10-423/10-623 Generative AI

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Variational Autoencoders (VAEs)

Matt Gormley
Lecture 9
Feb. 14, 2024

Reminders

- Homework 2: Generative Models of Images
  - Out: Thu, Feb 8
  - Due: Tue, Feb 20 at 11:59pm

Recall…

KL DIVERGENCE

KL Divergence

Recall…

- *Definition*: for two distributions q(x) and p(x) over x ∈ $\mathcal{X}$, the **KL Divergence** is:

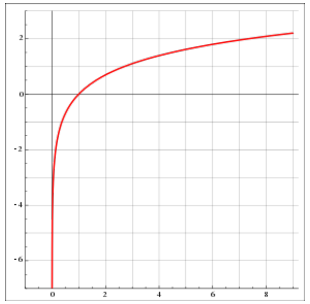
$$\text{KL}(q||p) = E_{q(x)}\left[\log \frac{q(x)}{p(x)}\right] = \begin{cases} \sum_x q(x) \log \frac{q(x)}{p(x)} \\ \int_x q(x) \log \frac{q(x)}{p(x)} dx \end{cases}$$

- *Properties*:
  - KL(q || p) measures the **proximity** of two distributions q and p
  - KL is **not** symmetric: KL(q || p) ≠ KL(p || q)
  - KL is minimized when q(x) = p(x) for all x ∈ $\mathcal{X}$

$$\text{KL}(q||p) = E_{q(x)}\left[\log \frac{q(x)}{p(x)}\right]$$

KL Divergence

Recall…



Understanding the Behavior of KL as an objective function

Example 1: Keeping all else constant, consider the effect of a particular x' on KL(q || p)

x'	q(x')	p(x')	q(x') log(q(x')/p(x'))	effect on KL(q || p)
1	0.9	0.9	0	no increase
2	0.9	0.1	1.97	big increase
3	0.1	0.9	-0.21	little decrease
4	0.1	0.1	0	little decrease

KL **does** insist on good approximations for values that have **high** probability in q

KL **does not** insist on good approximations for values that have **low** probability in q

Example 2: Which q distribution minimizes KL(q || p)?

$$\mathbf{p} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix} \quad \mathbf{q}^{(1)} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \quad \mathbf{q}^{(2)} = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix} \quad \mathbf{q}^{(3)} = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$$

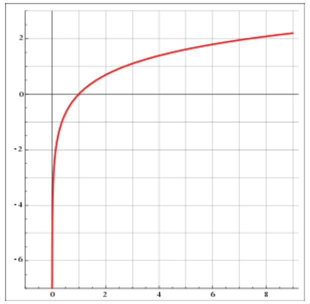
Q: If we're minimizing KL, why not return $q^{(3)}$?

A: Because it's not a distribution!

