

# Deep Generative Models

## Lecture 13

Roman Isachenko



Spring, 2022

## 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):

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

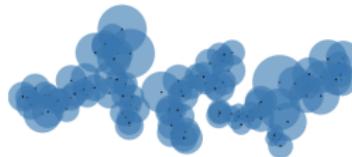
Define binary function:

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

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



(a) True manifold



(b) Approx. manifold

## Recap of previous lecture



2018

- ▶ **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.

## Recap of previous lecture

### ELBO objective

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

What is the problem to make the variational posterior model an **implicit** model?

We have to estimate density ratio

$$r(x, z) = \frac{q_1(x, z)}{q_2(x, z)} = \frac{p(z)\pi(x)}{q(z|x, \phi)\pi(x)}.$$

## Adversarial Variational Bayes

$$\max_D \left[ \mathbb{E}_{\pi(x)} \mathbb{E}_{q(z|x, \phi)} \log D(x, z) + \mathbb{E}_{\pi(x)} \mathbb{E}_{p(z)} \log(1 - D(x, z)) \right]$$

# Outline

# Neural ODE

Consider Ordinary Differential Equation

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

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

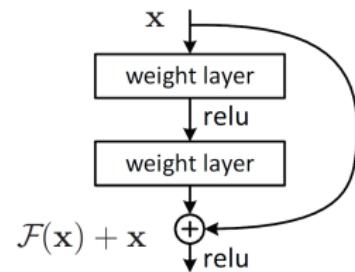
Euler update step

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

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.



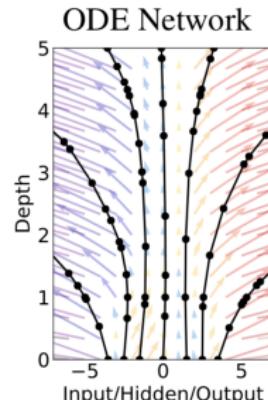
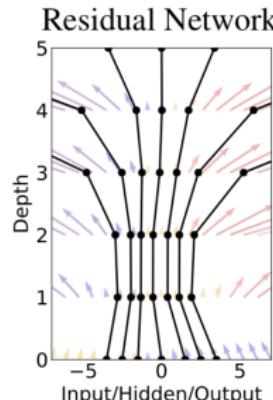
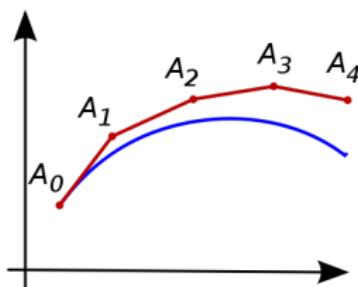
# Neural ODE

## 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  $\theta$ .

# 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

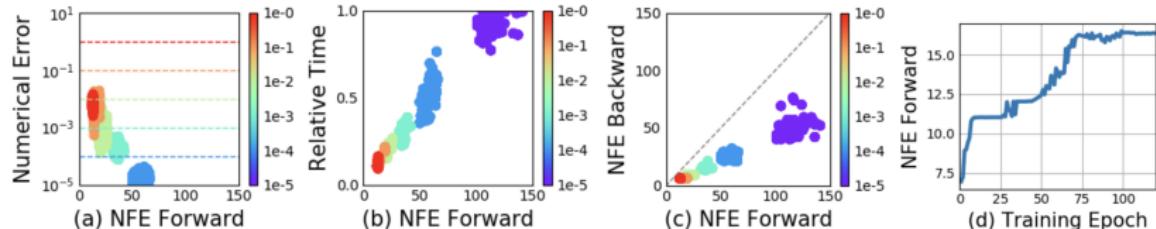
## Forward pass

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

## 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.



# 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

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

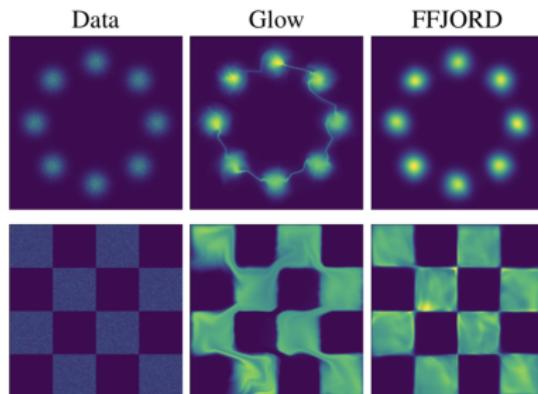
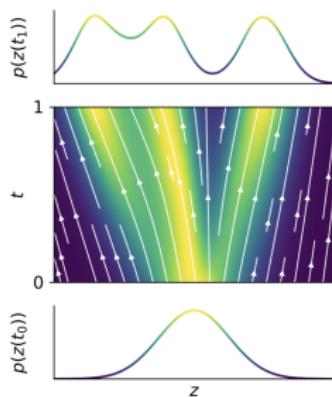
**Adjoint** method is used to integral evaluation.

