

Deep Generative Models

Lecture 12

Roman Isachenko



Spring, 2022

Recap of previous lecture

Let take some pretrained image classification model to get the conditional label distribution $p(y|\mathbf{x})$ (e.g. ImageNet classifier).

Evaluation of likelihood-free models

- ▶ Sharpness \Rightarrow low $H(y|\mathbf{x}) = -\sum_y \int_{\mathbf{x}} p(y, \mathbf{x}) \log p(y|\mathbf{x}) d\mathbf{x}$.
- ▶ Diversity \Rightarrow high $H(y) = -\sum_y p(y) \log p(y)$.

Inception Score

$$IS = \exp(H(y) - H(y|\mathbf{x})) = \exp(\mathbb{E}_{\mathbf{x}} KL(p(y|\mathbf{x}) || p(y)))$$

Frechet Inception Distance

$$D^2(\pi, p) = \|\mathbf{m}_\pi - \mathbf{m}_p\|_2^2 + \text{Tr} \left(\boldsymbol{\Sigma}_\pi + \boldsymbol{\Sigma}_p - 2\sqrt{\boldsymbol{\Sigma}_\pi \boldsymbol{\Sigma}_p} \right).$$

FID is related to moment matching.

Salimans T. et al. *Improved Techniques for Training GANs*, 2016

Heusel M. et al. *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*, 2017

Recap of previous lecture

- ▶ $\mathcal{S}_\pi = \{\mathbf{x}_i\}_{i=1}^n \sim \pi(\mathbf{x})$ – real samples;
- ▶ $\mathcal{S}_p = \{\mathbf{x}_i\}_{i=1}^n \sim p(\mathbf{x}|\theta)$ – generated samples.

Embed samples using pretrained classifier network (as previously):

$$\mathcal{G}_\pi = \{\mathbf{g}_i\}_{i=1}^n, \quad \mathcal{G}_p = \{\mathbf{g}_i\}_{i=1}^n.$$

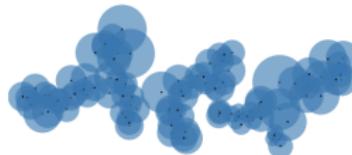
Define binary function:

$$f(\mathbf{g}, \mathcal{G}) = \begin{cases} 1, & \text{if exists } \mathbf{g}' \in \mathcal{G} : \|\mathbf{g} - \mathbf{g}'\|_2 \leq \|\mathbf{g}' - \text{NN}_k(\mathbf{g}', \mathcal{G})\|_2; \\ 0, & \text{otherwise.} \end{cases}$$

$$\text{Precision}(\mathcal{G}_\pi, \mathcal{G}_p) = \frac{1}{n} \sum_{\mathbf{g} \in \mathcal{G}_p} f(\mathbf{g}, \mathcal{G}_\pi); \quad \text{Recall}(\mathcal{G}_\pi, \mathcal{G}_p) = \frac{1}{n} \sum_{\mathbf{g} \in \mathcal{G}_\pi} f(\mathbf{g}, \mathcal{G}_p).$$



(a) True manifold



(b) Approx. manifold

Recap of previous lecture



- ▶ **Self-Attention GAN** allows to make huge receptive field and reduce convolution inductive bias.
- ▶ **BigGAN** shows that large batch size increase model quality gradually.
- ▶ **Progressive Growing GAN** starts from a low resolution, adds new layers that model fine details as training progresses.
- ▶ **StyleGAN** introduces mapping network to get more disentangled latent representation.

Outline

1. Discrete VAE latent representations

Vector quantization

Gumbel-softmax

2. Neural ODE

3. Continuous-in-time normalizing flows

Outline

1. Discrete VAE latent representations
2. Neural ODE
3. Continuous-in-time normalizing flows

Discrete VAE latents

Motivation

- ▶ Previous VAE models had **continuous** latent variables \mathbf{z} .
- ▶ **Discrete** representations \mathbf{z} are potentially a more natural fit for many of the modalities.
- ▶ Powerful autoregressive models (like PixelCNN) have been developed for modelling distributions over discrete variables.
- ▶ All cool transformer-like models work with discrete tokens.

