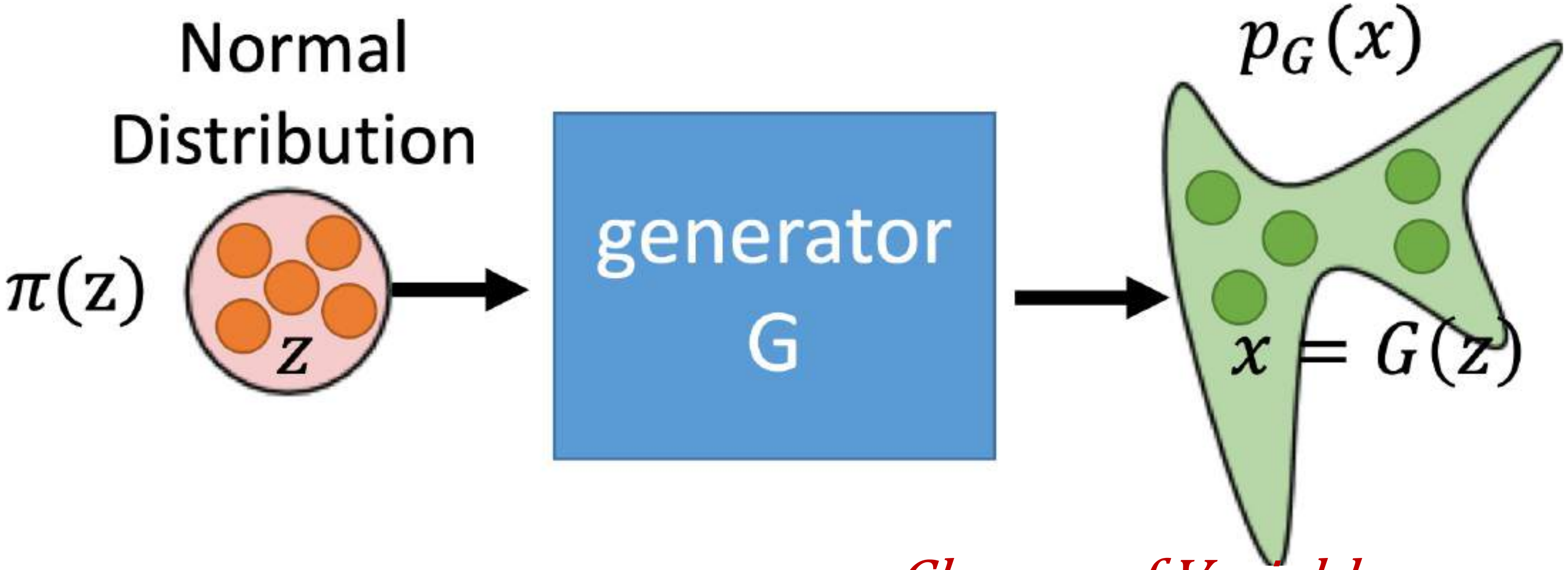
深度学习

Lecture 10 VAE

Pang Tongyao, YMSC

Recaps

Generator: transform a simple distribution to the data distribution



Change of Variable

Generation: sample $z_0 \sim \pi(z), x = G(z_0)$ $\quad p_G(x) = \pi(z)\,|\det(J_{G^{-1}}(x))\;|, z = G^{-1}(x)$

$J_{G^{-1}}$: Jacobian matrix of G^{-1}

Recaps

- Finite Normalizing flows
  - Linear Coupling Layers: [NICE, Dinh et al. 2014]
  - Affine Coupling Layers: [RealNVP, Dinh et al. 2016]
  - Invertible 1×1 Conv Layer: [Glow, Kingma et al. 2018]
- Continuous Normalizing flows
  - [Neural ODE, Chen et al. 2018]
  - Unbiased estimate of trace: [FFJORD, Grathwohl et al. 2018]
- [iResNet, Behrmann et al. 2018]: fixed point iteration approximate inverse; unbiased estimate of trace.

Variational Autoencoders

Probabilistic Model

- The generator $z \to x$ is probabilistic

$$p_\theta(x) = \int p_\theta(x|z)\pi(z)dz$$

if G is deterministic and invertible,

$$p(x) = \int \delta_{G(z)}\pi(z)dz = \int det\left|\frac{dz}{dx}\right| \delta_{G(z)}\pi\left(G^{-1}(x)\right)dx$$

$$= \pi\left(G^{-1}(x)\right)\det|G^{-1}(x)|$$

- Training goal of generative model: maximum likelihood

$$\max\log p_\theta(x)$$

How to calculate $\log\int p_\theta(x|z)\pi(z)dz$?

Gaussian Mixture Model

- z: latent variable
- If z can only take finite number of values, e.g.

$$p(z = i) = \pi_i, i = 1,2, \dots, N.$$

and $p(x|z)$ is Gaussian, then

$$p(x) = \sum_{i=1}^{N} \pi_i \mathcal{N}(\mu_i, \Sigma_i)$$

Gaussian Mixture Model

(Generalized K-means with soft assignments)

Recaps:K-means

- K-means solves the following problem alternatively

$$\min_{\gamma_{ij}\in S, c_j} \frac{1}{2N}\sum_{ij}\gamma_{ij}||x_i - c_j||_2^2$$

K-means Clustering Algorithm

Input: $\mathcal{D} = \{x_i\}_{i=1}^N, x_i \in \mathbb{R}^d$, K(number of clusters)

Initialize: $C = (c_1, c_2, \cdots, c_K)$.