# 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.



# FFJORD

Method		One-pass Sampling	Exact log-likelihood	Free-form Jacobian
Variational Autoencoders	Variational Autoencoders	✓	✗	✓
	Generative Adversarial Nets	✓	✗	✓
	Likelihood-based Autoregressive	✗	✓	✗
Change of Variables	Normalizing Flows	✓	✓	✗
	Reverse-NF, MAF, TAN	✗	✓	✗
	NICE, Real NVP, Glow, Planar CNF	✓	✓	✗
	<b>FFJORD</b>	✓	✓	✓

## Density estimation (forward KL)

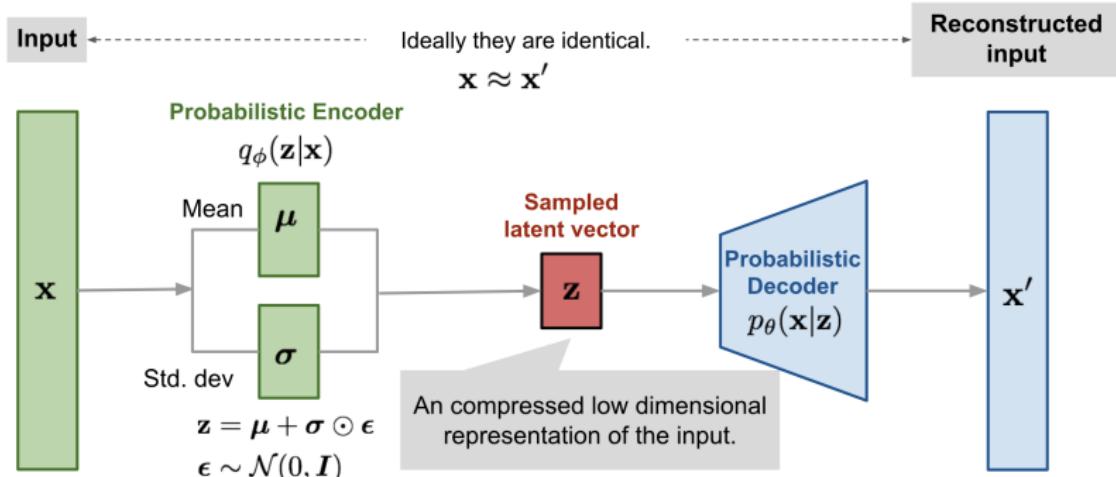
	POWER	GAS	HEPMASS	MINIBOONE	BSDS300	MNIST	CIFAR10
Real NVP	-0.17	-8.33	18.71	13.55	-153.28	1.06*	3.49*
Glow	-0.17	-8.15	18.92	11.35	-155.07	1.05*	<b>3.35*</b>
<b>FFJORD</b>	<b>-0.46</b>	<b>-8.59</b>	<b>14.92</b>	<b>10.43</b>	<b>-157.40</b>	<b>0.99*</b> (1.05 <sup>†</sup> )	3.40*

## Flows for variational inference (reverse KL)

	MNIST	Omniglot	Frey Faces	Caltech Silhouettes
IAF	$84.20 \pm .17$	$102.41 \pm .04$	$4.47 \pm .05$	$111.58 \pm .38$
Sylvester	$83.32 \pm .06$	$99.00 \pm .04$	$4.45 \pm .04$	$104.62 \pm .29$
<b>FFJORD</b>	<b><math>82.82 \pm .01</math></b>	<b><math>98.33 \pm .09</math></b>	<b><math>4.39 \pm .01</math></b>	<b><math>104.03 \pm .43</math></b>

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

# Discrete VAE



- ▶ Previous VAE models had **continuous** latent variables  $z$ .
- ▶ **Discrete** representations  $z$  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.

# Discrete VAE

If  $\mathbf{z}$  is a discrete random variable we cannot differentiate through it.

## Gumbel-Max trick

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

$$z = \arg \max_k (\log \pi_k + G_k), \quad \sum_k \pi_k = 1$$

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

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

## Gumbel-Softmax relaxation

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

Here  $\tau$  is a temperature parameter.

---

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

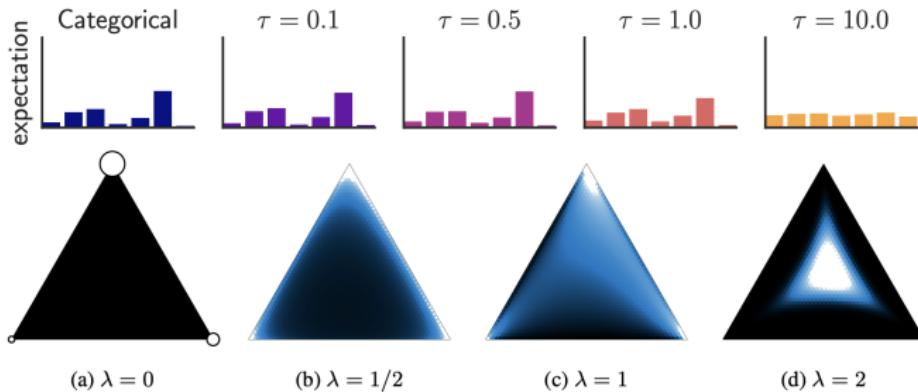
# Discrete VAE

## 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  $\tau$  is a temperature parameter. Now we have differentiable operation.