ELBO

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log p(\mathbf{x}|\mathbf{z}, \theta) - KL(q(\mathbf{z}|\mathbf{x}, \phi) || p(\mathbf{z})) \rightarrow \max_{\phi, \theta} .$$

- ▶ Reparametrization trick to get unbiased gradients.
- ▶ Normal assumptions for $q(\mathbf{z}|\mathbf{x}, \phi)$ and $p(\mathbf{z})$ to compute KL analytically.

Discrete VAE latents

Assumptions

- ▶ Define dictionary (word book) space $\{\mathbf{e}_k\}_{k=1}^K$, where $\mathbf{e}_k \in \mathbb{R}^C$, K is the size of the dictionary.
- ▶ Let $c \sim \text{Categorical}(\boldsymbol{\pi})$, where
$$\boldsymbol{\pi} = (\pi_1, \dots, \pi_K), \quad \pi_k = P(c = k), \quad \sum_{k=1}^K \pi_k = 1.$$
- ▶ Let VAE model has discrete latent representation c with prior $p(c) = \text{Uniform}\{1, \dots, K\}$.

How it should work?

- ▶ Our variational posterior $q(c|\mathbf{x}, \phi) = \text{Categorical}(\boldsymbol{\pi}(\mathbf{x}, \phi))$ (encoder) outputs discrete probabilities vector.
- ▶ We sample c^* from $q(c|\mathbf{x}, \phi)$ (reparametrization trick analogue).
- ▶ Our generative distribution $p(\mathbf{x}|\mathbf{e}_{c^*}, \theta)$ (decoder).

Discrete VAE latents

ELBO

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{q(c|\mathbf{x}, \phi)} \log p(\mathbf{x}|c, \theta) - KL(q(c|\mathbf{x}, \phi)||p(c)) \rightarrow \max_{\phi, \theta}.$$

KL term

$$KL(q(c|\mathbf{x}, \phi)||p(c)) = \sum_{k=1}^K q(k|\mathbf{x}, \phi) \log \frac{q(k|\mathbf{x}, \phi)}{p(k)} = \\ = \sum_{k=1}^K q(k|\mathbf{x}, \phi) [\log q(k|\mathbf{x}, \phi) - \log p(k)] = -H(q(z|\mathbf{x}, \phi)) + \log K.$$

- ▶ Reparametrization trick does not work now (we sample from discrete distribution).
- ▶ Entropy term should be estimated.

Outline

1. Discrete VAE latent representations

Vector quantization

Gumbel-softmax

2. Neural ODE

3. Continuous-in-time normalizing flows

Vector quantization

Quantized representation

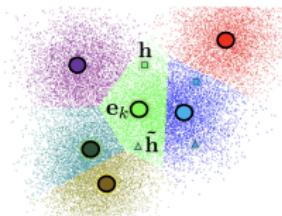
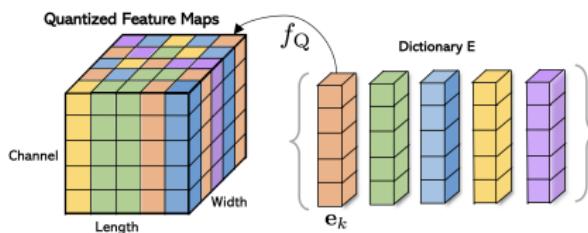
$\mathbf{z}_q \in \mathbb{R}^C$ for $\mathbf{z} \in \mathbb{R}^C$ is defined by a nearest neighbour look-up using the shared dictionary space

$$\mathbf{z}_q = \mathbf{e}_{k^*}, \quad \text{where } k^* = \arg \min_k \|\mathbf{z} - \mathbf{e}_k\|.$$

- ▶ Let our encoder outputs continuous representation \mathbf{z} .
- ▶ Quantization will give us the discrete distribution $q(c|x, \phi)$.

Quantization procedure

If we have tensor with the spatial dimensions we apply the quantization for each of $W \times H$ locations.