Iterate Until Converge:

- **Update $\boldsymbol{\gamma}$**: $\gamma_{ij} = \begin{cases} 1 & if\ j = argmin_k ||x_i - c_k||_2^2 \\ 0 & else \end{cases}$.
- **Update $\boldsymbol{C}$**: $c_j = (\sum_i \gamma_{ij} x_i)/\sum_i \gamma_{ij}$

Output: Cluster centers C and cluster assignment γ .

Gaussian Mixture Model

GMM optimization

- update assignment $\gamma_{ij} = p(z_i = j|x_i)$ (posterior)

$$\gamma_{ij} := p(z_i = j|x_i) = \frac{p(z_i = j)p(x_i|z_i = j)}{p(x_i)} = \frac{p(z_i = j)p(x_i|z_i = j)}{\sum_j p(z_i = j)p(x_i|z_i = j)} = \frac{\pi_j^{(k)}\mathcal{N}(x_i|\mu_j^{(k)}, \Sigma_j^{(k)})}{\sum_j \pi_j^{(k)}\mathcal{N}(x_i|\mu_j^{(k)}, \Sigma_j^{(k)})}$$

- update center (μ, Σ) and $\pi_i = p(z = i)$ (prior)

$$\pi_j = p(z = j) = \int p(z = j|x)p(x)dx \approx \frac{\sum_{i=1}^{N} p(z = j|x_i)}{N} = \frac{\sum_i \gamma_{ij}}{N}$$

Gaussian Mixture Model

Expectation Maximization (EM) Algorithm (Variational Inference)

$$\sum_i \log\left(\sum_j \gamma_{ij}\pi_j N\left(x_i \middle| \mu_j, \Sigma_j\right)/\gamma_{ij}\right) \geq \sum_i \left(\sum_j \gamma_{ij} \log \frac{\pi_j N\left(x_i \middle| \mu_j, \Sigma_j\right)}{\gamma_{ij}}\right)$$ Jensen's Inequality

- Expectation Step:

$$\gamma_{ij}{}^{(k+1)} = \arg \max_{\{\gamma_{ij}:\sum_j \gamma_{ij}=1, \gamma_{ij}\in[0,1]\}} \sum_i \sum_j \gamma_{ij} \log \pi_i{}^{(k)} \mathcal{N}\left(x_i \middle| \mu_j^{(k)}, \Sigma_j^{(k)}\right) - \gamma_{ij} \log \gamma_{ij}$$

- Maximization Step:

$$\left(\pi_j^{(k+1)}, \mu_j^{(k+1)}, \Sigma_j^{(k+1)}\right) = \arg\max_\theta \sum_i \sum_j \gamma_{ij}^{(k+1)} \log \pi_j N\left(x_i \middle| \mu_j, \Sigma_j\right)$$

Evidence Lower Bound

- **Jensen's Inequality**

$$\log \int p(z)f(z)dz \geq \int p(z)logf(z)dz$$

- **Evidence Lower BOund (ELBO)**

$$\log p(x) = \log \int p(z)p(x|z)dz = \log \int q_x(z)\frac{p(z)p(x|z)}{q_x(z)}dz$$

$$\geq \int q_x(z)log\frac{p(z)p(x|z)}{q_x(z)}dz$$

$$= \int q_x(z)logp(x|z)dz - KL(q_x(z)||p(z))$$

Evidence Lower Bound

- Gap between $\log p(x)$ and ELBO

$$\log p(x) - \int q_x(z) logp(x|z)dz + KL(q_x(z)||p(z))$$
$$= \int q_x(z)\log \mathrm{p(x)dz} \; - \int q_x(z) logp(x|z)dz + KL(q_x(z)||p(z))$$
$$= \int q_x(z)[\log p(x) - \log p(x|z) - \log p(z)\,]dz + \int q_x(z) log q_x(z)dz$$
$$= -\int q_x(z) \log \frac{p(x|z)p(z)}{p(x)} dz + \int q_x(z) log q_x(z)dz$$
$$= -\int q_x(z) \log p(z|x)dz + \int q_x(z) log q_x(z)dz = KL(q_x(z)||p(z|x))$$

- When $q_x(z) = p(z|x)$, the gap is closed.

(In GMM, $q_{x_i}(z = j) = \gamma_{ij} = p(z = j|x_i)$)

Evidence Lower Bound

- EM Algorithm: Alternatively Maximize ELBO

$$\max_{\phi,\psi} \int q_{\phi(x)}(z) log p_{\psi}(x|z) dz - KL(q_{\phi(x)}(z)||p(z))$$

  - Expectation: $\max_{\phi} ELBO \Rightarrow q_{\phi(x)}(z) = p(z|x)$

  However, we can't obtain $p(z|x)$ in general; propose a parameterized distribution $q_{\phi(x)}(z)$ we know we can work with easily to approximate $p(z|x)$.

  - Maximization: $\max_{\psi} ELBO$

- For Gaussian Mixture Models, the negative log-likelihood (NLL) function is non-convex, and directly minimizing it can lead to poor performance due to local optima. In contrast, the EM algorithm guarantees a monotonic decrease in the NLL at each iteration.

Evidence Lower Bound

- Why ELBO?

$$p(x) = \mathbb{E}_{z\sim p(z)} p(x|z)$$

Samples from $p(z)$ are usually not informative! Direct optimization via Monte Carlo sampling leads to high-variance estimates, and is often too noisy to train.

