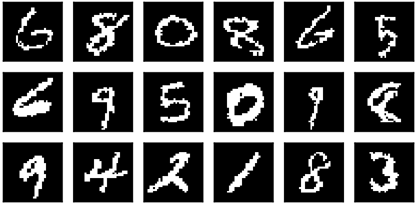


$x^{j,k} \sim \text{Bern}(f^{j,k}(z))$



Illustrative example of a (deep) generative model - recap

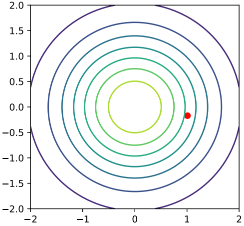
Training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ binary MNIST



Generation

$$\mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

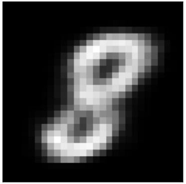
$$\mathbf{z} = (1.0097, -0.1711)$$



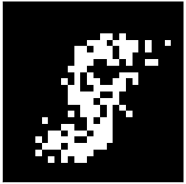
Generative model for $\mathbf{z} \in \mathbb{R}^2$ and $\mathbf{x} \in \{0, 1\}^{28 \times 28}$

$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

$f(z)$



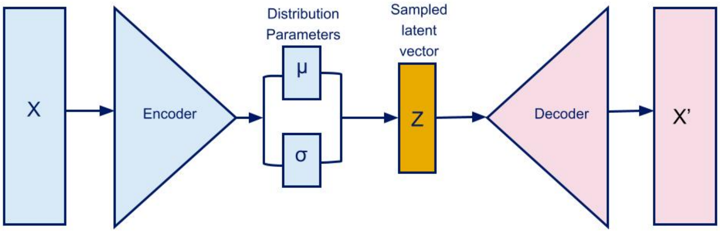
$x^{j,k} \sim \text{Bern}(f^{j,k}(z))$



Decoder network

$$f(\mathbf{z}) = \text{Sigmoid}(\mathbf{V} \tanh(\mathbf{W}\mathbf{z} + \mathbf{b}) + \boldsymbol{\beta})$$

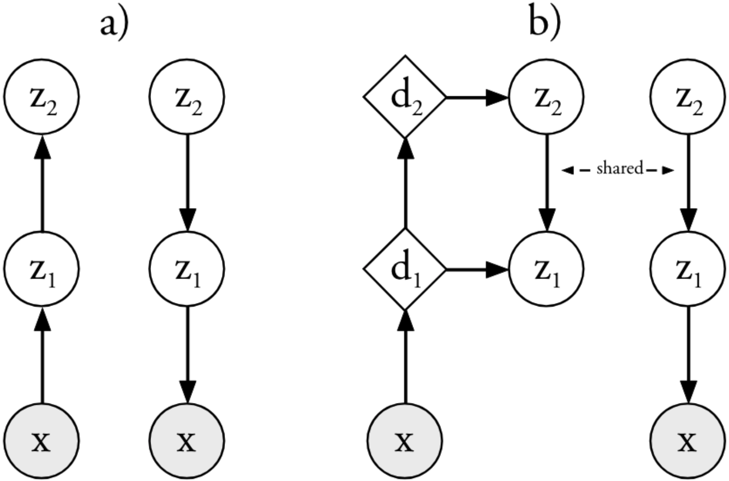
VAE - recap



- The log likelihood lower bound (aka the ELBO)

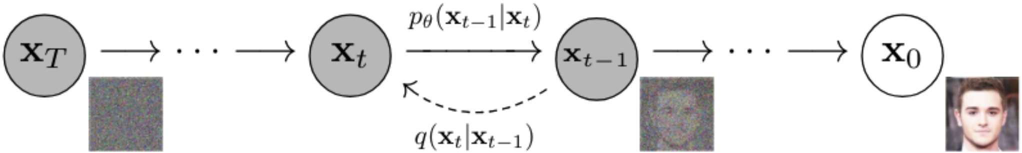
$$\log p(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right]$$

Hierarchical VAEs - recap



Ladder VAE, C. K. Sønderby et al, 2016

Diffusion models DDPM and VDM - recap



- May be viewed as a VAE with
  - simple noising encoder with marginal - $q(\mathbf{z}_t|\mathbf{x}) = \int q(\mathbf{z}_0, \ldots, \mathbf{z}_T|\mathbf{x})d\mathbf{z}_{-t}$:

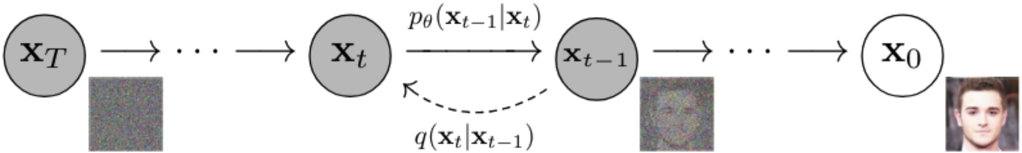
$$q(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}(\mathbf{z}_t|\alpha_t\mathbf{x}, \sigma_t^2\mathbf{I})$$

with

$$\mathrm{SNR}(t) = \frac{\alpha_t^2}{\sigma_t^2}$$

decreasing closer to the prior

Diffusion models DDPM and VDM - recap



- May be viewed as a VAE with
  - simple noising encoder with marginal - $q(\mathbf{z}_t|\mathbf{x}) = \int q(\mathbf{z}_0, \ldots, \mathbf{z}_T|\mathbf{x}) d\mathbf{z}_{-t}$:

    $$q(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}(\mathbf{z}_t|\alpha_t \mathbf{x}, \sigma_t^2 \mathbf{I})$$

    with

    $$\mathrm{SNR}(t) = \frac{\alpha_t^2}{\sigma_t^2}$$

    decreasing closer to the prior
  - generative model parameter sharing

    $$p(\mathbf{z}_{t-1}|\mathbf{z}_t) = \mathcal{N}(\mathbf{z}_{t-1}|\mu_\theta(\mathbf{z}_t, t), \Sigma_\theta(t)) \ .$$

