Deep Generative Models

Lecture 12

Roman Isachenko

Ozon Masters

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

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

Frechet Inception Distance

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

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

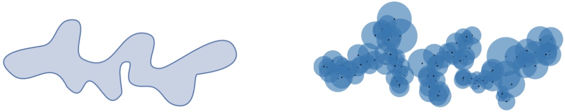
Embed samples using pretrained classifier network (as previously):

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

Define binary function:

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

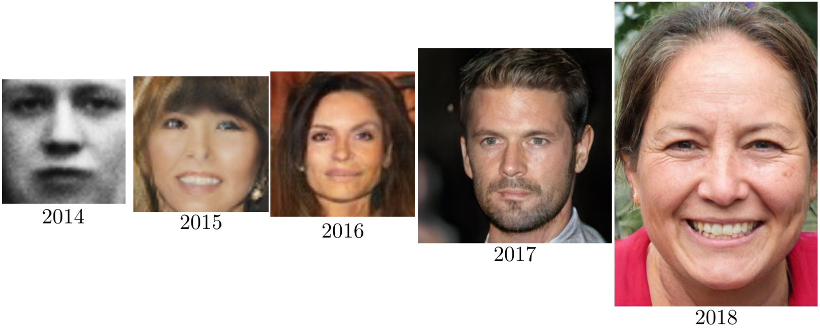
$$\text{Precision}(\mathcal{G}_\pi, \mathcal{G}_p) = \frac{1}{n} \sum_{\mathbf{g} \in \mathcal{G}_p} f(\mathbf{g}, \mathcal{G}_\pi); \quad \text{Recall}(\mathcal{G}_\pi, \mathcal{G}_p) = \frac{1}{n} \sum_{\mathbf{g} \in \mathcal{G}_\pi} f(\mathbf{g}, \mathcal{G}_p).$$



(a) True manifold (b) Approx. manifold

Kynkäänniemi T. et al. Improved precision and recall metric for assessing generative models, 2019

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
   Gumbel-softmax
   Vector quantization

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.

ELBO

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

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

Discrete VAE latents

Let $z \sim \text{Categorical}(\boldsymbol{\pi})$, where

$$\boldsymbol{\pi} = (\pi_1, \dots, \pi_K), \quad \pi_k = P(z = z_k), \quad \sum_k \pi_k = 1.$$

We assume that the prior distribution $p(z) = \text{Uniform}\{z_1, \dots, z_K\}$.

$$KL(q(z|\mathbf{x}, \phi)||p(z)) = \sum_{k=1}^{K} q(z_k|\mathbf{x}, \phi) \log \frac{q(z_k|\mathbf{x}, \phi)}{p(z_k)} =$$

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

ELBO

$$\mathcal{L}(\phi, \boldsymbol{\theta}) = \mathbb{E}_{q(z|\mathbf{x}, \phi)} \log p(\mathbf{x}|z, \boldsymbol{\theta}) + H(q(z|\mathbf{x}, \phi)) - \log K \to \max_{\boldsymbol{\phi}, \boldsymbol{\theta}}.$$

- Reparametrization trick does not work now.
- Entropy term should be estimated.

Outline

1. Discrete VAE latent representations
   Gumbel-softmax
   Vector quantization

2. Neural ODE

3. Continuous-in-time normalizing flows

Gumbel-softmax trick

If z is a discrete random variable we cannot differentiate through it.