Recall…

$\text{KL}(q||p) = E_{q(x)}\left[\log \frac{q(x)}{p(x)}\right]$

KL Divergence

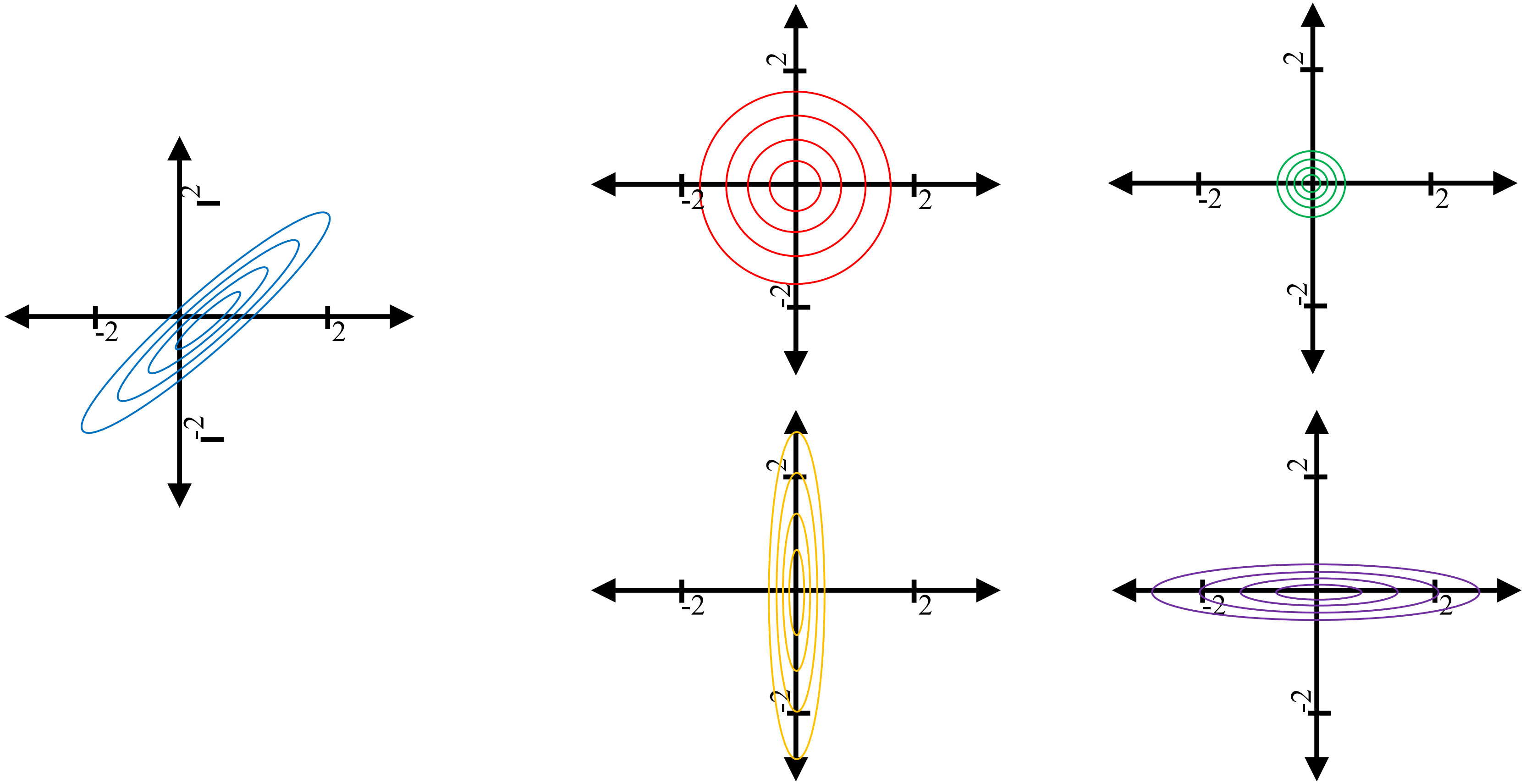


Understanding the Behavior of KL as an objective function

Example 3: Which q distribution minimizes KL(q || p)?

$$p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu} = [0,0]^T, \boldsymbol{\Sigma})$$

$$q(x_1, x_2) = \mathcal{N}_1(x_1 \mid \mu_1, \sigma_1^2)\mathcal{N}_2(x_2 \mid \mu_2, \sigma_2^2)$$



VARIATIONAL DIFFUSION MODELS AND VARITIONAL AUTOENCODERS (VAES)

Diffusion Models

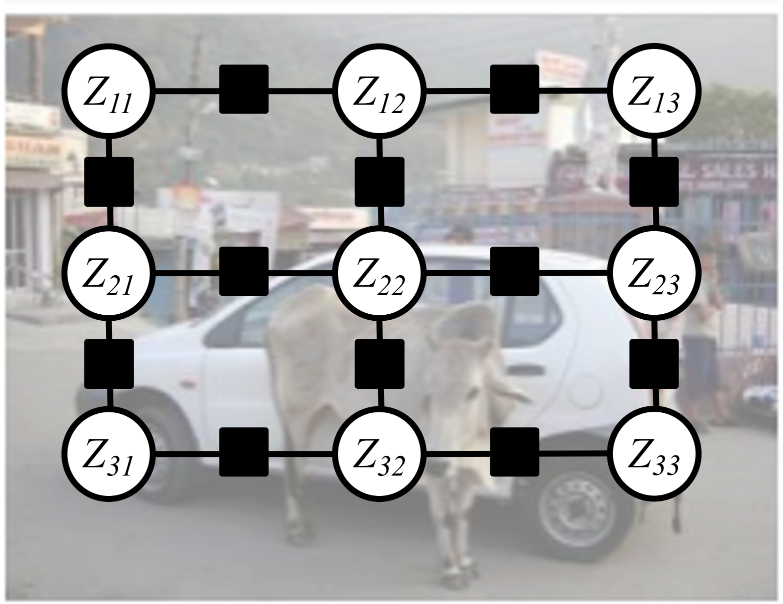
- Next we will consider (1) **diffusion models** and (2) **variational autoencoders (VAEs)**
  - Although VAEs came first, we're going to dive into diffusion models since they will receive more of our attention
- The steps in defining these models is roughly:
  - Define a probability distribution involving Gaussian noise
  - Use a variational lower bound as an objective function
  - Learn the parameters of the probability distribution by optimizing the objective function
- So what is a variational lower bound?