DDPM, Ho et al, 2020, VDM, Kingma et al, 2021

VDM

- $T+1$ layers and indices $t, s \in [0,1]$: $s(i) = \frac{i-1}{T}$ and $t(i) = \frac{i}{T}$
- Hierarchical generative model

$$p_\theta(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z}_0)p(\mathbf{z}_1)\prod_{i=1}^{T} p_\theta(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)})$$

VDM

- $T+1$ layers and indices $t, s \in [0,1]$: $s(i) = \frac{i-1}{T}$ and $t(i) = \frac{i}{T}$
- Hierarchical generative model

$$p_\theta(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z}_0)p(\mathbf{z}_1)\prod_{i=1}^{T} p_\theta(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)})$$

- ELBO

$$\mathrm{ELBO}(\mathbf{x}) = \underbrace{\mathbb{E}_{q(\mathbf{z}_0|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z}_0)\right]}_{\text{reconstruction}} - \underbrace{\mathcal{D}_{\mathrm{KL}}(q(\mathbf{z}_1|\mathbf{x})||p(\mathbf{z}_1))}_{\text{prior}} -$$
$$\underbrace{\sum_{i=1}^{T} \mathbb{E}_{q(\mathbf{z}_{t(i)}|\mathbf{x})}\left[\mathcal{D}_{\mathrm{KL}}(q(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)}, \mathbf{x})\|p_\theta(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)}))\right]}_{\text{diffusion}}$$

VDM

- $T+1$ layers and indices $t, s \in [0,1]$: $s(i) = \frac{i-1}{T}$ and $t(i) = \frac{i}{T}$
- Hierarchical generative model

$$p_\theta(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z}_0)p(\mathbf{z}_1)\prod_{i=1}^{T} p_\theta(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)})$$

- ELBO

$$\mathrm{ELBO}(\mathbf{x}) = \underbrace{\mathbb{E}_{q(\mathbf{z}_0|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z}_0)\right]}_{\text{reconstruction}} - \underbrace{\mathcal{D}_{\mathrm{KL}}(q(\mathbf{z}_1|\mathbf{x})||p(\mathbf{z}_1))}_{\text{prior}} - \underbrace{\sum_{i=1}^{T} \mathbb{E}_{q(\mathbf{z}_{t(i)}|\mathbf{x})}\left[\mathcal{D}_{\mathrm{KL}}(q(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)}, \mathbf{x})\|p_\theta(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)}))\right]}_{\text{diffusion}}$$

- Use Bayes to reverse the direction of diffusion, $s < t$, for details VDM, Kingma et al, 2021

$$q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x}) = \mathcal{N}(\mathbf{z}_s|\mu_Q, \sigma_Q^2\mathbf{I})$$

VDM

- $T+1$ layers and indices $t, s \in [0,1]$: $s(i) = \frac{i-1}{T}$ and $t(i) = \frac{i}{T}$
- Hierarchical generative model

$$p_\theta(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z}_0)p(\mathbf{z}_1)\prod_{i=1}^{T} p_\theta(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)})$$

- ELBO

$$\mathrm{ELBO}(\mathbf{x}) = \underbrace{\mathbb{E}_{q(\mathbf{z}_0|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z}_0)\right]}_{\text{reconstruction}} - \underbrace{\mathcal{D}_{\mathrm{KL}}(q(\mathbf{z}_1|\mathbf{x})||p(\mathbf{z}_1))}_{\text{prior}} -$$
$$\underbrace{\sum_{i=1}^{T} \mathbb{E}_{q(\mathbf{z}_{t(i)}|\mathbf{x})}\left[\mathcal{D}_{\mathrm{KL}}(q(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)}, \mathbf{x})\|p_\theta(\mathbf{z}_{s(i)}|\mathbf{z}_{t(i)}))\right]}_{\text{diffusion}}$$

- Use Bayes to reverse the direction of diffusion, $s < t$, for details VDM, Kingma et al, 2021

$$q(\mathbf{z}_s|\mathbf{z}_t, \mathbf{x}) = \mathcal{N}(\mathbf{z}_s|\mu_Q, \sigma_Q^2\mathbf{I})$$

- We can choose the generative model to have the same functional form:

$$p(\mathbf{z}_s|\mathbf{z}_t) = \mathcal{N}(\mathbf{z}_s|\mu_P, \sigma_P^2\mathbf{I})$$

- Make even closer: $\mu_Q = \mu_Q(\mathbf{z}_t, \mathbf{x}_t, t)$, $\mu_P = \mu_Q(\mathbf{z}_t, \hat{\mathbf{x}}(\mathbf{z}_t, t), t)$ and $\sigma_P = \sigma_Q = \sigma_Q(t)$.

Diffusion loss and the $T \to \infty$ limit

- The diffusion term now becomes

$$-\sum_{i=1}^{T} \mathbb{E}_{q(\mathbf{z}_{t(i)}|\mathbf{x})} \left[\frac{1}{2\sigma_Q^2} \|\mu_P - \mu_Q\|_2^2 \right] = -\sum_{i=1}^{T} \frac{\Delta \mathrm{SNR}(\mathrm{t(i)})}{2} \mathbb{E}_{q(\mathbf{z}_{t(i)}|\mathbf{x})} \left[\|\hat{\mathbf{x}}(\mathbf{z}_{t(i)}, t(i)) - \mathbf{x}\|_2^2 \right]$$

with $\Delta \mathrm{SNR}(t) = \mathrm{SNR}(t) - \mathrm{SNR}(s)$.