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

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

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

Reparametrization trick

$$\nabla_\phi \mathbb{E}_{q(z|\phi) f(z)} = \mathbb{E}_{\text{Gumbel}(0,1)} \nabla_\phi f \left(\arg\max_k [\log q(z_k|\phi) + g_k] \right).$$

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

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

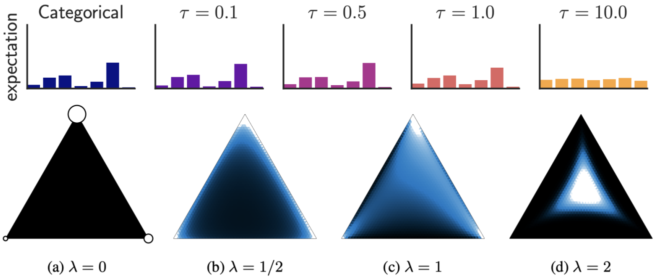
Concrete distribution

Gumbel-softmax relaxation

Concrete distribution = continuous + discrete

$$z_k = \frac{\exp((\log \pi_k + G_k)/\tau)}{\sum_{j=1}^{K} \exp((\log \pi_j + G_j)/\tau)}, \quad k = 1, \dots, K.$$

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

Outline

1. Discrete VAE latent representations
   Gumbel-softmax
   Vector quantization

2. Neural ODE

3. Continuous-in-time normalizing flows

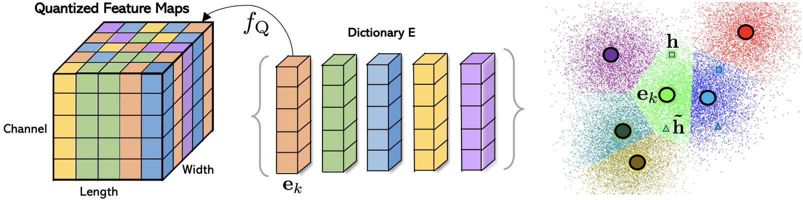
Vector quantization

- Define dictionary space $\{\mathbf{e}_k\}_{k=1}^{K}$, where $\mathbf{e}_k \in \mathbb{R}^C$, K is the size of the dictionary.
- Quantized representation $\mathbf{z}_q \in \mathbb{R}^{W \times H \times C}$ for $\mathbf{z} \in \mathbb{R}^C$ is defined by a nearest neighbour look-up using the shared dictionary space

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

Quantization procedure

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



Zhao Y. et al. Feature Quantization Improves GAN Training, 2020

Vector Quantized VAE

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

$$q(\hat{\mathbf{z}}|\mathbf{x}, \phi) = \prod_{i=1}^{W} \prod_{j=1}^{H} q(\hat{z}_{ij}|\mathbf{x}, \phi); \quad p(\hat{\mathbf{z}}) = \prod_{i=1}^{W} \prod_{j=1}^{H} \text{Uniform}\{z_1, \dots, z_K\}.$$

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

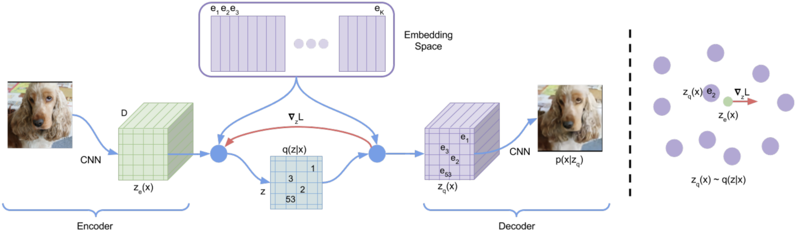
$$q(\hat{z}_{ij} = z_k^*|\mathbf{x}, \phi) = \begin{cases} 1, & \text{for } k^* = \arg\min_k \|[\mathbf{z}_e]_{ij} - \mathbf{e}_k\| \\ 0, & \text{otherwise.} \end{cases}$$

- VAE posterior distribution $q(\hat{\mathbf{z}}|\mathbf{x})$ is deterministic (zero entropy).
- $KL(q(\hat{z}|\mathbf{x})||p(\hat{z}))$ term in ELBO is constant.

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

Oord A., Vinyals O., Kavukcuoglu K. Neural Discrete Representation Learning, 2017

Vector Quantized VAE



Objective

$$\log p(\mathbf{x}|\mathbf{z}_q) + \|\mathrm{sg}(\mathbf{z}_e) - \mathbf{z}_q\| + \beta\|\mathbf{z}_e - \mathrm{sg}(\mathbf{z}_q)\|$$

- First term is ELBO part.
- Quantization operation is not differentiable.
- Straight-through gradient estimation is used to backpropagate the quantization operation.