HIGH-LEVEL INTRO TO VARIATIONAL INFERENCE

Variational Inference

Problem:

– For observed variables **x** and latent variables **z**, estimating the posterior p(**z** | **x**) is intractable

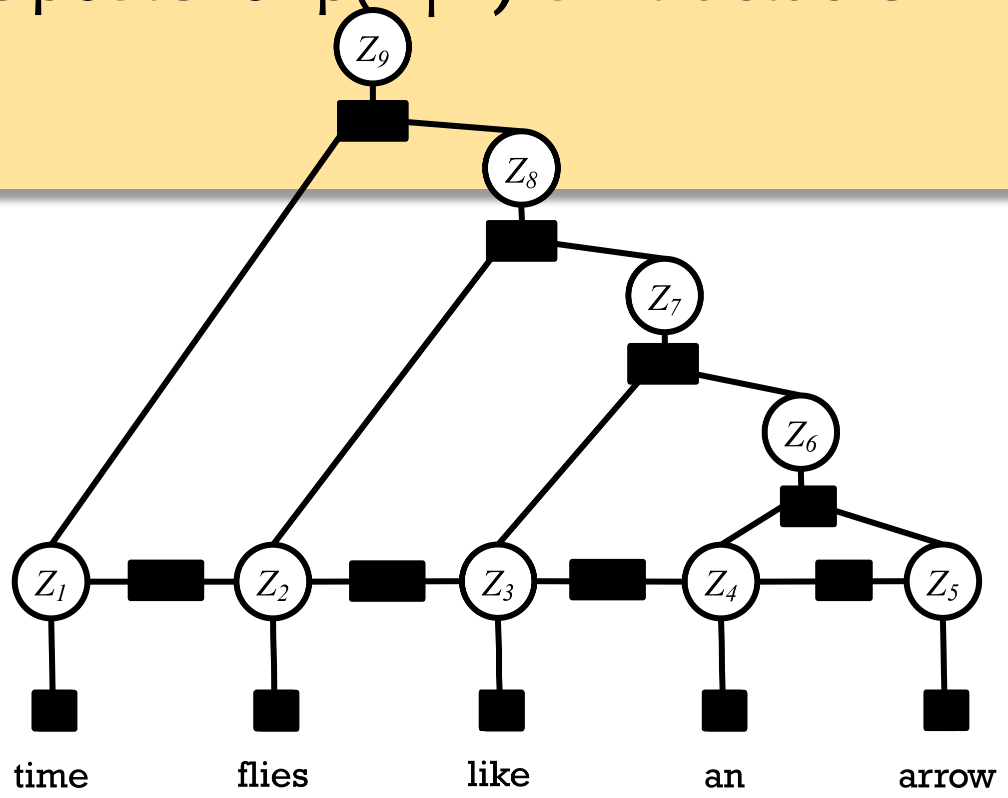


Narrative adapted from Jason Eisner's High-Level Explanation of VI:
https://www.cs.jhu.edu/~jason/tutorials/variational.html

Variational Inference

Problem:

– For observed variables **x** and latent variables **z**, estimating the posterior p(**z** | **x**) is intractable

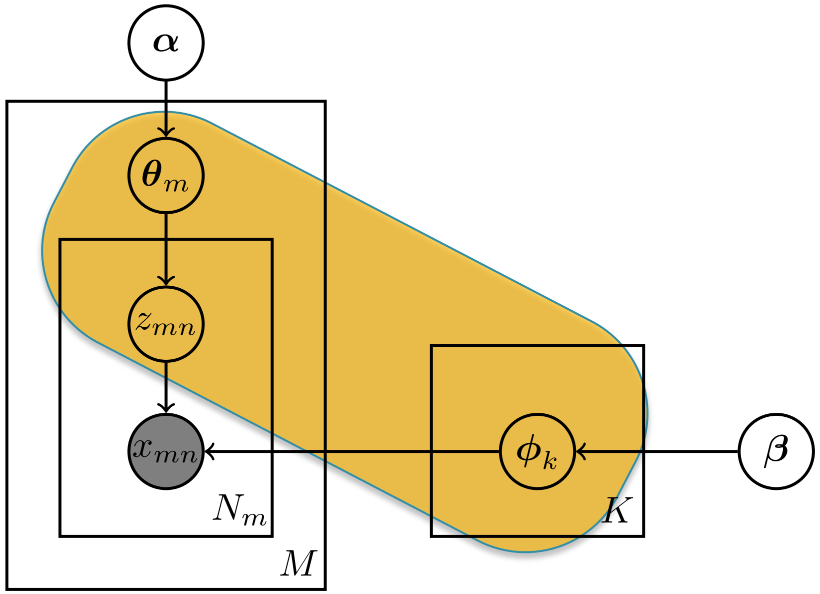


Narrative adapted from Jason Eisner's High-Level Explanation of VI:
https://www.cs.jhu.edu/~jason/tutorials/variational.html

Variational Inference

Problem:

