

Deep Generative Models

Lecture 13

Roman Isachenko

Moscow Institute of Physics and Technology

Autumn, 2021

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.

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?

- ▶ The third term requires the ability to calculate the value of $q(z|x, \phi)$.
- ▶ We could join second and third terms:

$$\mathbb{E}_{q(z|x, \phi)} \log \frac{p(z)}{q(z|x, \phi)} = \mathbb{E}_{q(z|x, \phi)} \log \frac{p(z)\pi(x)}{q(z|x, \phi)\pi(x)}.$$

- ▶ We have to estimate density ratio

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

Recap of previous lecture

Consider two distributions $q_1(\mathbf{x})$, $q_2(\mathbf{x})$ and probabilistic model

$$p(\mathbf{x}|y) = \begin{cases} q_1(\mathbf{x}), & \text{if } y = 1, \\ q_2(\mathbf{x}), & \text{if } y = 0, \end{cases} \quad y \sim \text{Bern}(0.5).$$

Density ratio trick

$$\begin{aligned} \frac{q_1(\mathbf{x})}{q_2(\mathbf{x})} &= \frac{p(\mathbf{x}|y=1)}{p(\mathbf{x}|y=0)} = \frac{p(y=1|\mathbf{x})p(\mathbf{x})}{p(y=1)} \Big/ \frac{p(y=0|\mathbf{x})p(\mathbf{x})}{p(y=0)} = \\ &= \frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})} = \frac{p(y=1|\mathbf{x})}{1 - p(y=1|\mathbf{x})} = \frac{D(\mathbf{x})}{1 - D(\mathbf{x})} \end{aligned}$$

Here $D(\mathbf{x})$ is a discriminator model the output of which is a probability that \mathbf{x} is a sample from $q_1(\mathbf{x})$ rather than from $q_2(\mathbf{x})$.

Adversarial Variational Bayes

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

Neural ODE

Consider Ordinary Differential Equation

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

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

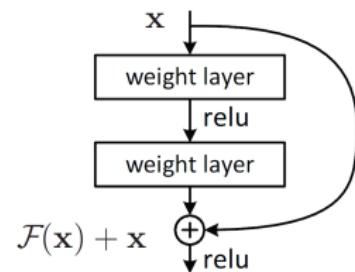
Euler update step

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

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.



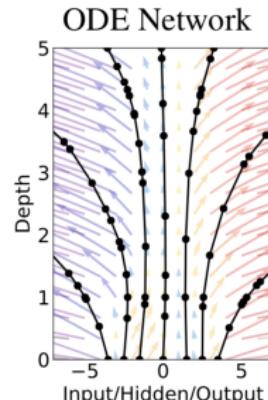
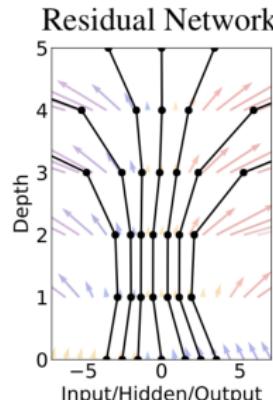
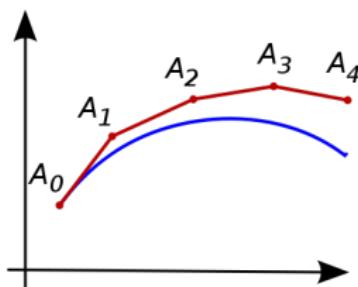
Neural ODE

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

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

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

Note: These equations are solved back in time.

Neural ODE

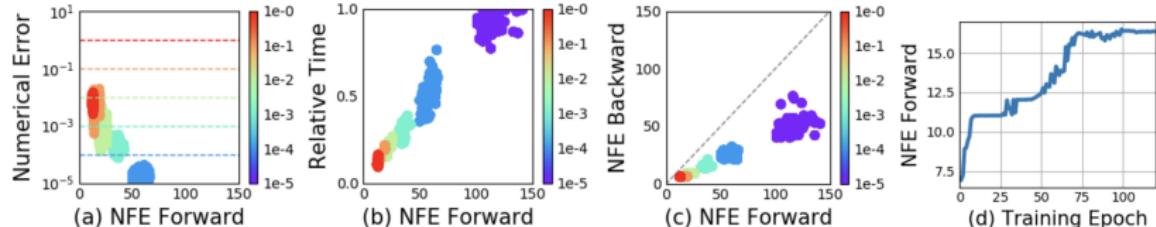
Forward pass

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

Backward pass

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

Note: This transformation is a generalization of usual backprop.