Oord A., Vinyals O., Kavukcuoglu K. Neural Discrete Representation Learning, 2017

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

DALL-E

Deterministic VQ-VAE posterior

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

- It is possible to use Gumbel-Softmax trick to relax this distribution to continuous one.
- Since latent space is discrete we could train autoregressive transformers in it.
- It is a natural way to incorporate text and image spaces.

TEXT PROMPT

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

AI-GENERATED IMAGES

Ramesh A. et al. Zero-shot text-to-image generation, 2021

Outline

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

Neural ODE

Consider Ordinary Differential Equation

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

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

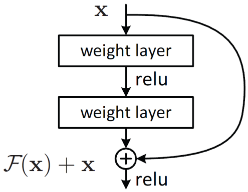
Euler update step

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

Residual block

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

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



Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

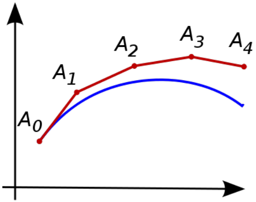
Neural ODE

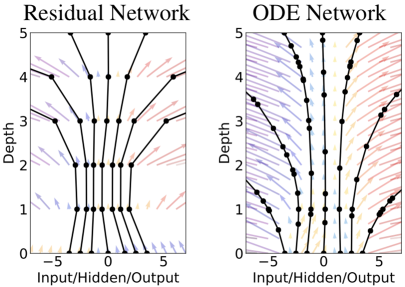
Residual block

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

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:

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





Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Neural ODE

Forward pass (loss function)

$$L(\mathbf{y}) = L(\mathbf{z}(t_1)) = L\left(\mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), \boldsymbol{\theta})dt\right)$$
$$= L\big(\mathrm{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \boldsymbol{\theta})\big)$$

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

Backward pass (gradients computation)

For fitting parameters we need gradients:

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

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 $\boldsymbol{\theta}$.

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Neural ODE

Adjoint functions

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

Theorem (Pontryagin)

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

Do we know any initilal condition?

Solution for adjoint function

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

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

Note: These equations are solved back in time.

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Neural ODE

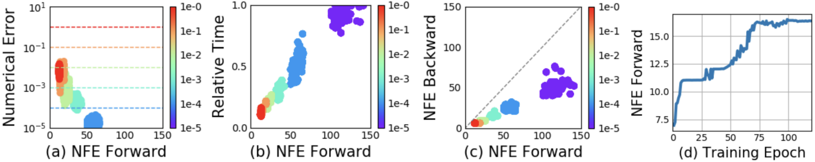
Forward pass

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

Backward pass

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

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



Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Outline

1. Discrete VAE latent representations

2. Neural ODE

3. Continuous-in-time normalizing flows

Continuous Normalizing Flows

Discrete Normalizing Flows

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

Continuous-in-time dynamic transformation

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

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

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

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

Papamakarios G. et al. Normalizing flows for probabilistic modeling and inference, 2019

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

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

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

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

Adjoint method is used to integral evaluation.

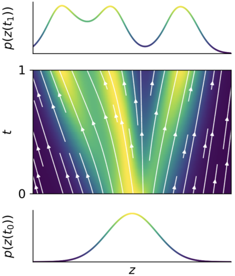
Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

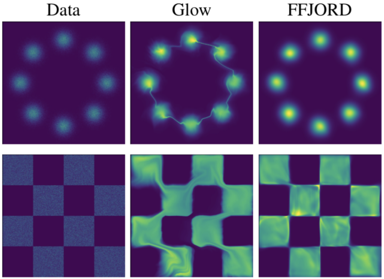
Continuous Normalizing Flows

Forward transform + log-density

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

- 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.





Grathwohl W. et al. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models, 2018

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 generaton.
- 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.