Diffusion loss and the $T \to \infty$ limit

- The diffusion term now becomes

$$-\sum_{i=1}^{T} \mathbb{E}_{q(\mathbf{z}_{t(i)}|\mathbf{x})}\left[\frac{1}{2\sigma_Q^2}\|\mu_P - \mu_Q\|_2^2\right] = -\sum_{i=1}^{T} \frac{\Delta\mathrm{SNR}(\mathrm{t(i)})}{2}\mathbb{E}_{q(\mathbf{z}_{t(i)}|\mathbf{x})}\left[\|\hat{\mathbf{x}}(\mathbf{z}_{t(i)}, t(i)) - \mathbf{x}\|_2^2\right]$$

  with $\Delta\mathrm{SNR}(t) = \mathrm{SNR}(t) - \mathrm{SNR}(s)$.
- The $T \to \infty$ limit is well-defined with $\Delta t = t - s = 1/T$ and

$$-\frac{1}{2}\sum_{i=1}^{T} \frac{\Delta\mathrm{SNR}(t(i))}{\Delta t}\mathbb{E}_{q(\mathbf{z}_{t(i)}|\mathbf{x})}\left[\|\hat{\mathbf{x}}(\mathbf{z}_{t(i)}, t(i)) - \mathbf{x}\|_2^2\right]\Delta t \to$$

$$-\frac{1}{2}\int_0^1 \frac{d\mathrm{SNR}(t)}{dt}\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})}\left[\|\hat{\mathbf{x}}(\mathbf{z}_t, t) - \mathbf{x}\|_2^2\right] dt$$

  using

$$\sum_{i=1}^{T} \ldots \Delta t \to \int_0^1 \ldots dt \ \text{ for } \ T \to \infty \ .$$

DiffEnc - Can we make the encoder more powerful?

- Let the encoder be depth dependent - marginal:

$$q(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}(\mathbf{z}_t|\alpha_t \mathbf{x}_t, \sigma_t^2 \mathbf{I})$$

with $\mathbf{x}_t = F_\phi(\mathbf{x}, t)$ learned encoder.

DiffEnc - Can we make the encoder more powerful?

- Let the encoder be depth dependent - marginal:

$$q(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}(\mathbf{z}_t|\alpha_t \mathbf{x}_t, \sigma_t^2 \mathbf{I})$$

  with $\mathbf{x}_t = F_\phi(\mathbf{x}, t)$ learned encoder.

- Yes, we can with a small modification of the diffusion term:

$$-\frac{1}{2}\sum_{i=1}^{T} \frac{\Delta \mathrm{SNR}(t(i))}{\Delta t} \mathbb{E}_{q(\mathbf{z}_{t(i)}|\mathbf{x})} \left[\left\| \hat{\mathbf{x}}(\mathbf{z}_{t(i)}, t(i)) - \mathbf{x}_{t(i)} - \frac{\Delta \mathbf{x}_{t(i)}}{\Delta \log \mathrm{SNR}(t(i))} \right\|_2^2 \right] \Delta t \rightarrow$$

$$-\frac{1}{2}\int_0^1 \frac{d\mathrm{SNR}(t)}{dt} \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})} \left[\left\| \hat{\mathbf{x}}(\mathbf{z}_t, t) - \mathbf{x}_t - \frac{d\mathbf{x}_t}{d \log \mathrm{SNR}(t)} \right\|_2^2 \right] dt$$

  with $\Delta \mathbf{x}_t = \mathbf{x}_t - \mathbf{x}_s$.

DiffEnc - Can we make the encoder more powerful?

- Let the encoder be depth dependent - marginal:

$$q(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}(\mathbf{z}_t|\alpha_t \mathbf{x}_t, \sigma_t^2 \mathbf{I})$$

  with $\mathbf{x}_t = F_\phi(\mathbf{x}, t)$ learned encoder.

- Yes, we can with a small modification of the diffusion term:

$$-\frac{1}{2}\sum_{i=1}^{T} \frac{\Delta \mathrm{SNR}(t(i))}{\Delta t} \mathbb{E}_{q(\mathbf{z}_{t(i)}|\mathbf{x})} \left[\left\| \hat{\mathbf{x}}(\mathbf{z}_{t(i)}, t(i)) - \mathbf{x}_{t(i)} - \frac{\Delta \mathbf{x}_{t(i)}}{\Delta \log \mathrm{SNR}(t(i))} \right\|_2^2 \right] \Delta t \to$$

$$-\frac{1}{2}\int_0^1 \frac{d\mathrm{SNR}(t)}{dt} \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x})} \left[\left\| \hat{\mathbf{x}}(\mathbf{z}_t, t) - \mathbf{x}_t - \frac{d\mathbf{x}_t}{d \log \mathrm{SNR}(t)} \right\|_2^2 \right] dt$$

  with $\Delta \mathbf{x}_t = \mathbf{x}_t - \mathbf{x}_s$.

- Must be a better way to find the continuous limit! This is unsatisfactory!
- Use continuous time formulation to see if we can make formulation more flexible

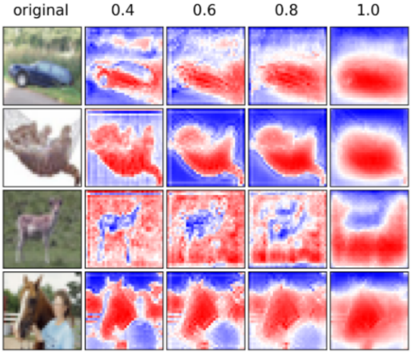
DiffEnc, Nielsen et al, ICLR, 2024

Encoder helps a bit

Model	Type	CIFAR-10	ImageNet 32×32
Flow Matching OT (Lipman et al., 2022)	Flow	2.99	3.53
Stochastic Int. (Albergo & Vanden-Eijnden, 2022)	Flow	2.99	3.48*
NVAE (Vahdat & Kautz, 2020)	VAE	2.91	3.92*
Image Transformer (Parmar et al., 2018)	AR	2.90	3.77*
VDVAE (Child, 2020)	VAE	2.87	3.80*
ScoreFlow (Song et al., 2021)	Diff	2.83	3.76*
Sparse Transformer (Child et al., 2019)	AR	2.80	–
Reflected Diffusion Models (Lou & Ermon, 2023)	Diff	2.68	3.74*
VDM (Kingma et al., 2021) *(10M steps)*	Diff	2.65	3.72*
ARDM (Hoogeboom et al., 2021)	AR	2.64	–
Flow Matching TN (Zheng et al., 2023)	Flow	2.60	3.45
Our experiments (8M and 1.5M steps, 3 seed avg)			
VDM with **v**-parameterization	Diff	2.64	3.46
DiffEnc Trainable (ours)	Diff	2.62	3.46