Vector Quantized VAE

Let VAE latent variable $\mathbf{c} \in \mathbb{R}^{W \times H}$ is the discrete with spatial-independent variational posterior and prior distributions

$$q(\mathbf{c}|\mathbf{x}, \phi) = \prod_{i=1}^W \prod_{j=1}^H q(c_{ij}|\mathbf{x}, \phi); \quad p(\mathbf{c}) = \prod_{i=1}^W \prod_{j=1}^H \text{Uniform}\{1, \dots, K\}.$$

Let $\mathbf{z}_e = \text{NN}(\mathbf{x}, \phi) \in \mathbb{R}^{W \times H \times C}$ is the encoder output.

Deterministic variational posterior

$$q(\hat{z}_{ij} = z_{k^*}|\mathbf{x}, \phi) = \begin{cases} 1, & \text{for } k^* = \arg \min_k \|[\mathbf{z}_e]_{ij} - \mathbf{e}_k\|; \\ 0, & \text{otherwise.} \end{cases}$$

$KL(q(c|\mathbf{x}, \phi)||p(c))$ term in ELBO is constant, entropy of the posterior is zero.

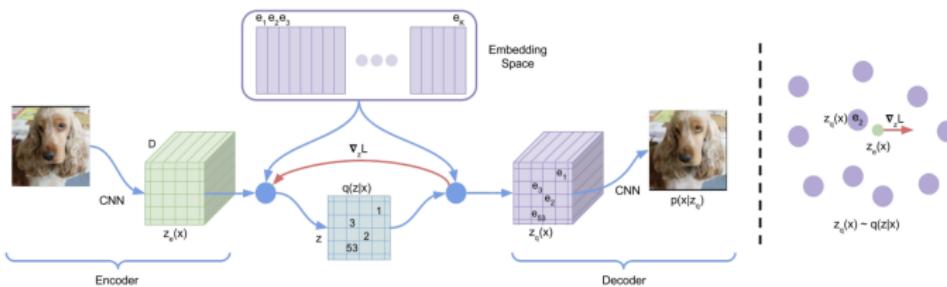
$$KL(q(c|\mathbf{x}, \phi)||p(c)) = -H(q(c|\mathbf{x}, \phi)) + \log K = \log K.$$

Vector Quantized VAE

ELBO

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{q(c|x, \phi)} \log p(x|e_c, \theta) - \log K = \log p(x|z_q, \theta) - \log K,$$

where $z_q = e_{k^*}$, $k^* = \arg \min_k \|z_e - e_k\|$.



Problem: $\arg \min$ is not differentiable.

Straight-through gradient estimation

$$\frac{\partial \log p(x|z_q, \theta)}{\partial \phi} = \frac{\partial \log p(x|z_q, \theta)}{\partial z_q} \cdot \frac{\partial z_q}{\partial \phi} \approx \frac{\partial \log p(x|z_q, \theta)}{\partial z_q} \cdot \frac{\partial z_e}{\partial \phi}$$

Vector Quantized VAE-2

Samples 1024x1024



Samples diversity



VQ-VAE (Proposed)

BigGAN deep

Razavi A., Oord A., Vinyals O. Generating Diverse High-Fidelity Images with VQ-VAE-2, 2019

Outline

1. Discrete VAE latent representations

Vector quantization

Gumbel-softmax

2. Neural ODE

3. Continuous-in-time normalizing flows

Gumbel-softmax trick

- ▶ VQ-VAE has deterministic variational posterior (it allows to get rid of discrete sampling and reparametrization trick).
- ▶ There is no uncertainty in the encoder output.

Gumbel-max trick

Let $g_k \sim \text{Gumbel}(0, 1)$ for $k = 1, \dots, K$, i.e. $g = -\log(\log u)$, $u \sim \text{Uniform}[0, 1]$. Then a discrete random variable

$$c = \arg \max_k [\log \pi_k + g_k],$$

has a categorical distribution $c \sim \text{Categorical}(\pi)$.

- ▶ Let our encoder $q(c|\mathbf{x}, \phi) = \text{Categorical}(\pi(\mathbf{x}, \phi))$ outputs logits of $\pi(\mathbf{x}, \phi)$.
- ▶ We could sample from the discrete distribution using Gumbel-max reparametrization.