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

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

- ▶ 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.

It is possible to reduce cost from $O(d^2)$ to $O(d)$!

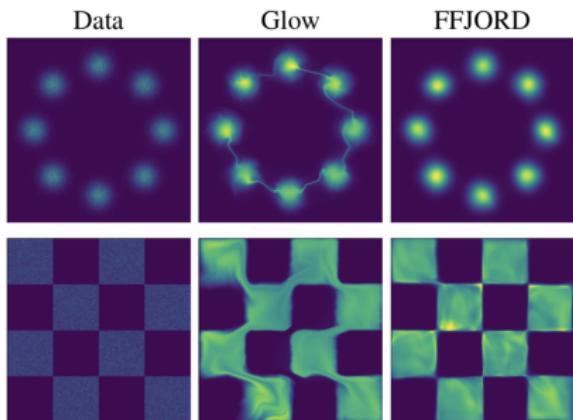
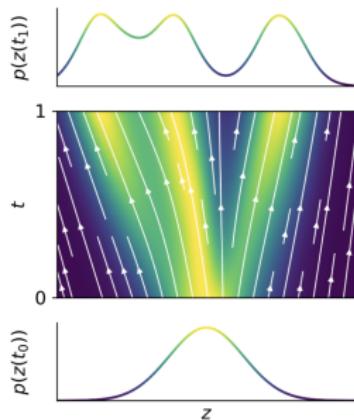
Hutchinson's trace estimator

$$\text{trace}(A) = \mathbb{E}_{p(\epsilon)} \left[\epsilon^T A \epsilon \right]; \quad \mathbb{E}[\epsilon] = 0; \quad \text{Cov}(\epsilon) = I.$$

FFJORD density estimation

$$\log p(\mathbf{z}(t_1)) = \log p(\mathbf{z}(t_0)) - \mathbb{E}_{p(\epsilon)} \int_{t_0}^{t_1} \left[\epsilon^T \frac{\partial f}{\partial \mathbf{z}} \epsilon \right] dt.$$

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	✓	✓	✗
	FFJORD	✓	✓	✓

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

FFJORD

Density estimation (forward KL)

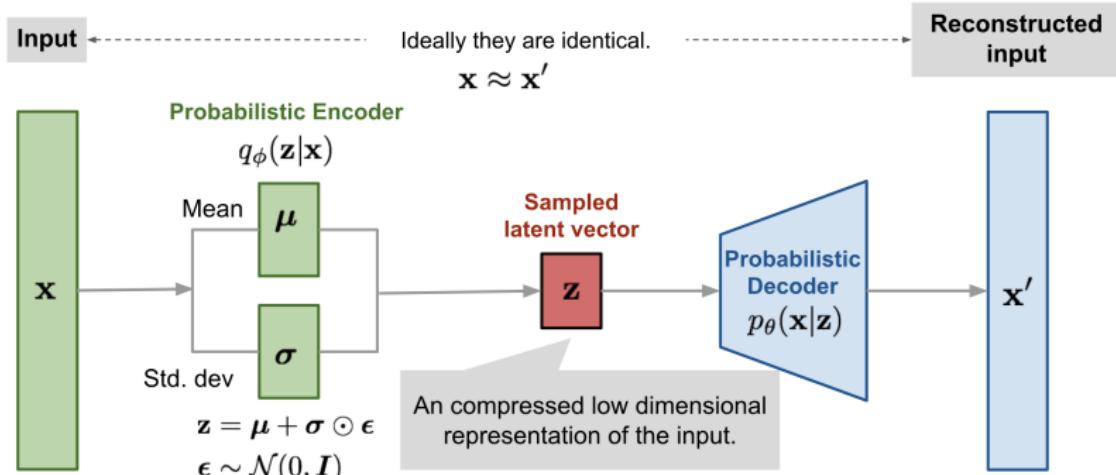
	POWER	GAS	HEPMASS	MINIBOONE	BSDS300	MNIST	CIFAR10
Real NVP	-0.17	-8.33	18.71	13.55	-153.28	1.06*	3.49*
Glow	-0.17	-8.15	18.92	11.35	-155.07	1.05*	3.35*
FFJORD	-0.46	-8.59	14.92	10.43	-157.40	0.99* (1.05 [†])	3.40*
MADE	3.08	-3.56	20.98	15.59	-148.85	2.04	5.67
MAF	-0.24	-10.08	17.70	11.75	-155.69	1.89	4.31
TAN	-0.48	-11.19	15.12	11.01	-157.03	-	-
MAF-DDSF	-0.62	-11.96	15.09	8.86	-157.73	-	-