Encoder helps a bit

Model	Type	CIFAR-10	ImageNet 32×32
Flow Matching OT (Lipman et al., 2022)	Flow	2.99	3.53
Stochastic Int. (Albergo & Vanden-Eijnden, 2022)	Flow	2.99	3.48*
NVAE (Vahdat & Kautz, 2020)	VAE	2.91	3.92*
Image Transformer (Parmar et al., 2018)	AR	2.90	3.77*
VDVAE (Child, 2020)	VAE	2.87	3.80*
ScoreFlow (Song et al., 2021)	Diff	2.83	3.76*
Sparse Transformer (Child et al., 2019)	AR	2.80	–
Reflected Diffusion Models (Lou & Ermon, 2023)	Diff	2.68	3.74*
VDM (Kingma et al., 2021) *(10M steps)*	Diff	2.65	3.72*
ARDM (Hoogeboom et al., 2021)	AR	2.64	–
Flow Matching TN (Zheng et al., 2023)	Flow	2.60	3.45
Our experiments (8M and 1.5M steps, 3 seed avg)			
VDM with **v**-parameterization	Diff	2.64	3.46
DiffEnc Trainable (ours)	Diff	2.62	3.46



DiffEnc, Nielsen et al, ICLR, 2024 and more recently
Neural Flow Diffusion Models, Bartosh et al, 2024

Time dependent stochastic variables - setting the stage

- Let us learn how to formulate models directly in the continuous setting!

Time dependent stochastic variables - setting the stage

- Let us learn how to formulate models directly in the continuous setting!
- Alert: Change of notation:
  - Time dependent 1d stochastic variable $x(t)$ and data point y
  - $t \in [0, 1]$ with $t = 0$ "=" prior and $t = 1$ "=" data.

Time dependent stochastic variables - setting the stage

- Let us learn how to formulate models directly in the continuous setting!
- Alert: Change of notation:
  - Time dependent 1d stochastic variable $x(t)$ and data point y
  - $t \in [0, 1]$ with $t = 0$ "=" prior and $t = 1$ "=" data.
- We model time depedence in different ways, e.g. by a (learned) ordinary differential equation (ODE):

$$\frac{dx(t)}{dt} = V_0(x, t) \ ,$$

Time dependent stochastic variables - setting the stage

- Let us learn how to formulate models directly in the continuous setting!
- Alert: Change of notation:
  - Time dependent 1d stochastic variable $x(t)$ and data point y
  - $t \in [0, 1]$ with $t = 0$ "=" prior and $t = 1$ "=" data.
- We model time depedence in different ways, e.g. by a (learned) ordinary differential equation (ODE):

$$\frac{dx(t)}{dt} = V_0(x, t) \ ,$$

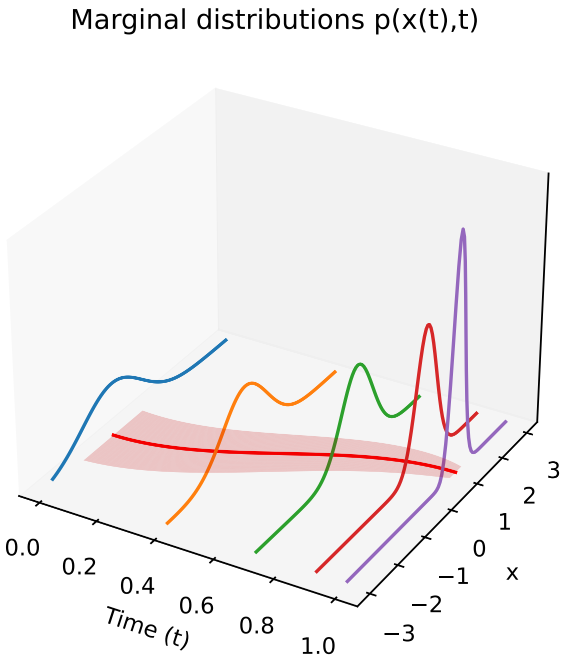
- Can also be a stochastic differential equation (SDE)

$$dx = a(x, t)dt + b(x, t)dW$$

- Interpolant

$$x(t) = t\, y_1 + (1 - t)\, y_0 + \gamma(t)\epsilon, \ \ \epsilon \sim \mathcal{N}(0, 1) \ .$$

The marginal



Fokker-Planck equation - setting the stage

- Marginal distribution at time t by $p(x(t), t)$
- $x(t)$ is the stochastic variable and t parameter: $\int p(x, t)dx = 1$.