Maddison C. J., Mnih A., Teh Y. W. *The Concrete distribution: A continuous relaxation of discrete random variables*, 2016

Jang E., Gu S., Poole B. *Categorical reparameterization with Gumbel-Softmax*, 2016

Gumbel-softmax trick

Reparametrization trick

$$\nabla_{\phi} \mathbb{E}_{q(c|\mathbf{x}, \phi)} \log p(\mathbf{x}|\mathbf{e}_c, \theta) = \mathbb{E}_{\text{Gumbel}(0,1)} \nabla_{\phi} \log p(\mathbf{x}|\mathbf{e}_{k^*}, \theta),$$

where $k^* = \arg \max_k [\log q(k|\mathbf{x}, \phi) + g_k]$.

Problem: We still have non-differentiable $\arg \max$ operation.

Gumbel-softmax relaxation

Concrete distribution = continuous + discrete

$$\hat{c}_k = \frac{\exp\left(\frac{\log q(k|\mathbf{x}, \phi) + g_k}{\tau}\right)}{\sum_{j=1}^K \exp\left(\frac{\log q(j|\mathbf{x}, \phi) + g_j}{\tau}\right)}, \quad k = 1, \dots, K.$$

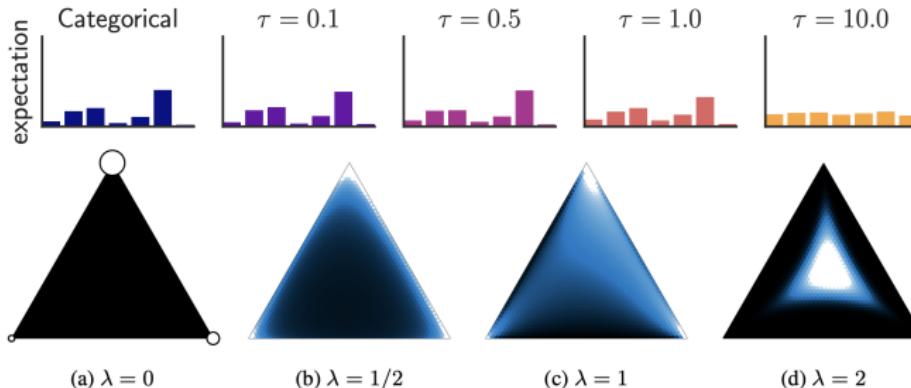
Here τ is a temperature parameter. Now we have differentiable operation, but the gradient estimate is biased now.

Maddison C. J., Mnih A., Teh Y. W. *The Concrete distribution: A continuous relaxation of discrete random variables*, 2016

Jang E., Gu S., Poole B. *Categorical reparameterization with Gumbel-Softmax*, 2016

Gumbel-softmax trick

Concrete distribution



Reparametrization trick

$$\nabla_{\phi} \mathbb{E}_{q(c|x, \phi)} \log p(\mathbf{x}|\mathbf{e}_c, \theta) = \mathbb{E}_{\text{Gumbel}(0,1)} \nabla_{\phi} \log p(\mathbf{x}|\mathbf{z}, \theta),$$

where $\mathbf{z} = \sum_{k=1}^K \hat{c}_k \mathbf{e}_k$ (all operations are differentiable now).

Maddison C. J., Mnih A., Teh Y. W. *The Concrete distribution: A continuous relaxation of discrete random variables*, 2016

Jang E., Gu S., Poole B. *Categorical reparameterization with Gumbel-Softmax*, 2016

DALL-E

Deterministic VQ-VAE posterior

$$q(\hat{z}_{ij} = k^* | \mathbf{x}) = \begin{cases} 1, & \text{for } k^* = \arg \min_k \|[\mathbf{z}_e]_{ij} - \mathbf{e}_k\| \\ 0, & \text{otherwise.} \end{cases}$$

- ▶ Gumbel-Softmax trick allows to make true categorical distribution and sample from it.
- ▶ Since latent space is discrete we could train autoregressive transformers in it.
- ▶ It is a natural way to incorporate text and image token spaces.