Flows for variational inference (reverse KL)

	MNIST	Omniglot	Frey Faces	Caltech Silhouettes
No Flow	$86.55 \pm .06$	$104.28 \pm .39$	$4.53 \pm .02$	$110.80 \pm .46$
Planar	$86.06 \pm .31$	$102.65 \pm .42$	$4.40 \pm .06$	$109.66 \pm .42$
IAF	$84.20 \pm .17$	$102.41 \pm .04$	$4.47 \pm .05$	$111.58 \pm .38$
Sylvester	$83.32 \pm .06$	$99.00 \pm .04$	$4.45 \pm .04$	$104.62 \pm .29$
FFJORD	$82.82 \pm .01$	$98.33 \pm .09$	$4.39 \pm .01$	$104.03 \pm .43$

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

$$z = \arg \max_k (\log \pi_k + G_k)$$

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 τ 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

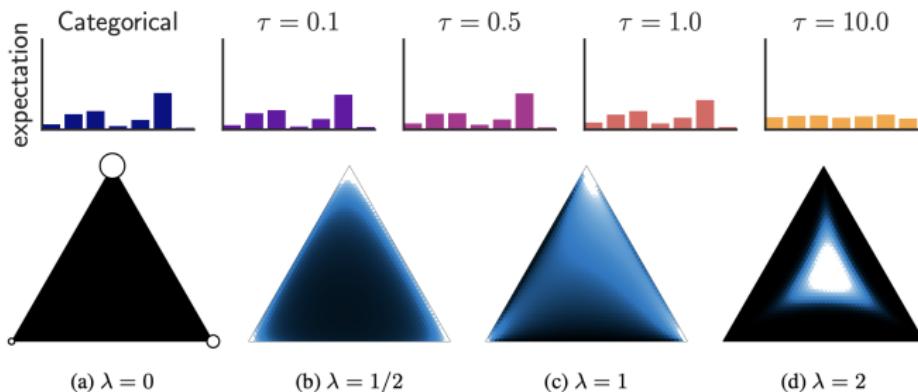
Discrete VAE

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 τ 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

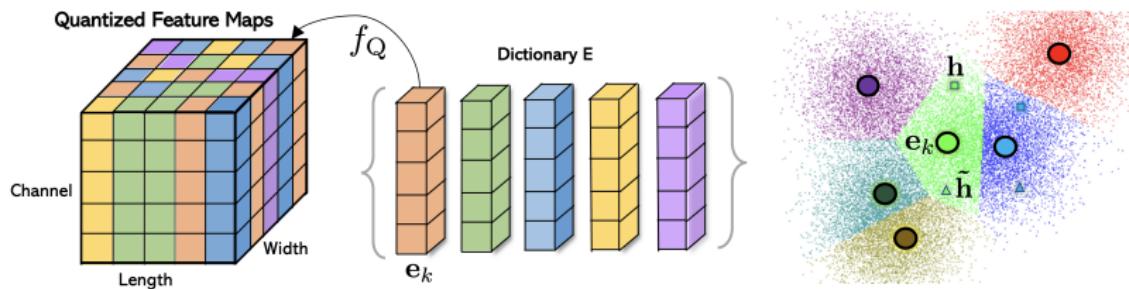
Jang E., Gu S., Poole B. *Categorical reparameterization with Gumbel-Softmax*, 2016