Fokker-Planck equation - setting the stage

- Marginal distribution at time t by $p(x(t), t)$
- $x(t)$ is the stochastic variable and t parameter: $\int p(x,t)dx = 1$.
- It turns out that for all the systems we consider the time evolution of the marginal is governed by the Fokker-Planck (or Kolmogorov forward) partial differential equation

$$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[V(x,t)p(x,t)\right] + \left(\frac{\partial}{\partial x}\right)^2\left[D(x,t)p(x,t)\right] \ .$$

- $V(x,t)$ is the drift coefficient
- $D(x,t)$ is the diffusion coefficient

Fokker-Planck equation - setting the stage

- Marginal distribution at time t by $p(x(t),t)$
- $x(t)$ is the stochastic variable and t parameter: $\int p(x,t)dx = 1$.
- It turns out that for all the systems we consider the time evolution of the marginal is governed by the Fokker-Planck (or Kolmogorov forward) partial differential equation

$$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[V(x,t)p(x,t)\right] + \left(\frac{\partial}{\partial x}\right)^2\left[D(x,t)p(x,t)\right] \ .$$

- $V(x,t)$ is the drift coefficient
- $D(x,t)$ is the diffusion coefficient
- In the following we will see how to connect these quantities with the time evolution for $x(t)$

Fokker-Planck equation - setting the stage

- Marginal distribution at time t by $p(x(t), t)$
- $x(t)$ is the stochastic variable and t parameter: $\int p(x,t)dx = 1$.
- It turns out that for all the systems we consider the time evolution of the marginal is governed by the Fokker-Planck (or Kolmogorov forward) partial differential equation

$$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[V(x,t)p(x,t)\right] + \left(\frac{\partial}{\partial x}\right)^2\left[D(x,t)p(x,t)\right] \; .$$

- $V(x,t)$ is the drift coefficient
- $D(x,t)$ is the diffusion coefficient
- In the following we will see how to connect these quantities with the time evolution for $x(t)$
- Derivation: Use definition of partial derivative, integrate it over an arbitrary function $h(x)$, use conditional distribution, integration by parts and truncation of Taylor expansion to second order.

SDE and Wiener process

- SDE

$$dx = a(x,t)dt + b(x,t)dW$$

- Wiener process $W(t)$ with independent Gaussian increment: $dW \sim \mathcal{N}(0, dt)$

SDE and Wiener process

- SDE
$$dx = a(x,t)dt + b(x,t)dW$$
- Wiener process $W(t)$ with independent Gaussian increment: $dW \sim \mathcal{N}(0, dt)$
- Derivation of Fokker-Planck for SDE: Same as above and derive the stochastic process $h(x)$ where h is an arbitrary function:
$$\begin{aligned} dh &= h(x+dx) - h(x) \\ &= h'(x)dx + \frac{1}{2}h''(x)(dx)^2 + \mathcal{O}((dx)^3) \end{aligned}$$

SDE and Wiener process

- SDE
$$dx = a(x,t)dt + b(x,t)dW$$
- Wiener process $W(t)$ with independent Gaussian increment: $dW \sim \mathcal{N}(0, dt)$
- Derivation of Fokker-Planck for SDE: Same as above and derive the stochastic process $h(x)$ where h is an arbitrary function:
$$\begin{aligned} dh &= h(x+dx) - h(x) \\ &= h'(x)dx + \frac{1}{2}h''(x)(dx)^2 + \mathcal{O}((dx)^3) \end{aligned}$$
- Count powers: Drift $a(x,t)dt = \mathcal{O}(dt)$ and noise $b(x,t)dW = \mathcal{O}(\sqrt{dt})$
- Wiener process $(dW)^2 = \mathbb{E}[(dW)^2] + \text{Noise} = dt + \text{Noise}$, where $\text{Noise} = \mathcal{O}(dt)$.

SDE and Wiener process

- SDE
$$dx = a(x,t)dt + b(x,t)dW$$
- Wiener process $W(t)$ with independent Gaussian increment: $dW \sim \mathcal{N}(0, dt)$
- Derivation of Fokker-Planck for SDE: Same as above and derive the stochastic process $h(x)$ where h is an arbitrary function:
$$\begin{aligned} dh &= h(x+dx) - h(x) \\ &= h'(x)dx + \frac{1}{2}h''(x)(dx)^2 + \mathcal{O}((dx)^3) \end{aligned}$$
- Count powers: Drift $a(x,t)dt = \mathcal{O}(dt)$ and noise $b(x,t)dW = \mathcal{O}(\sqrt{dt})$
- Wiener process $(dW)^2 = \mathbb{E}[(dW)^2] + \mathrm{Noise} = dt + \mathrm{Noise}$, where $\mathrm{Noise} = \mathcal{O}(dt)$.
- In infinitesimal limit keep lowest power only (Ito's lemma):
$$dh = \left(h'(x)a(x,t) + \frac{1}{2}h''(x)b(x,t) \right) dt + h'(x)b(x,t)dW$$

SDE and Wiener process

- SDE

$$dx = a(x,t)dt + b(x,t)dW$$

- Wiener process $W(t)$ with independent Gaussian increment: $dW \sim \mathcal{N}(0, dt)$
- Derivation of Fokker-Planck for SDE: Same as above and derive the stochastic process $h(x)$ where h is an arbitrary function:

$$\begin{aligned} dh &= h(x+dx) - h(x) \\ &= h'(x)dx + \frac{1}{2}h''(x)(dx)^2 + \mathcal{O}((dx)^3) \end{aligned}$$

- Count powers: Drift $a(x,t)dt = \mathcal{O}(dt)$ and noise $b(x,t)dW = \mathcal{O}(\sqrt{dt})$
- Wiener process $(dW)^2 = \mathbb{E}[(dW)^2] + \mathrm{Noise} = dt + \mathrm{Noise}$, where $\mathrm{Noise} = \mathcal{O}(dt)$.
- In infinitesimal limit keep lowest power only (Ito's lemma):

$$dh = \left(h'(x)a(x,t) + \frac{1}{2}h''(x)b(x,t) \right) dt + h'(x)b(x,t)dW$$

- Fokker-Planck for SDE

$$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[a(x,t)p(x,t)\right] + \frac{1}{2}\left(\frac{\partial}{\partial x}\right)^2 \left[b^2(x,t)p(x,t)\right] \; .$$

- Drift: $V(x,t) = a(x,t)$ and diffusion: $D(x,t) = \frac{b^2(x,t)}{2}$.