TEXT PROMPT

an armchair in the shape of an avocado [...]

AI-GENERATED IMAGES



Outline

1. Discrete VAE latent representations
2. Neural ODE
3. Continuous-in-time normalizing flows

Neural ODE

Consider Ordinary Differential Equation

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), \theta); \quad \text{with initial condition } \mathbf{z}(t_0) = \mathbf{z}_0.$$

$$\mathbf{z}(t_1) = \int_{t_0}^{t_1} f(\mathbf{z}(t), \theta) dt + \mathbf{z}_0 = \text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \theta).$$

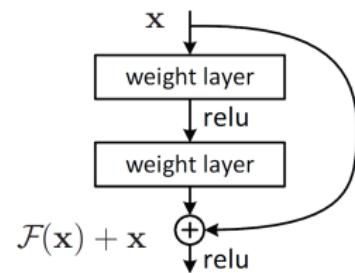
Euler update step

$$\frac{\mathbf{z}(t + \Delta t) - \mathbf{z}(t)}{\Delta t} = f(\mathbf{z}(t), \theta) \Rightarrow \mathbf{z}(t + \Delta t) = \mathbf{z}(t) + \Delta t f(\mathbf{z}(t), \theta).$$

Residual block

$$\mathbf{z}_{t+1} = \mathbf{z}_t + f(\mathbf{z}_t, \theta)$$

- ▶ It is equivalent to Euler update step for solving ODE with $\Delta t = 1$!
- ▶ Euler update step is unstable and trivial.
There are more sophisticated methods.



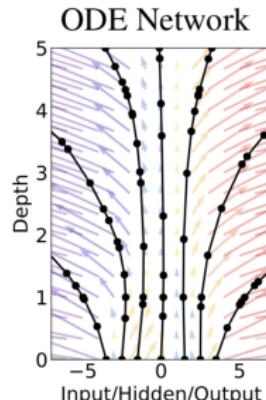
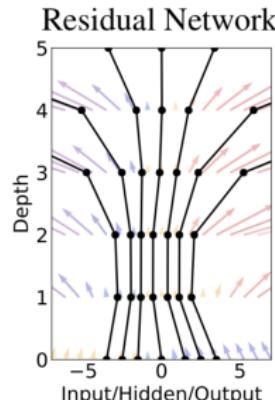
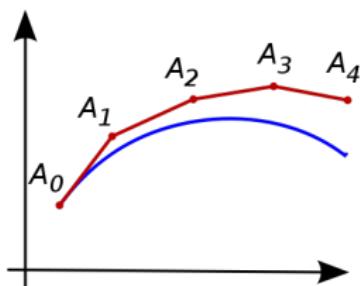
Neural ODE

Residual block

$$\mathbf{z}_{t+1} = \mathbf{z}_t + f(\mathbf{z}_t, \theta).$$

In the limit of adding more layers and taking smaller steps, we parameterize the continuous dynamics of hidden units using an ODE specified by a neural network:

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), t, \theta); \quad \mathbf{z}(t_0) = \mathbf{x}; \quad \mathbf{z}(t_1) = \mathbf{y}.$$



Neural ODE

Forward pass (loss function)

$$\begin{aligned} L(\mathbf{y}) &= L(\mathbf{z}(t_1)) = L \left(\mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), \theta) dt \right) \\ &= L(\text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \theta)) \end{aligned}$$

Note: ODESolve could be any method (Euler step, Runge-Kutta methods).

Backward pass (gradients computation)

For fitting parameters we need gradients:

$$\mathbf{a}_\mathbf{z}(t) = \frac{\partial L(\mathbf{y})}{\partial \mathbf{z}(t)}; \quad \mathbf{a}_\theta(t) = \frac{\partial L(\mathbf{y})}{\partial \theta(t)}.$$

In theory of optimal control these functions called **adjoint** functions. They show how the gradient of the loss depends on the hidden state $\mathbf{z}(t)$ and parameters θ .