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

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

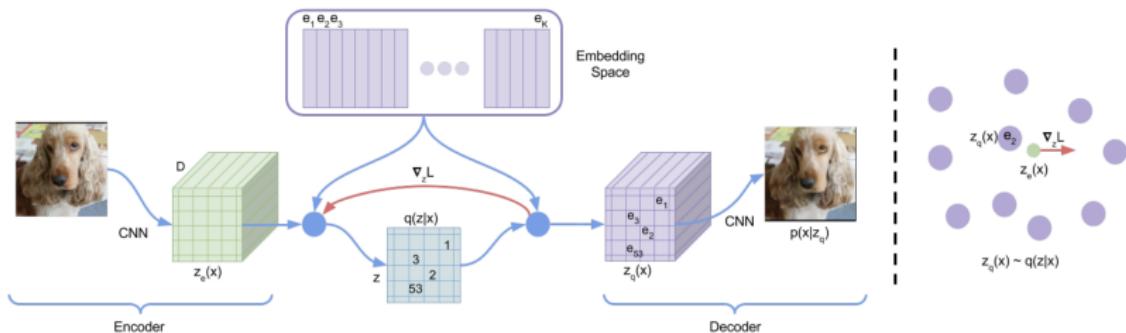
$$q(\hat{z}_{ij} = k^*|\mathbf{x}) = \begin{cases} 1, & \text{for } k^* = \arg \min_k \|[\mathbf{z}_e]_{ij} - \mathbf{e}_k\| \\ 0, & \text{otherwise.} \end{cases}$$

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

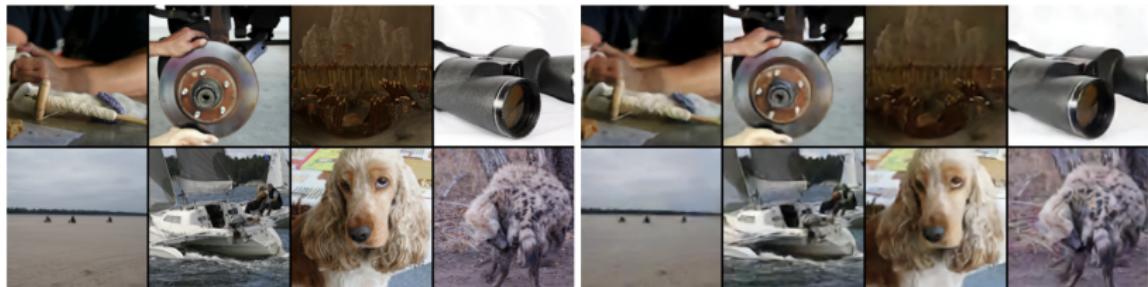
$$\log p(x|z_q) + \|\text{sg}(z_e) - z_q\| + \beta \|z_e - \text{sg}(z_q)\|$$

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

Vector Quantized VAE

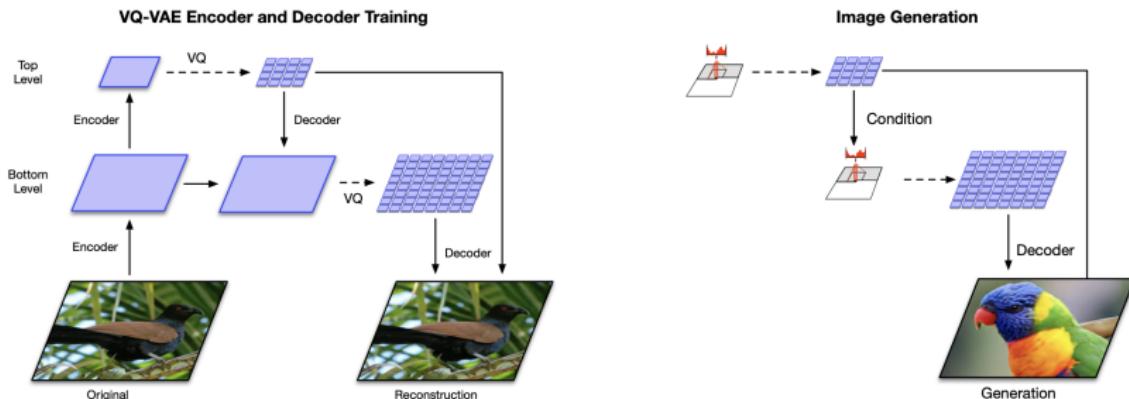
- ▶ The prior distribution over the discrete latents $p(\hat{z})$ is a categorical distribution.
- ▶ It could be made autoregressive by depending on other \hat{z} in the feature map.
- ▶ While training the VQ-VAE, the prior is kept constant and uniform.
- ▶ After training, fit an autoregressive distribution (using PixelCNN) over \hat{z} .