- For observed variables **x** and latent variables **z**, estimating the posterior p(**z** | **x**) is intractable
- For training data **x** and parameters **z**, estimating the posterior p(**z** | **x**) is intractable



Narrative adapted from Jason Eisner's High-Level Explanation of VI: https://www.cs.jhu.edu/~jason/tutorials/variational.html

Variational Inference

Problem:

- For observed variables **x** and latent variables **z**, estimating the posterior p(**z** | **x**) is intractable
- For training data **x** and parameters **z**, estimating the posterior p(**z** | **x**) is intractable

Solution:

- Approximate p(**z** | **x**) with a simpler q(**z**)
- Typically q(**z**) has more independence assumptions than p(**z** | **x**) – fine b/c q(**z**) is tuned for a specific **x**
- **Key idea:** pick a single q(**z**) from some family Q that best approximates p(**z** | **x**)

Narrative adapted from Jason Eisner's High-Level Explanation of VI: https://www.cs.jhu.edu/~jason/tutorials/variational.html

Variational Inference

Terminology:

- $q(\mathbf{z})$: the **variational approximation**
- Q: the **variational family**
- Usually $q_\theta(\mathbf{z})$ is parameterized by some θ called **variational parameters**
- Usually $p_\alpha(\mathbf{z} \mid \mathbf{x})$ is parameterized by some fixed α – we'll call them the parameters

Example Algorithms:

- mean-field variational inference
- loopy belief propagation
- tree-reweighted belief propagation
- expectation propagation

Narrative adapted from Jason Eisner's High-Level Explanation of VI: https://www.cs.jhu.edu/~jason/tutorials/variational.html

Variational Inference

Is this trivial?

- Note: We are not defining a new distribution simple $q_\theta(\mathbf{z} \mid \mathbf{x})$, there is one simple $q_\theta(\mathbf{z})$ for each $p_\alpha(\mathbf{z} \mid \mathbf{x})$
- Consider the MCMC equivalent of this:
  - you could draw samples $z^{(i)} \sim p(\mathbf{z} \mid \mathbf{x})$
  - then train some simple $q_\theta(\mathbf{z})$ on $z^{(1)}, z^{(2)}, \ldots, z^{(N)}$
  - hope that the sample adequately represents the posterior for the given **x**
- How is VI different from this?
  - VI doesn't require sampling
  - VI is fast and deterministic
  - Why? b/c we choose an objective function (KL divergence) that defines which q_θ best approximates p_α, and exploit the special structure of q_θ to optimize it

Narrative adapted from Jason Eisner's High-Level Explanation of VI: https://www.cs.jhu.edu/~jason/tutorials/variational.html

Variational Inference

V.I. offers a new design decision

- Choose the distribution $p_\alpha(\mathbf{z} \mid \mathbf{x})$ that you really want, i.e. don't just simpify it to make it computationally convenient
- Then design a the structure of another distribution $q_\theta(\mathbf{z})$ such that V.I. is efficient

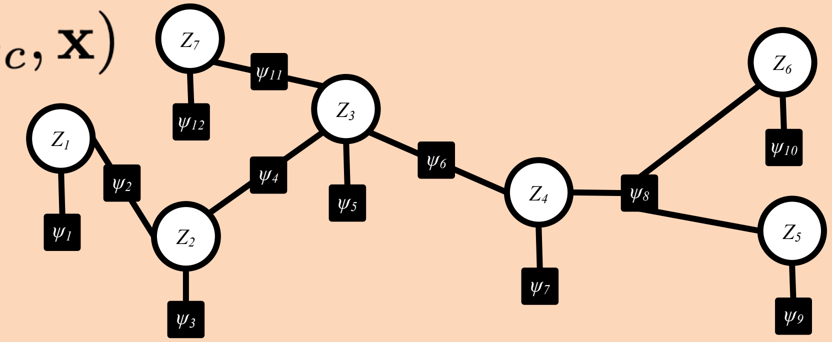
Narrative adapted from Jason Eisner's High-Level Explanation of VI:
https://www.cs.jhu.edu/~jason/tutorials/variational.html

THE MEAN FIELD APPROXIMATION

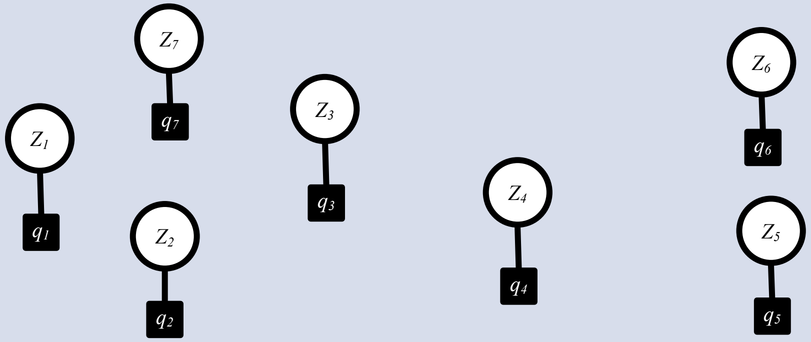
Mean Field Approximation

The **mean field approximation** assumes our variational approximation $q_\theta(\mathbf{z})$ treats each variable as independent