- Importance Sampling

$$\mathbb{E}_p[f(x)] = \int p(x)f(x)dx = \int p(x)\frac{q(x)}{q(x)}f(x)dx = \int q(x)[f(x)\frac{p(x)}{q(x)}]dx = \mathbb{E}_q[f(x)\frac{p(x)}{q(x)}]$$

Evidence Lower Bound

- Importance Sampling: we want to sample z compatible with x. It works well when $q(x)/p(x) \approx f(x)$ (with low variance).
- For VAE

$$p(x) = \int p(z)p(x|z)dz = \int q_x(z)\frac{p(z)p(x|z)}{q_x(z)}dz$$

When $q_x(z) = p(z|x)$, the Monte Carlo sampling variance is accurate with only one sample ($\frac{p(z)p(x|z)}{q_x(z)} = p(x)$ is constant irrelevant to z.) The equality also holds in **Jensen's Inequality:**

$$\log \int q_x(z)\frac{p(z)p(x|z)}{q_x(z)}dz = \int q_x(z)log\frac{p(z)p(x|z)}{q_x(z)}dz$$

(The right hand side is much easier to compute in many cases.)

VAEs

- Parametrize $q_x(z) = \mathcal{N}(\mu_\theta(x), \sigma_\theta(x))$ and $p_\psi(x|z) = \mathcal{N}(\psi(z), \beta I)$

$$ELBO = \int q_x(z) log p(x|z) dz - KL(q_x(z)||p(z))$$

$$= -\frac{1}{2\beta}\mathbb{E}_{q_x(z)}||\psi(z) - x||_2^2 - KL(q_x(z)||p(z))$$

$$= -\frac{1}{2\beta}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}||\psi(z) - x||_2^2 - KL(q_x(z)||p(z))$$

where $z = \mu_\theta(x) + \sigma_\theta(x)\epsilon$.

Let $p(x) = N(\mu_1, \sigma_1)$ and $q(x) = N(\mu_2, \sigma_2)$.

$$KL(p, q) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

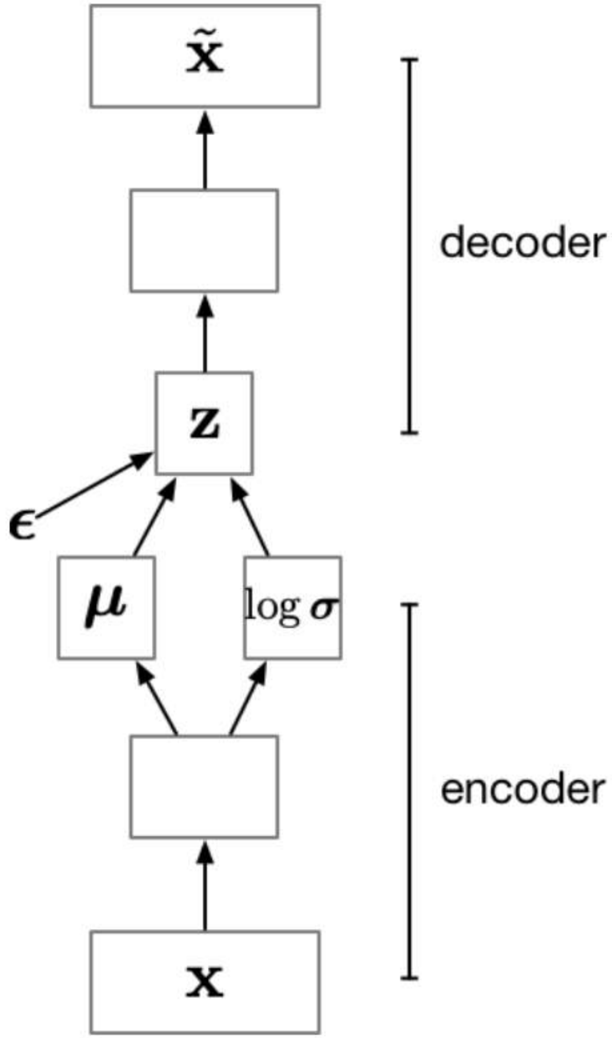
If $\mathrm{p}(z) = \mathcal{N}(0, I)$,

$$KL(q_x(z)||p(z)) = \frac{1}{2}||\mu_\theta(x)||_2^2 + \frac{1}{2}\sigma_\theta^2(x) - \log\sigma_\theta(x) + c$$

VAEs

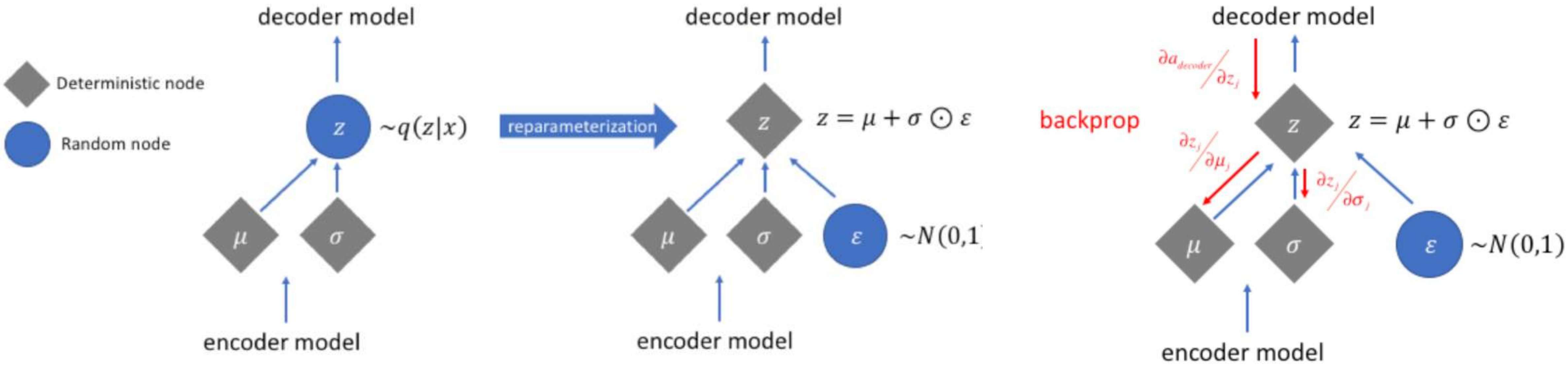
Two terms in ELBO:

- The reconstruction term $\mathbb{E}_{q_x(z)}||\psi(z) - x||_2^2$ is minimized if $q_x(z)$ is a point mass, which means the generator is deterministic.
- However, the penalty term $KL(q_x(z)||p(z))$ prevents $q_x(z)$ to be a point mass ($KL(q_x(z)||p(z))$ is infinity if $q_x(z)$ is a point mass). So it encourage the latent code to be stochastic.



VAEs

- Training skill: Reparameterization



From JEREMY JORDAN's Blog: https://www.jeremyjordan.me/variational-autoencoders/

Interpretable latent code

Varying z_1: degree of smile

Varying z_2: head pose

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

VAEs

- Blurry samples
- The gap between $q_x(z)$ and the true posterior $p(z|x)$ may be large for assumed Gaussian latent code distribution.

More informative $p(z)$ and tractable $p(z|x)$?

VQ-VAE

- Recap Gaussian mixture: discrete latent code
- Vector-Quantization VAE (VQ-VAE, DeepMind, NIPS 2017) proposed one-hot latent code distribution:

$$q(z = k|x) = \begin{cases} 1 & \text{for } \mathrm{k} = \mathrm{argmin}_j \|z_e(x) - e_j\|_2, \\ 0 & \text{otherwise} \end{cases}$$

- Quantize encoder output

$$\text{Quantize}(E(\mathbf{x})) = \mathbf{e}_k \quad \text{where } k = \arg\min_j ||E(\mathbf{x}) - \mathbf{e}_j||$$

“Neural Discrete Representation Learning” , DeepMind, 2018

VQ-VAE

How to learn the dictionary?

"Sampling" from the posterior is to compute the nearest code.

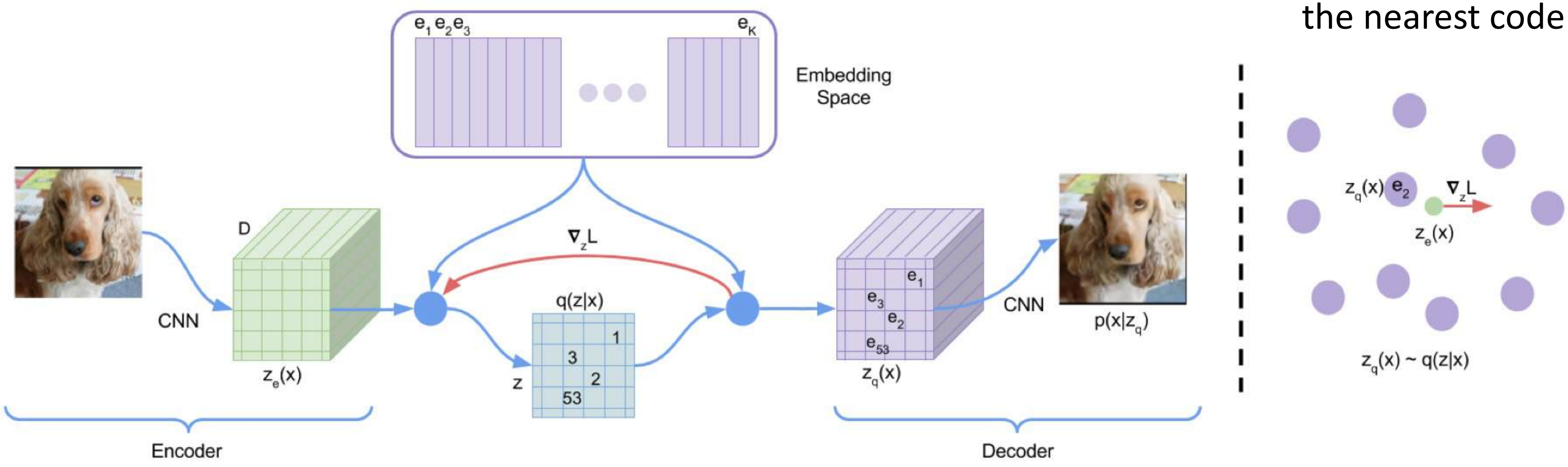


Figure 1: Left: A figure describing the VQ-VAE. Right: Visualisation of the embedding space. The output of the encoder $z(x)$ is mapped to the nearest point e_2. The gradient $\nabla_z L$ (in red) will push the encoder to change its output, which could alter the configuration in the next forward pass.