Shifting diffusion into drift

- Fokker-Planck again:

$$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[V(x,t)p(x,t)\right] + \left(\frac{\partial}{\partial x}\right)^2 \left[D(x,t)p(x,t)\right] \ .$$

- We can shift the diffusion into the drift

$$\left(\frac{\partial}{\partial x}\right)^2 \left[D(x)p(x,t)\right] = \frac{\partial}{\partial x}\left[\frac{\partial D(x)}{\partial x}p(x,t) + D(x)\frac{\partial \log p(x,t)}{\partial x}p(x,t)\right] \ .$$

- using $\frac{\partial p}{\partial x} = p\frac{\partial \log p}{\partial x}$.

Shifting diffusion into drift

- Fokker-Planck again:

$$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[V(x,t)p(x,t)\right] + \left(\frac{\partial}{\partial x}\right)^2 \left[D(x,t)p(x,t)\right] \ .$$

- We can shift the diffusion into the drift

$$\left(\frac{\partial}{\partial x}\right)^2 \left[D(x)p(x,t)\right] = \frac{\partial}{\partial x}\left[\frac{\partial D(x)}{\partial x}p(x,t) + D(x)\frac{\partial \log p(x,t)}{\partial x}p(x,t)\right] \ .$$

- using $\frac{\partial p}{\partial x} = p\frac{\partial \log p}{\partial x}$.
- Define

$$V_D(x,t) \equiv \frac{\partial D(x)}{\partial x} + D(x)\frac{\partial \log p(x,t)}{\partial x}$$

- Fokker-Planck again:

$$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[V_k(x,t)p(x,t)\right] + k\left(\frac{\partial}{\partial x}\right)^2 \left[D(x)p(x,t)\right] \tag{1}$$

$$V_k(x,t) = V_0(x) + kV_D(x,t) \ , \tag{2}$$

- Any choice of k will give the same model.

SDE, ODE and reverse time SDE

Probability	State
$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[V_1(x,t)p(x,t)\right] + \left(\frac{\partial}{\partial x}\right)^2 \left[D(x,t)p(x,t)\right]$ $V_k(x,t) = V_0(x,t) + kV_D(x,t)$ $V_D(x,t) = \frac{\partial D(x,t)}{\partial x} + D(x,t)\frac{\partial \log p(x,t)}{\partial x}$	$dx = V_1(x)dt + b(x,t)dW$
$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[V_0(x,t)p(x,t)\right]$	$dx = V_0(x,t)dt$
$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[V_{-1}(x,t)p(x,t)\right] - \left(\frac{\partial}{\partial x}\right)^2 \left[D(x,t)p(x,t)\right]$	$dx = V_{-1}(x,t)dt + b(x,t)d\overline{W}$ $dt < 0$

- Diffusion coefficient: $D(x,t) = b^2(x,t)/2$.
- $W(t)$ Wiener process
- $\overline{W}(t)$ reverse time Wiener process, $dt < 0$ increments independent (no proof)

SDE, ODE and reverse time SDE

Probability	State
$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[V_1(x,t)p(x,t)\right] + \left(\frac{\partial}{\partial x}\right)^2 \left[D(x,t)p(x,t)\right]$ $V_k(x,t) = V_0(x,t) + kV_D(x,t)$ $V_D(x,t) = \frac{\partial D(x,t)}{\partial x} + D(x,t)\frac{\partial \log p(x,t)}{\partial x}$	$dx = V_1(x)dt + b(x,t)dW$
$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[V_0(x,t)p(x,t)\right]$	$dx = V_0(x,t)dt$
$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[V_{-1}(x,t)p(x,t)\right] - \left(\frac{\partial}{\partial x}\right)^2 \left[D(x,t)p(x,t)\right]$	$dx = V_{-1}(x,t)dt + b(x,t)d\overline{W}$ $dt < 0$

- Diffusion coefficient: $D(x,t) = b^2(x,t)/2$.
- $W(t)$ Wiener process
- $\overline{W}(t)$ reverse time Wiener process, $dt < 0$ increments independent (no proof)
- Reverse time diffusion, Anderson, 1982 - prove that $x(t)$ and $\overline{W}(s)$, $s \leq t$ are independent with $\overline{W}(t)$ defined by

$$d\overline{W} = \frac{1}{p(x,t)}\frac{\partial}{\partial x}\left[b(x,t)p(x,t)\right]dt + dW$$

- The we can have run time in reverse, $dt < 0$ and have independent increments $d\overline{W}$.

ODE - Generative model

- Likelihood $p(y|x(1))$ and prior $p(x, 0)$
- (Neural) ODE, Chen et al, 2018:

$$dx = f_0(x, t)dt$$

ODE - Generative model

- Likelihood $p(y|x(1))$ and prior $p(x, 0)$
- (Neural) ODE, Chen et al, 2018:

$$dx = f_0(x, t)dt$$

- Fokker-Planck without diffusion (aka Liouville)

$$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[f_0(x,t)p(x,t)\right]$$

- Full derivative

$$\begin{aligned}
\frac{dp(x,t)}{dt} &= \frac{\partial p(x,t)}{\partial t} + \frac{\partial p(x,t)}{\partial x}\frac{dx}{dt} \\
&= -\frac{\partial}{\partial x}\left[f_0(x,t)p(x,t)\right] + \frac{\partial p(x,t)}{\partial x}\frac{dx}{dt} \\
&= -\frac{\partial f_0(x,t)}{\partial x}p(x,t) + \left(\frac{dx}{dt} - f_0(x,t)\right) \\
&= -\frac{\partial f_0(x,t)}{\partial x}p(x,t)
\end{aligned}$$

ODE - Generative model

- Likelihood $p(y|x(1))$ and prior $p(x,0)$
- (Neural) ODE, Chen et al, 2018:

$$dx = f_0(x,t)dt$$

- Fokker-Planck without diffusion (aka Liouville)

$$\frac{\partial p(x,t)}{\partial t} = -\frac{\partial}{\partial x}\left[f_0(x,t)p(x,t)\right]$$