$$p_\alpha(\mathbf{z} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{z}_c, \mathbf{x})$$



$$q_\theta(\mathbf{z}) = \prod_{t=1}^{T} q_t(z_t)$$

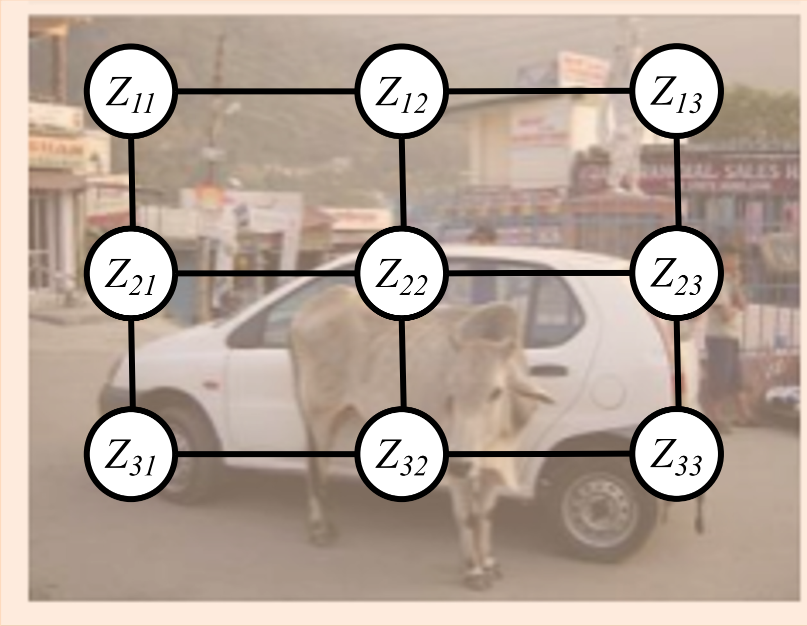


Mean Field Approximation

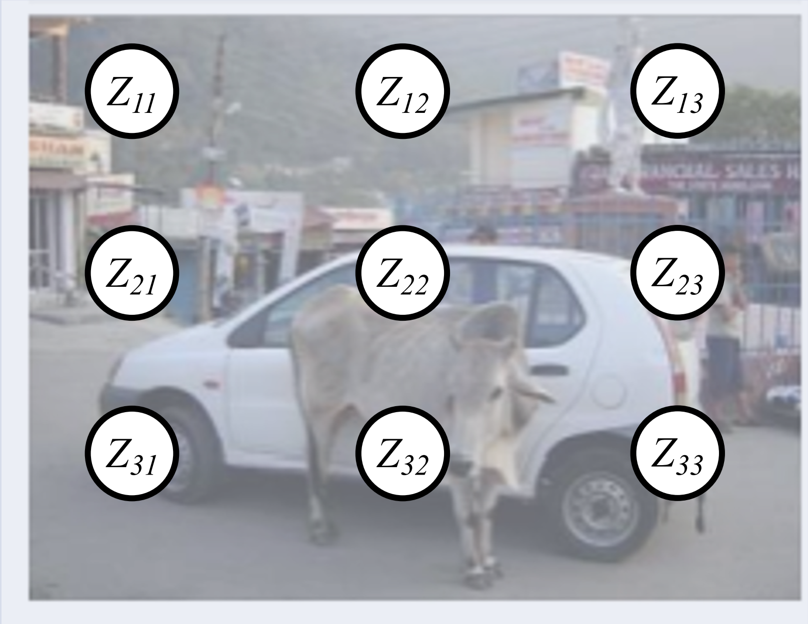
The **mean field approximation** assumes our variational approximation $q_\theta(\mathbf{z})$ treats each variable as independent

Ising Model

$$p_\alpha(\mathbf{z} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{z}_c, \mathbf{x})$$