Samples



Vector Quantized VAE-2

- ▶ Use multi-scale hierarchical model.
- ▶ Use autoregressive prior model in each scale of the hierarchy.
- ▶ Improve autoregressive prior (PixelSNAIL with self-attention in bottom layer, PixelCNN++ in bottom layer).
- ▶ Train the encoder and decoder at the first stage, train the priors at the second stage.



Vector Quantized VAE-2

Algorithm 1 VQ-VAE training (stage 1)

Require: Functions E_{top} , E_{bottom} , D , \mathbf{x} (batch of training images)

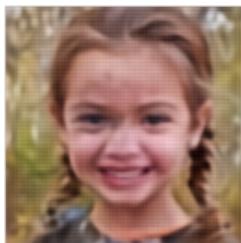
- 1: $\mathbf{h}_{top} \leftarrow E_{top}(\mathbf{x})$
 ▷ quantize with top codebook eq 1
- 2: $\mathbf{e}_{top} \leftarrow Quantize(\mathbf{h}_{top})$
- 3: $\mathbf{h}_{bottom} \leftarrow E_{bottom}(\mathbf{x}, \mathbf{e}_{top})$
 ▷ quantize with bottom codebook eq 1
- 4: $\mathbf{e}_{bottom} \leftarrow Quantize(\mathbf{h}_{bottom})$
- 5: $\hat{\mathbf{x}} \leftarrow D(\mathbf{e}_{top}, \mathbf{e}_{bottom})$
 ▷ Loss according to eq 2
- 6: $\theta \leftarrow Update(\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}))$

Algorithm 2 Prior training (stage 2)

- 1: $\mathbf{T}_{top}, \mathbf{T}_{bottom} \leftarrow \emptyset$ ▷ training set
- 2: **for** $\mathbf{x} \in$ training set **do**
- 3: $\mathbf{e}_{top} \leftarrow Quantize(E_{top}(\mathbf{x}))$
- 4: $\mathbf{e}_{bottom} \leftarrow Quantize(E_{bottom}(\mathbf{x}, \mathbf{e}_{top}))$
- 5: $\mathbf{T}_{top} \leftarrow \mathbf{T}_{top} \cup \mathbf{e}_{top}$
- 6: $\mathbf{T}_{bottom} \leftarrow \mathbf{T}_{bottom} \cup \mathbf{e}_{bottom}$
- 7: **end for**
- 8: $p_{top} = TrainPixelCNN(\mathbf{T}_{top})$
- 9: $p_{bottom} = TrainCondPixelCNN(\mathbf{T}_{bottom}, \mathbf{T}_{top})$

 ▷ Sampling procedure

- 10: **while** true **do**
- 11: $\mathbf{e}_{top} \sim p_{top}$
- 12: $\mathbf{e}_{bottom} \sim p_{bottom}(\mathbf{e}_{top})$
- 13: $\mathbf{x} \leftarrow D(\mathbf{e}_{top}, \mathbf{e}_{bottom})$
- 14: **end while**



h_{top}



h_{top}, h_{middle}



$h_{top}, h_{middle}, h_{bottom}$



Original

Vector Quantized VAE-2

Samples 1024x1024



Samples diversity



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

DALL-E

Deterministic VQ-VAE posterior

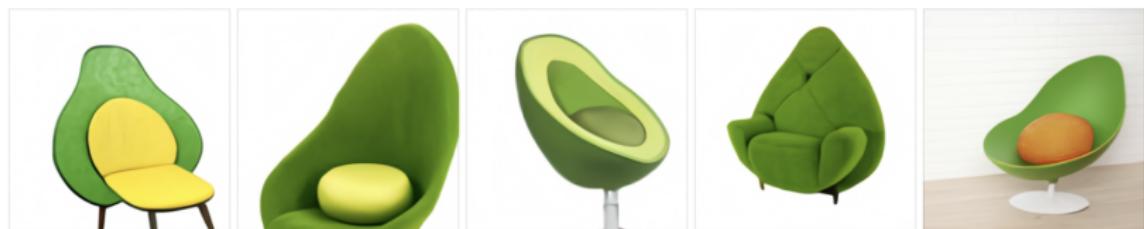
$$q(\hat{z}_{ij} = k^* | \mathbf{x}) = \begin{cases} 1, & \text{for } k^* = \arg \min_k \|[\mathbf{z}_e]_{ij} - \mathbf{e}_k\| \\ 0, & \text{otherwise.} \end{cases}$$

- ▶ 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.