- Full derivative

$$\begin{aligned}\frac{dp(x,t)}{dt} &= \frac{\partial p(x,t)}{\partial t} + \frac{\partial p(x,t)}{\partial x}\frac{dx}{dt} \\ &= -\frac{\partial}{\partial x}\left[f_0(x,t)p(x,t)\right] + \frac{\partial p(x,t)}{\partial x}\frac{dx}{dt} \\ &= -\frac{\partial f_0(x,t)}{\partial x}p(x,t) + \left(\frac{dx}{dt} - f_0(x,t)\right) \\ &= -\frac{\partial f_0(x,t)}{\partial x}p(x,t)\end{aligned}$$

- Solution using $\frac{dp}{dt} = p\frac{d\log p}{dt}$ and $x(t)$, $t \in [0,1]$ being the solution to the ODE:

$$\log p(x(1),1) = \log p(x(0),0) - \int_0^1 \frac{\partial f_0(x(t),t)}{\partial x(t)}dt$$

SDE - Generative model

- Generative model

$$p : \qquad dx = f_1(x,t)dt + b(x,t)dW \ .$$

- Variational distribution

$$q : \qquad dx = g_1(x,y,t)dt + b(x,t)dW' \ .$$

- Same noise $b(x,t)$ required (no proof) Archambeau et al, 2007

SDE - Generative model

- Generative model

$$p: \qquad dx = f_1(x,t)dt + b(x,t)dW \ .$$

- Variational distribution

$$q: \qquad dx = g_1(x,y,t)dt + b(x,t)dW' \ .$$

- Same noise $b(x,t)$ required (no proof) Archambeau et al, 2007
- Let us compute the ELBO - continuous version Song et al, NeurIPS, 2021
- Discretization $T+1$ "layers", $i = 0, \dots, T$ setting $t(i) = \frac{i}{T}$:

$$\Delta x_t = x_{t+\Delta t} - x_t = \underbrace{f_1(x(t),t)}_{\equiv f_1(t)} \Delta t + \underbrace{b(x(t),t)}_{\equiv b(t)} \Delta W_i$$

$$\Delta x_t = x_{t+\Delta t} - x_t = g_1(x(t),y,t)\Delta t + b(x(t),t)\Delta W_i' \ .$$

SDE - Generative model

- Generative model

$$p: \qquad dx = f_1(x,t)dt + b(x,t)dW \ .$$

- Variational distribution

$$q: \qquad dx = g_1(x,y,t)dt + b(x,t)dW' \ .$$

- Same noise $b(x,t)$ required (no proof) Archambeau et al, 2007
- Let us compute the ELBO - continuous version Song et al, NeurIPS, 2021
- Discretization $T+1$ "layers", $i = 0, \ldots, T$ setting $t(i) = \frac{i}{T}$:

$$\Delta x_t = x_{t+\Delta t} - x_t = \underbrace{f_1(x(t),t)}_{\equiv f_1(t)} \Delta t + \underbrace{b(x(t),t)}_{\equiv b(t)} \Delta W_i$$

$$\Delta x_t = x_{t+\Delta t} - x_t = g_1(x(t),y,t)\Delta t + b(x(t),t)\Delta W_i' \ .$$

- Gaussian conditionals (changing the notation slightly here):

$$p(\mathbf{x}) = p(x_0) \prod_{i=0}^{T-1} p(x_{t(i+1)}|x_{t(i)}) = p(x_0) \prod_{i=0}^{T-1} \mathcal{N}(x_{t(i+1)}|x_{t(i)} + f_1(t(i))\Delta t, b^2(t(i))\Delta t)$$

$$q(\mathbf{x}|y) = q(x_0|y) \prod_{i=0}^{T-1} q(x_{t(i+1)}|x_{t(i)}, y) = q(x_0|y) \prod_{i=0}^{T-1} \mathcal{N}(x_{t(i+1)}|x_{t(i)} + g_1(t(i))\Delta t, b^2(t(i))\Delta t)$$

SDE - Generative model

- $\mathrm{ELBO}(y) =$

$$\mathbb{E}_{q(\mathbf{x}|y)}\left[\log \frac{p(y|x_1)p(\mathbf{x})}{q(\mathbf{x}|y)}\right] = -\mathcal{D}_{\mathrm{KL}}(q(\mathbf{x}|y)||p(\mathbf{x})) + \mathbb{E}_{q(x_1|y)}\left[\log p(y|x_1)\right]$$

$$\begin{aligned}\mathcal{D}_{\mathrm{KL}}(q(\mathbf{x}|y), p(\mathbf{x})) =& \mathcal{D}_{\mathrm{KL}}(q(x_0|y), p(x_0)) + \\ & \sum_{i=0}^{T-1} \int q(x_{t(i)}|y)\mathcal{D}_{\mathrm{KL}}(q(x_{t(i+1)}|x_{t(i)}, y), p(x_{t(i+1)}|x_{t(i)}))dx_{t(i)}\end{aligned}$$

SDE - Generative model

- $\mathrm{ELBO}(y) =$

$$\mathbb{E}_{q(\mathbf{x}|y)}\left[\log \frac{p(y|x_1)p(\mathbf{x})}{q(\mathbf{x}|y)}\right] = -\mathcal{D}_{\mathrm{KL}}(q(\mathbf{x}|y)||p(\mathbf{x})) + \mathbb{E}_{q(x_1|y)}\left[\log p(y|x_1)\right]$$

$$\mathcal{D}_{\mathrm{KL}}(q(\mathbf{x}|y), p(\mathbf{x})) = \mathcal{D}_{\mathrm{KL}}(q(x_0|y), p(x_0)) + \sum_{i=0}^{T-1} \int q(x_{t(i)}|y)\mathcal{D}_{\mathrm{KL}}(q(x_{t(i+1)}|x_{t(i)}, y), p(x_{t(i+1)}|x_{t(i)}))dx_{t(i)}$$