Neural ODE

Adjoint functions

$$\mathbf{a}_z(t) = \frac{\partial L(\mathbf{y})}{\partial \mathbf{z}(t)}; \quad \mathbf{a}_{\theta}(t) = \frac{\partial L(\mathbf{y})}{\partial \theta(t)}.$$

Theorem (Pontryagin)

$$\frac{d\mathbf{a}_z(t)}{dt} = -\mathbf{a}_z(t)^T \cdot \frac{\partial f(\mathbf{z}(t), \theta)}{\partial \mathbf{z}}; \quad \frac{d\mathbf{a}_{\theta}(t)}{dt} = -\mathbf{a}_z(t)^T \cdot \frac{\partial f(\mathbf{z}(t), \theta)}{\partial \theta}.$$

Do we know any initial condition?

Solution for adjoint function

$$\frac{\partial L}{\partial \theta(t_0)} = \mathbf{a}_{\theta}(t_0) = - \int_{t_1}^{t_0} \mathbf{a}_z(t)^T \frac{\partial f(\mathbf{z}(t), \theta)}{\partial \theta(t)} dt + 0$$

$$\frac{\partial L}{\partial \mathbf{z}(t_0)} = \mathbf{a}_z(t_0) = - \int_{t_1}^{t_0} \mathbf{a}_z(t)^T \frac{\partial f(\mathbf{z}(t), \theta)}{\partial \mathbf{z}(t)} dt + \frac{\partial L}{\partial \mathbf{z}(t_1)}$$

Note: These equations are solved back in time.

Neural ODE

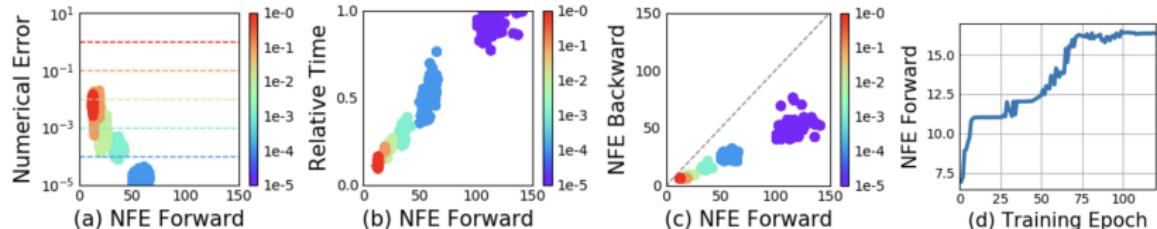
Forward pass

$$\mathbf{z}(t_1) = \int_{t_0}^{t_1} f(\mathbf{z}(t), \theta) dt + \mathbf{z}_0 \quad \Rightarrow \quad \text{ODE Solver}$$

Backward pass

$$\left. \begin{aligned} \frac{\partial L}{\partial \theta(t_0)} &= \mathbf{a}_\theta(t_0) = - \int_{t_1}^{t_0} \mathbf{a}_z(t)^T \frac{\partial f(\mathbf{z}(t), \theta)}{\partial \theta(t)} dt + 0 \\ \frac{\partial L}{\partial \mathbf{z}(t_0)} &= \mathbf{a}_z(t_0) = - \int_{t_1}^{t_0} \mathbf{a}_z(t)^T \frac{\partial f(\mathbf{z}(t), \theta)}{\partial \mathbf{z}(t)} dt + \frac{\partial L}{\partial \mathbf{z}(t_1)} \\ \mathbf{z}(t_0) &= - \int_{t_1}^{t_0} f(\mathbf{z}(t), \theta) dt + \mathbf{z}_1. \end{aligned} \right\} \Rightarrow \text{ODE Solver}$$

Note: These scary formulas are the standard backprop in the discrete case.



Outline

1. Discrete VAE latent representations
2. Neural ODE
3. Continuous-in-time normalizing flows

Continuous Normalizing Flows

Discrete Normalizing Flows

$$\mathbf{z}_{t+1} = f(\mathbf{z}_t, \theta); \quad \log p(\mathbf{z}_{t+1}) = \log p(\mathbf{z}_t) - \log \left| \det \frac{\partial f(\mathbf{z}_t, \theta)}{\partial \mathbf{z}_t} \right|.$$

Continuous-in-time dynamic transformation

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), \theta).$$

Assume that function f is uniformly Lipschitz continuous in \mathbf{z} and continuous in t . From Picard's existence theorem, it follows that the above ODE has a **unique solution**.