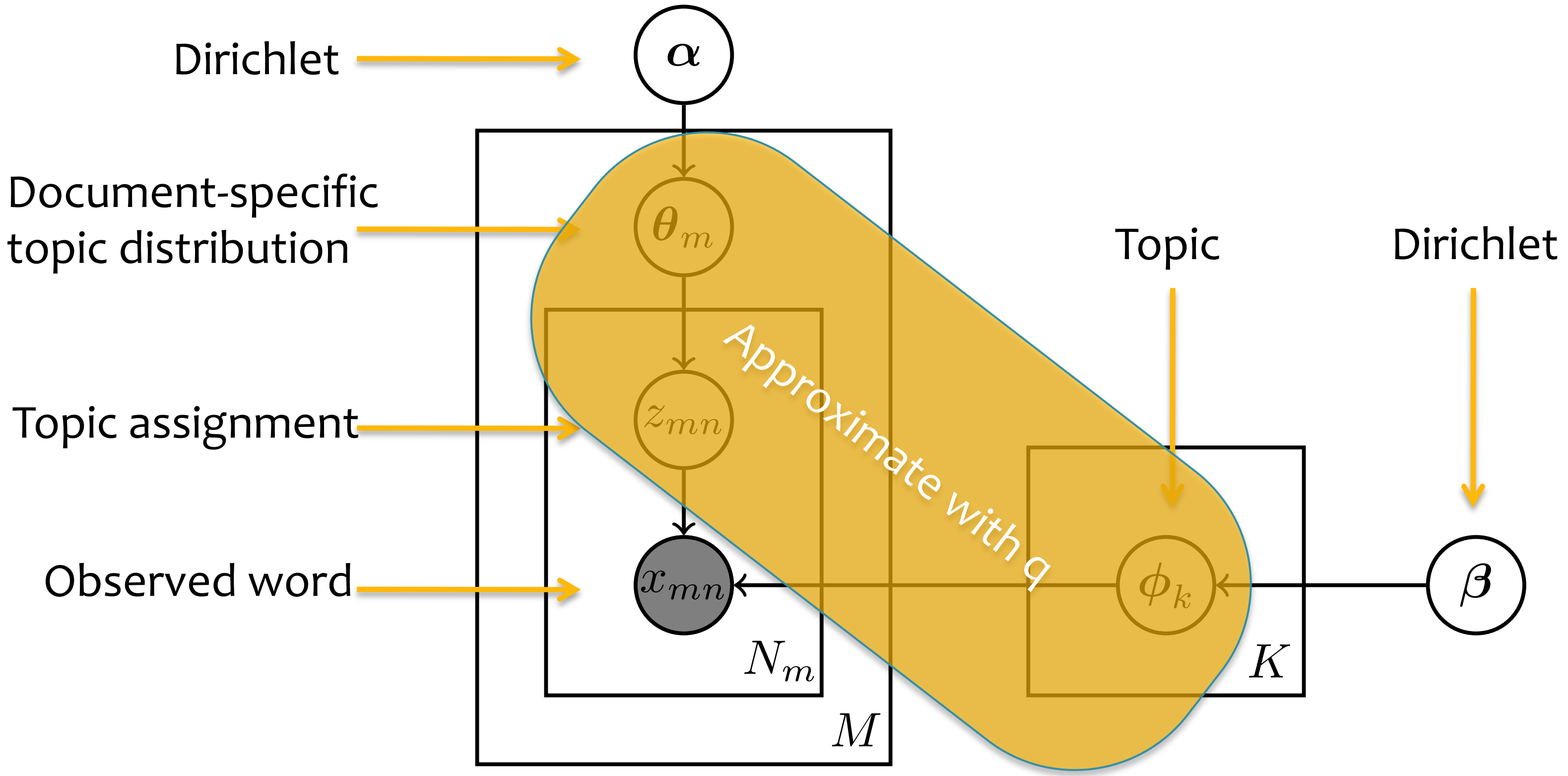
$$q_\theta(\mathbf{z}) = \prod_{t=1}^{T} q_t(z_t)$$



Mean Field Approximation

Latent Dirichlet Allocation (LDA)