- Tractable KL-divergence between Gaussian distributions:

$$\mathcal{D}_{\mathrm{KL}}(q(x_{t(i+1)}|x_{t(i)}, y), p(x_{t(i+1)}|x_{t(i)})) = \frac{||f_1(t(i)) - g_1(t(i))||^2}{2b^2(t(i))}\Delta t \ .$$

SDE - Generative model

- $\mathrm{ELBO}(y) =$

$$\mathbb{E}_{q(\mathbf{x}|y)}\left[\log \frac{p(y|x_1)p(\mathbf{x})}{q(\mathbf{x}|y)}\right] = -\mathcal{D}_{\mathrm{KL}}(q(\mathbf{x}|y)||p(\mathbf{x})) + \mathbb{E}_{q(x_1|y)}\left[\log p(y|x_1)\right]$$

$$\mathcal{D}_{\mathrm{KL}}(q(\mathbf{x}|y), p(\mathbf{x})) = \mathcal{D}_{\mathrm{KL}}(q(x_0|y), p(x_0)) + \sum_{i=0}^{T-1} \int q(x_{t(i)}|y)\mathcal{D}_{\mathrm{KL}}(q(x_{t(i+1)}|x_{t(i)}, y), p(x_{t(i+1)}|x_{t(i)}))dx_{t(i)}$$

- Tractable KL-divergence between Gaussian distributions:

$$\mathcal{D}_{\mathrm{KL}}(q(x_{t(i+1)}|x_{t(i)}, y), p(x_{t(i+1)}|x_{t(i)})) = \frac{||f_1(t(i)) - g_1(t(i))||^2}{2b^2(t(i))}\Delta t\ .$$

- Take $T \to \infty$ limit and changing notation again: $p(x_t) \to p(x, t)$ for marginal:

$$\mathrm{ELBO}(y) = \underbrace{-\mathcal{D}_{\mathrm{KL}}(q(x, 0|y), p(x, 0))}_{\text{Prior}} + \underbrace{\mathbb{E}_{q(x,1|y)}\left[\log p(y|x)\right]}_{\text{Likelihood}} \underbrace{- \int_0^1 \mathbb{E}_{q(x,t|y)}\left[\frac{||f_1(x,t) - g_1(x,y,t)||^2}{2b^2(x,t)}\right] dt}_{\text{Diffusion}}\ .$$

Closer look at the diffusion term and requirements on $q(x, t)$

- Diffusion term

$$-\int_0^1 \mathbb{E}_{q(x,t|y)} \left[\frac{||f_1(x,t) - g_1(x,y,t)||^2}{2b^2(x,t)} \right] dt$$

- For efficient computation we need to be able to sample $q(x, t|y)$

Closer look at the diffusion term and requirements on $q(x, t)$

- Diffusion term

$$-\int_0^1 \mathbb{E}_{q(x,t|y)} \left[\frac{||f_1(x,t) - g_1(x,y,t)||^2}{2b^2(x,t)} \right] dt$$

- For efficient computation we need to be able to sample $q(x, t|y)$
- Distributions defined as flows and mixtures of flows permit this (proof not given)
- Flows include Gaussian and Laplacian

Closer look at the diffusion term and requirements on $q(x,t)$

- Diffusion term

$$-\int_0^1 \mathbb{E}_{q(x,t|y)}\left[\frac{||f_1(x,t)-g_1(x,y,t)||^2}{2b^2(x,t)}\right]dt$$

- For efficient computation we need to be able to sample $q(x,t|y)$
- Distributions defined as flows and mixtures of flows permit this (proof not given)
- Flows include Gaussian and Laplacian
- Important points: Marginal determines drift term but noise $b(x,t)$ is a free parameter.

Closer look at the diffusion term and requirements on $q(x,t)$

- Diffusion term

$$-\int_0^1 \mathbb{E}_{q(x,t|y)}\left[\frac{||f_1(x,t) - g_1(x,y,t)||^2}{2b^2(x,t)}\right] dt$$

- For efficient computation we need to be able to sample $q(x,t|y)$
- Distributions defined as flows and mixtures of flows permit this (proof not given)
- Flows include Gaussian and Laplacian
- Important points: Marginal determines drift term but noise $b(x,t)$ is a free parameter.
- Gaussian marginal

$$q(x,t|y) = \mathcal{N}(x|\alpha(y,t), \beta^2(y,t))$$

- (VDM: $\alpha(y,t) = \alpha_t y_t$ and $\beta(y,t) = \sigma_t$)
- Drift term

$$\begin{aligned} g_0(x,y,t) &= \frac{\partial \beta(y,t)}{\partial t} \frac{x - \alpha(y,t)}{\beta(y,t)} + \frac{\partial \alpha(y,t)}{\partial t} \\ g_1(x,y,t) &= g_0(x,y,t) + \frac{\partial}{\partial x} \frac{b^2(x,t)}{2} + \frac{b^2(x,t)}{2} \frac{\partial \log q(x,t|y)}{\partial x} \quad . \end{aligned}$$

Summary and what was omitted

- Diffusion models are really hierarchical VAEs
- Connection even more apparent when we introduce more advanced encoders
- Continuous formulation more natural, but there is a learning curve.

Summary and what was omitted

- Diffusion models are really hierarchical VAEs
- Connection even more apparent when we introduce more advanced encoders
- Continuous formulation more natural, but there is a learning curve.
- What I didn't have time to talk about in detail:
  - Derivation master equation (Fokker-Planck)
  - Wiener process and derivation of Fokker-Planck for SDE
  - Derivation of reverse time SDE
  - Derivation of drift from marginal
  - Score/matching and stochastic interpolants.
- Watch out for Winther, Jeha and Dittadi, 2024, *in preparation*
- Learn to love calculus!

ODE and constant cumulative distribution path