VQ-VAE

- Loss

$$\mathcal{L}(\mathbf{x}, D(\mathbf{e})) = ||\mathbf{x} - D(\mathbf{e})||_2^2 + ||sg[E(\mathbf{x})] - \mathbf{e}||_2^2 + \beta||sg[\mathbf{e}] - E(\mathbf{x})||_2^2$$

- sg(·): stop gradient operator
- update dictionary (similar as K-means)

$$e_i = \frac{1}{n_i} \sum_j^{n_i} z_{i,j}.$$

work on minibatch: exponential moving average

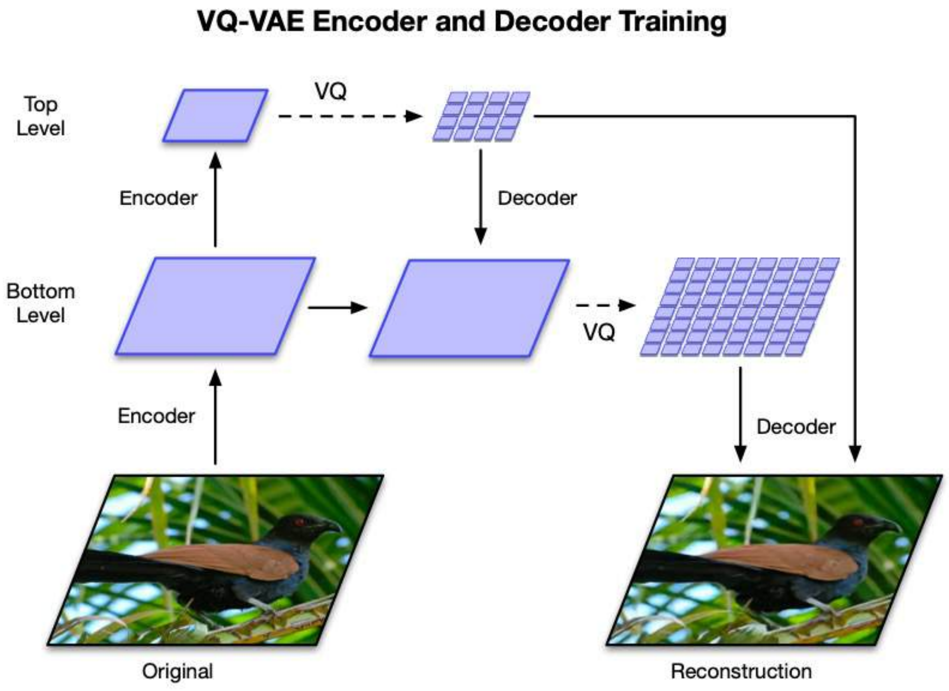
$$N_i^{(t)} := N_i^{(t-1)} * \gamma + n_i^{(t)}(1 - \gamma)$$
$$m_i^{(t)} := m_i^{(t-1)} * \gamma + \sum_j z_{i,j}^{(t)}(1 - \gamma)$$

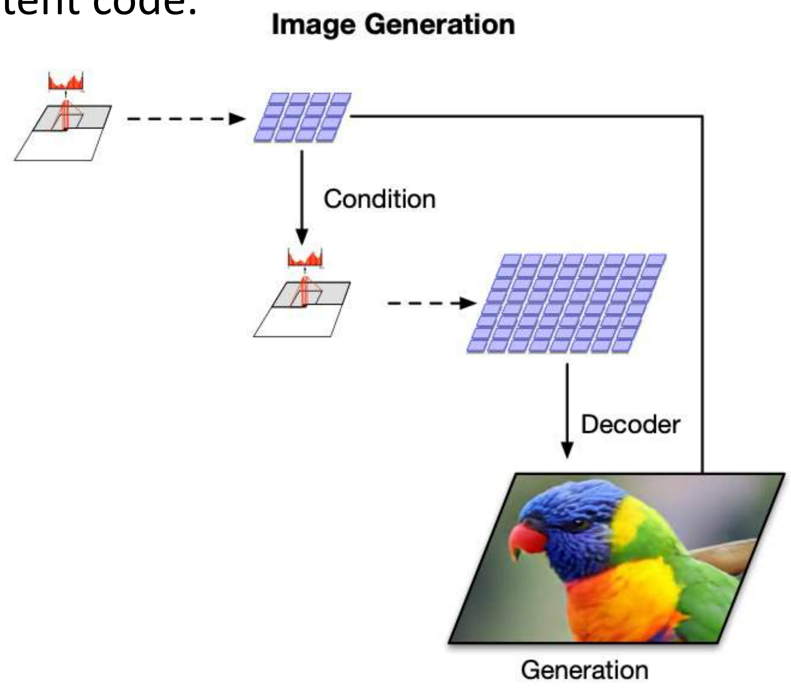
$$e_i^{(t)} := \frac{m_i^{(t)}}{N_i^{(t)}},$$

VQ-VAE2

Hierarchical structure

For sampling, autoregressive models are trained to approximate the prior distribution of the discrete latent code.





“Generating Diverse High-Fidelity Images with VQ-VAE-2”, DeepMind, 2019

VQ-VAE2

Algorithm 1 VQ-VAE training (stage 1)

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

$\mathbf{h}_{top} \leftarrow E_{top}(\mathbf{x})$

▷ quantize with top codebook eq 1

$\mathbf{e}_{top} \leftarrow Quantize(\mathbf{h}_{top})$

$\mathbf{h}_{bottom} \leftarrow E_{bottom}(\mathbf{x}, \mathbf{e}_{top})$

▷ quantize with bottom codebook eq 1

$\mathbf{e}_{bottom} \leftarrow Quantize(\mathbf{h}_{bottom})$

$\hat{\mathbf{x}} \leftarrow D(\mathbf{e}_{top}, \mathbf{e}_{bottom})$

▷ Loss according to eq 2

$\theta \leftarrow Update(\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}))$

Algorithm 2 Prior training (stage 2)