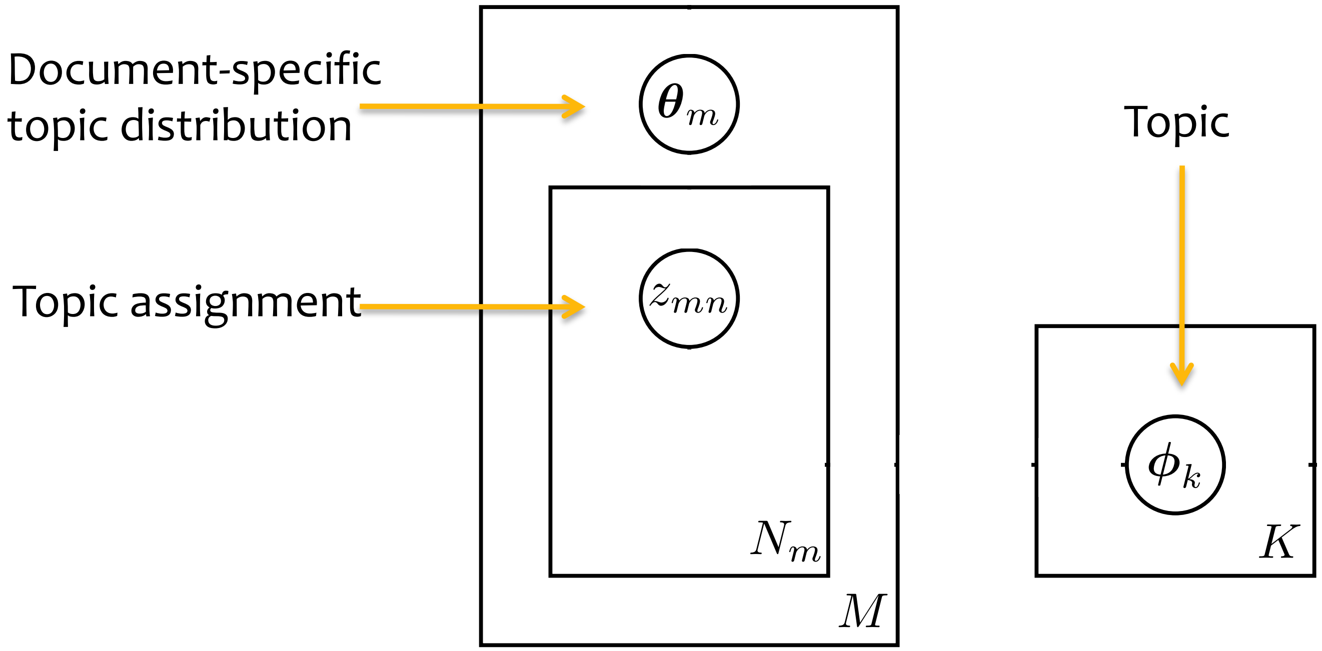
- Uncollapsed Variational Inference, aka. Explicit V.I. (original distribution)



Mean Field Approximation

Latent Dirichlet Allocation (LDA)

- Uncollapsed Variational Inference, aka. Explicit V.I. (mean field variational approximation)



MEAN FIELD VARIATIONAL INFERENCE

Two Cases for Intractability

Suppose we want to work with p(z|x)

- *Case 1*:

  given a **joint distribution** p(x, z)

$$\Rightarrow p(z \mid x) = \frac{p(x, z)}{p(x)}$$

we assume p(x) is intractable

- *Case 2*:

  give **factor graph** and potentials

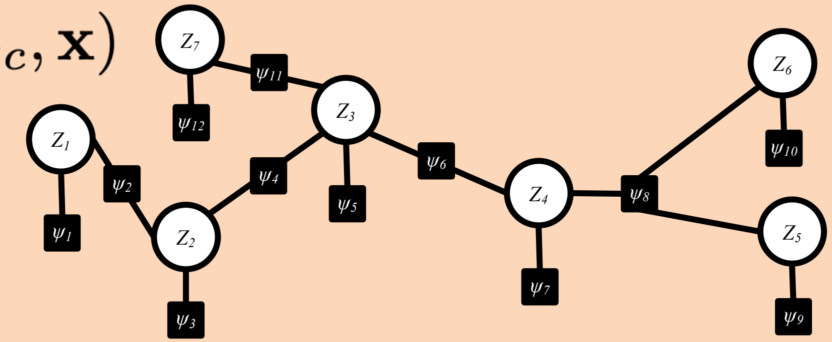
$$\Rightarrow p(z \mid x) = \frac{\tilde{p}(x, z)}{Z(x)}$$

we assume Z(x) is intractable

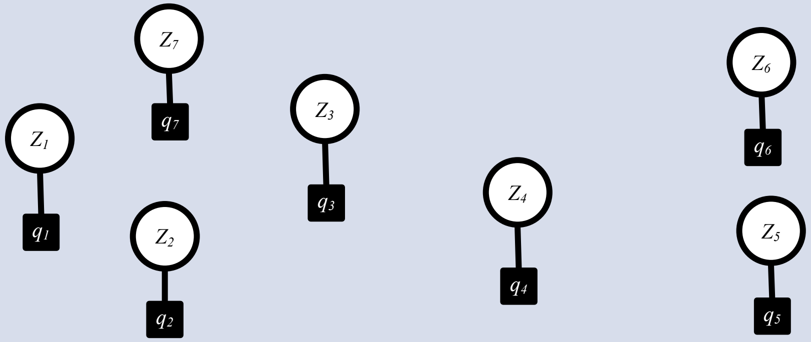
Mean Field Approximation

The **mean field approximation** assumes our variational approximation $q_\theta(\mathbf{z})$ treats each variable as independent

$$p_\alpha(\mathbf{z} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}} \psi_c(\mathbf{z}_c, \mathbf{x})$$