Forward and inverse transforms

$$\mathbf{x} = \mathbf{z}(t_1) = \mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), \theta) dt$$

$$\mathbf{z} = \mathbf{z}(t_0) = \mathbf{z}(t_1) + \int_{t_1}^{t_0} f(\mathbf{z}(t), \theta) dt$$

Continuous Normalizing Flows

To train this flow we have to get the way to calculate the density $p(\mathbf{z}(t))$.

Theorem (Fokker-Planck)

if function f is uniformly Lipschitz continuous in \mathbf{z} and continuous in t , then

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\text{trace} \left(\frac{\partial f(\mathbf{z}(t), \theta)}{\partial \mathbf{z}(t)} \right).$$

Note: Unlike discrete-in-time flows, the function f does not need to be bijective, because uniqueness guarantees that the entire transformation is automatically bijective.

Density evaluation

$$\log p(\mathbf{x}|\theta) = \log p(\mathbf{z}) - \int_{t_0}^{t_1} \text{trace} \left(\frac{\partial f(\mathbf{z}(t), \theta)}{\partial \mathbf{z}(t)} \right) dt.$$

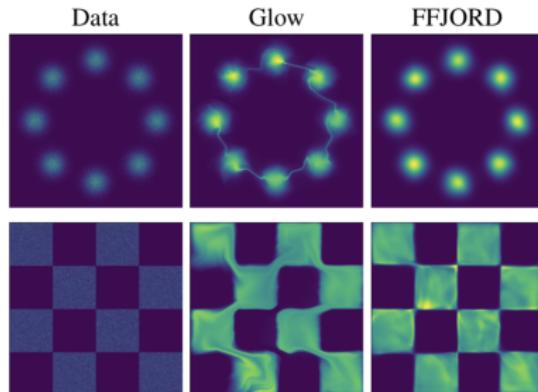
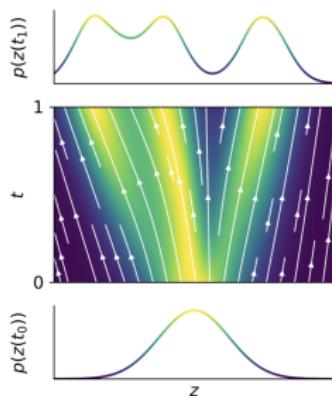
Adjoint method is used to integral evaluation.

Continuous Normalizing Flows

Forward transform + log-density

$$\begin{bmatrix} \mathbf{x} \\ \log p(\mathbf{x}|\theta) \end{bmatrix} = \begin{bmatrix} \mathbf{z} \\ \log p(\mathbf{z}) \end{bmatrix} + \int_{t_0}^{t_1} \begin{bmatrix} f(\mathbf{z}(t), \theta) \\ -\text{trace}\left(\frac{\partial f(\mathbf{z}(t), \theta)}{\partial \mathbf{z}(t)}\right) \end{bmatrix} dt.$$

- Discrete-in-time normalizing flows need invertible f . It costs $O(d^3)$ to get determinant of Jacobian.
- Continuous-in-time flows require only smoothness of f . It costs $O(d^2)$ to get trace of Jacobian.



Summary

- ▶ Gumbel-Softmax and Quantization are the two ways to create VAE with discrete latent space.
- ▶ It becomes more and more popular to use discrete latent spaces in the fields of image/video/music generation.
- ▶ Residual networks could be interpreted as solution of ODE with Euler method.
- ▶ Adjoint method generalizes backpropagation procedure and allows to train Neural ODE solving ODE for adjoint function back in time.
- ▶ Fokker-Planck theorem allows to construct continuous-in-time normalizing flow with less functional restrictions.
- ▶ FFJORD model makes such kind of flows scalable.