$\mathbf{T}_{top}, \mathbf{T}_{bottom} \leftarrow \emptyset$ ▷ training set

for $\mathbf{x} \in$ training set **do**

$\quad \mathbf{e}_{top} \leftarrow Quantize(E_{top}(\mathbf{x}))$

$\quad \mathbf{e}_{bottom} \leftarrow Quantize(E_{bottom}(\mathbf{x}, \mathbf{e}_{top}))$

$\quad \mathbf{T}_{top} \leftarrow \mathbf{T}_{top} \cup \mathbf{e}_{top}$

$\quad \mathbf{T}_{bottom} \leftarrow \mathbf{T}_{bottom} \cup \mathbf{e}_{bottom}$

end for

$p_{top} = \texttt{TrainPixelCNN}(\mathbf{T}_{top})$

$p_{bottom} = \texttt{TrainCondPixelCNN}(\mathbf{T}_{bottom}, \mathbf{T}_{top})$

▷ Sampling procedure

while true **do**

$\quad \mathbf{e}_{top} \sim p_{top}$

$\quad \mathbf{e}_{bottom} \sim p_{bottom}(\mathbf{e}_{top})$

$\quad \mathbf{x} \leftarrow D(\mathbf{e}_{top}, \mathbf{e}_{bottom})$

end while

VQ-VAE2

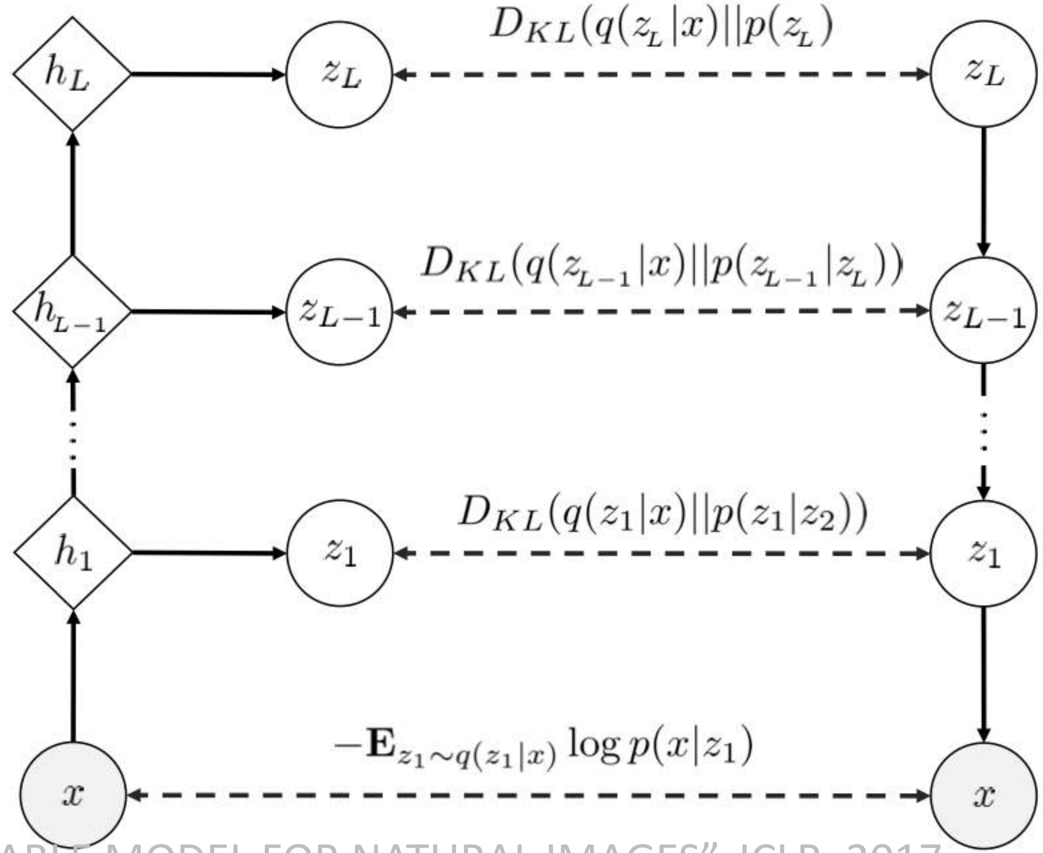
Figure 3: Reconstructions from a hierarchical VQ-VAE with three latent maps (top, middle, bottom). The rightmost image is the original. Each latent map adds extra detail to the reconstruction. These latent maps are approximately 3072x, 768x, 192x times smaller than the original image (respectively).

VQ-VAE2

VQ-VAE (Proposed) BigGAN deep

Hierarchical PixelVAE

Hierarchical latent space decomposition



"PIXELVAE: A LATENT VARIABLE MODEL FOR NATURAL IMAGES", ICLR, 2017

Latent Variable distribution

$$p(z_1, \cdots, z_L) = p(z_L)p(z_{L-1}|z_L) \cdots p(z_1|z_2)$$

Prior: Markov assumption

$$q(z_1, \cdots, z_L|x) = q(z_1|x) \cdots q(z_L|x)$$

Encoder: conditional independence assumption

ELBO

Decoder: Markov assumption

$$-L(x,q,p) = -E_{z_1 \sim q(z_1|x)} \log \boxed{p(x|z_1)} + D_{KL}(q(z_1, \cdots z_L|x)||p(z_1, \cdots, z_L))$$