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

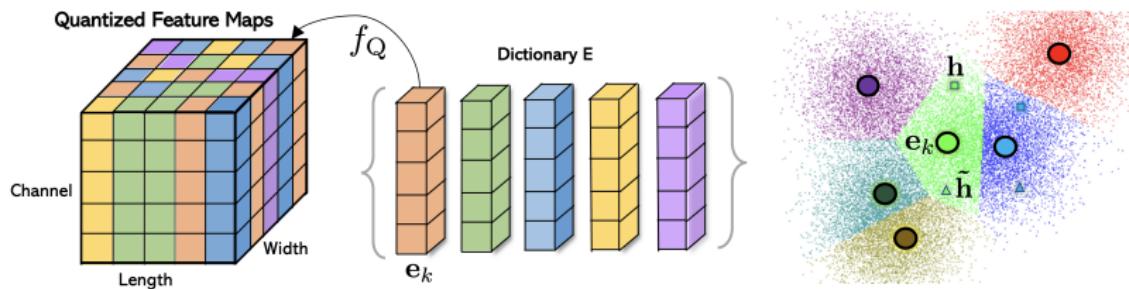
Jang E., Gu S., Poole B. *Categorical reparameterization with Gumbel-Softmax*, 2016

# Vector Quantized VAE

- ▶ Define dictionary space  $\{\mathbf{e}_k\}_{k=1}^K$ , where  $\mathbf{e}_k \in \mathbb{R}^C$ ,  $K$  is the size of the dictionary.
- ▶ Let  $\mathbf{z} = \text{NN}_e(\mathbf{x}) \in \mathbb{R}^{W \times H \times C}$  be an encoder output.
- ▶ Quantized representation  $\mathbf{z}_q \in \mathbb{R}^{W \times H \times C}$  is defined by a nearest neighbour look-up using the shared dictionary space for each of  $W \times H$  spatial locations

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

## Quantization procedure



## Vector Quantized VAE

Define VAE latent variable  $\hat{\mathbf{z}} \in \mathbb{R}^{W \times H}$  with prior distribution  $p(\hat{\mathbf{z}}) = \text{Uniform}\{1, \dots, K\}$  and variational posterior distribution

$$q(\hat{\mathbf{z}}|\mathbf{x}) = \prod_{i=1}^W \prod_{j=1}^H q(\hat{z}_{ij}|\mathbf{x})$$

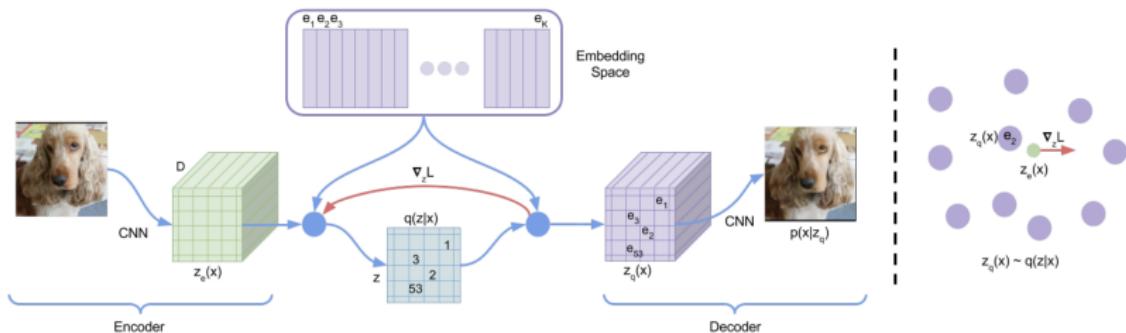
$$q(\hat{z}_{ij} = k^*|\mathbf{x}) = \begin{cases} 1, & \text{for } k^* = \arg \min_k \|[\mathbf{z}_e]_{ij} - \mathbf{e}_k\| \\ 0, & \text{otherwise.} \end{cases}$$

## ELBO objective

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

- ▶ VAE proposal distribution  $q(\hat{\mathbf{z}}|\mathbf{x})$  is deterministic.
- ▶  $KL(q(\hat{\mathbf{z}}|\mathbf{x})||p(\hat{\mathbf{z}}))$  term in ELBO is constant (equals to  $\log K$ ).

# Vector Quantized VAE



## Objective

$$\log p(x|z_q) + \|\text{sg}(z_e) - z_q\| + \beta \|z_e - \text{sg}(z_q)\|$$

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

# 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



## Summary

- ▶ 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.
- ▶ 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.