$$= -E_{z_1 \sim q(z_1|x)} \log p(x|z_1) + \int_{z_1, \cdots, z_L} \prod_{j=1}^{L} q(z_j|x) \sum_{i=1}^{L} \log \frac{q(z_i|x)}{p(z_i|z_{i+1})} dz_1 ... dz_L$$

$$= -E_{z_1 \sim q(z_1|x)} \log p(x|z_1) + \sum_{i=1}^{L} \int_{z_1, \cdots, z_L} \prod_{j=1}^{L} q(z_j|x) \log \frac{q(z_i|x)}{p(z_i|z_{i+1})} dz_1 ... dz_L$$

$$= -E_{z_1 \sim q(z_1|x)} \log p(x|z_1) + \sum_{i=1}^{L} \int_{z_i, z_{i+1}} q(z_{i+1}|x) q(z_i|x) \log \frac{q(z_i|x)}{p(z_i|z_{i+1})} dz_i dz_{i+1}$$

$$= -E_{z_1 \sim q(z_1|x)} \log p(x|z_1) + \sum_{i=1}^{L} \mathbf{E}_{z_{i+1} \sim q(z_{i+1}|x)} \left[D_{KL}(q(z_i|x)||p(z_i|z_{i+1})) \right]$$

MNIST

Model	NLL Test
DRAW (Gregor et al., 2016)	≤ 80.97
Discrete VAE (Rolfe, 2016)	$= 81.01$
IAF VAE (Kingma et al., 2016)	≈ 79.88
PixelCNN (van den Oord et al., 2016a)	$= 81.30$
PixelRNN (van den Oord et al., 2016a)	$= 79.20$
Convolutional VAE	≤ 87.41
PixelVAE	≤ 80.64
Gated PixelCNN (our implementation)	$= 80.10$
Gated PixelVAE	≈ 79.48 (≤ 80.02)
Gated PixelVAE without upsampling	$\approx$ **79.02** (≤ 79.66)

Model	NLL Validation (Train)
Convolutional DRAW (Gregor et al., 2016)	≤ 4.10 (4.04)
Real NVP (Dinh et al., 2016)	$= 4.01$ (3.93)
PixelRNN (van den Oord et al., 2016a)	$= 3.63$ (3.57)
Gated PixelCNN (van den Oord et al., 2016b)	$=$ **3.57** (3.48)
Hierarchical PixelVAE	≤ 3.66 (3.59)

Table 2: Model performance on 64x64 ImageNet.

Figure 6: Samples from hierarchical PixelVAE on the 64x64 ImageNet dataset.

more globally coherent than samples from PixelRNN.

NVAE

- Hierarchical Architecture

$$q(z) = \prod_{i=1}^{l} q(z_l|z_{<l}), \quad p(z|x) = \prod_{i=1}^{l} p(z_l|z_{<l}, x)$$

Exact decomposition: no additional assumption

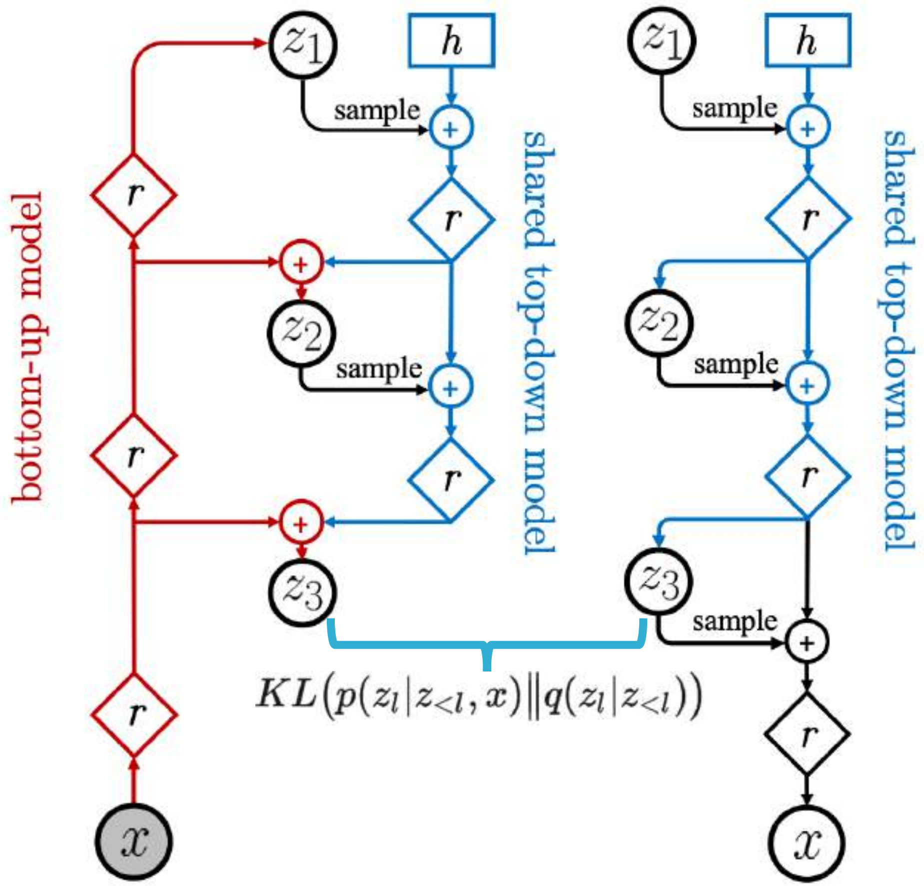
$$q(z_l|z_{<l}) = \mathcal{N}\left(z_l; \mu(z_{<l}), \sigma^2(z_{<l})\right)$$
$$p(z_l|z_{<l}, x) = \mathcal{N}\left(z_l; \mu(z_{<l}) + \Delta\mu(z_{<l}, x), \sigma^2(z_{<l}) \otimes \Delta\sigma^2(z_{<l}, x)\right)$$

$$KL(p(z|x)\|q(z)) = KL(p(z_1|x)\|q(z_1)) + \sum_{l=2}^{L} \mathbb{E}_{p(z_{<l}|x)}\left[KL(p(z_l|z_{<l}, x)\|q(z_l|z_{<l}))\right]$$

$$KL(p(z_l|z_{<l}, x)\|q(z_l|z_{<l})) = \frac{1}{2}\sum_{i=1}^{|z_l|}\left(\frac{\Delta\mu_{(i)}^2}{\sigma_{(i)}^2} + \Delta\sigma_{(i)}^2 - \log\Delta\sigma_{(i)}^2 - 1\right)$$

“NVAE: A Deep Hierarchical Variational Autoencoder” , NVIDIA, 2020

NVAE



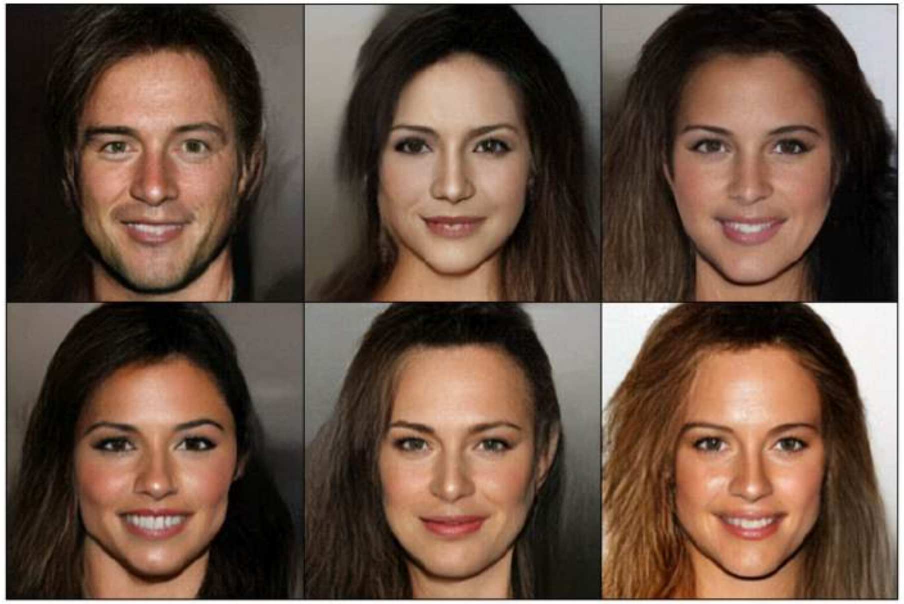
(a) Bidirectional Encoder (b) Generative Model

Method	MNIST 28×28	CIFAR-10 32×32	ImageNet 32×32	CelebA 64×64	CelebA HQ 256×256	FFHQ 256×256
NVAE w/o flow	**78.01**	2.93	-	2.04	-	0.71
NVAE w/ flow	78.19	**2.91**	3.92	**2.03**	**0.70**	**0.69**
VAE Models with an Unconditional Decoder						
BIVA [36]	78.41	3.08	3.96	2.48	-	-
IAF-VAE [4]	79.10	3.11	-	-	-	-
DVAE++ [20]	78.49	3.38	-	-	-	-
Conv Draw [42]	-	3.58	4.40	-	-	-
Flow Models without any Autoregressive Components in the Generative Model						
VFlow [59]	-	2.98	-	-	-	-
ANF [60]	-	3.05	3.92	-	0.72	-
Flow++ [61]	-	3.08	**3.86**	-	-	-
Residual flow [50]	-	3.28	4.01	-	0.99	-
GLOW [62]	-	3.35	4.09	-	1.03	-
Real NVP [63]	-	3.49	4.28	3.02	-	-
VAE and Flow Models with Autoregressive Components in the Generative Model						
δ-VAE [25]	-	2.83	3.77	-	-	-
PixelVAE++ [35]	78.00	2.90	-	-	-	-
VampPrior [64]	78.45	-	-	-	-	-
MAE [65]	77.98	2.95	-	-	-	-
Lossy VAE [66]	78.53	2.95	-	-	-	-
MaCow [67]	-	3.16	-	-	0.67	-
Autoregressive Models						
SPN [68]	-	-	3.85	-	0.61	-
PixelSNAIL [34]	-	2.85	3.80	-	-	-
Image Transformer [69]	-	2.90	3.77	-	-	-
PixelCNN++ [70]	-	2.92	-	-	-	-
PixelRNN [41]	-	3.00	3.86	-	-	-
Gated PixelCNN [71]	-	3.03	3.83	-	-	-

(d) CelebA HQ ($t = 0.6$)

(e) FFHQ ($t = 0.5$)

NVAE generates diverse high-quality samples.



(f) MaCow [67] trained on CelebA HQ ($t = 0.7$)

(g) Glow [62] trained on CelebA HQ ($t = 0.7$)

Summary

- VAE:

$$\max ELBO = \int q_x(z) log p(x|z) dz - KL(q_x(z)||p(z))$$

GAP between log likelihood and ELBO is

$$KL(q_x(z)||p(z|x))$$

- Encoder: Hierarchical structure, discrete code distribution

latent code prior: Gaussian prior, auto-regressive model