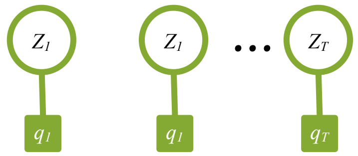


$$q_\theta(\mathbf{z}) = \prod_{t=1}^{T} q_t(z_t)$$



Mean Field V.I. Overview

1. *Goal*: estimate $p_\alpha(\mathbf{z} \mid \mathbf{x})$
   we assume this is intractable to compute exactly
2. *Idea*: approximate with another distribution $q_\theta(\mathbf{z}) \approx p_\alpha(\mathbf{z} \mid \mathbf{x})$ for each $\mathbf{x}$
3. *Mean Field*: assume $q_\theta(\mathbf{z}) = \prod_t q_t(z_t; \theta)$
   i.e., we decompose over variables
   other choices for the decomposition of $q_\theta(\mathbf{z})$ give rise to "structured mean field"



4. *Optimization Problem*: pick the q that minimizes KL(q || p)

$$\hat{q}(\mathbf{z}) = \underset{q(\mathbf{z}) \in \mathcal{Q}}{\operatorname{argmin}} \, \mathsf{KL}(q(\mathbf{z}) || p(\mathbf{z} \mid \mathbf{x}))$$

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \, \mathsf{KL}(q_\theta(\mathbf{z}) || p_\alpha(\mathbf{z} \mid \mathbf{x}))$$

equivalent

5. *Optimization Algorithm*: coordinate descent
   i.e. pick the best $q_t(z_t)$ based on the other $\{ q_s(z_s) \}_{s \neq t}$ being fixed

Optimizing KL Divergence

- *Question*: How do we minimize KL?

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \, \mathsf{KL}(q_\theta(\mathbf{z}) \parallel p_\alpha(\mathbf{z} \mid \mathbf{x}))$$

- *Answer #1*: Oh no! We can't even compute this KL.

Why we can't compute KL…

$$\begin{aligned}
\mathsf{KL}(q(\mathbf{z}) \parallel p(\mathbf{z} \mid \mathbf{x})) &= E_{q(\mathbf{z})}\left[\log\left(\frac{q(\mathbf{z})}{p(\mathbf{z} \mid \mathbf{x})}\right)\right] \\
&= E_{q(\mathbf{z})}[\log q(\mathbf{z})] - E_{q(\mathbf{z})}[\log p(\mathbf{z} \mid \mathbf{x})] \\
&= E_{q(\mathbf{z})}[\log q(\mathbf{z})] - E_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z})] + E_{q(\mathbf{z})}[\log p(\mathbf{x})] \\
&= E_{q(\mathbf{z})}[\log q(\mathbf{z})] - E_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z})] + \log p(\mathbf{x})
\end{aligned}$$

we have the same problem with an intractable data likelihood p(x) or an intractable partition function Z(x)

we assumed this is intractable to compute!

Optimizing KL Divergence

- *Question*: How do we minimize KL?

$$\hat{\theta} = \underset{\theta \in \Theta}{\text{argmin}}\, \text{KL}(q_\theta(\mathbf{z}) \parallel p_\alpha(\mathbf{z} \mid \mathbf{x}))$$

- *Answer #1*: Oh no! We can't even compute this KL.

Why we can't compute KL…

$$\begin{aligned} \text{KL}(q(\mathbf{z}) \parallel p(\mathbf{z} \mid \mathbf{x})) &= E_{q(\mathbf{z})}\left[\log\left(\frac{q(\mathbf{z})}{p(\mathbf{z} \mid \mathbf{x})}\right)\right] \\ &= E_{q(\mathbf{z})}[\log q(\mathbf{z})] - E_{q(\mathbf{z})}[\log p(\mathbf{z} \mid \mathbf{x})] \\ &= E_{q(\mathbf{z})}[\log q(\mathbf{z})] - E_{q(\mathbf{z})}[\log \tilde{p}(\mathbf{z} \mid \mathbf{x})] + E_{q(\mathbf{z})}[\log Z(\mathbf{x})] \\ &= E_{q(\mathbf{z})}[\log q(\mathbf{z})] - E_{q(\mathbf{z})}[\log \tilde{p}(\mathbf{z} \mid \mathbf{x})] + \log Z(\mathbf{x}) \end{aligned}$$

we have the same problem with an intractable data likelihood p(x) or an intractable partition function Z(x)

we assumed this is intractable to compute!

Optimizing KL Divergence

- *Question*: How do we minimize KL?

$$\hat{\theta} = \operatorname*{argmin}_{\theta \in \Theta} \text{KL}(q_\theta(\mathbf{z}) \parallel p_\alpha(\mathbf{z} \mid \mathbf{x}))$$

- *Answer #2*: We don't need to compute this KL
  We can instead maximize the ELBO (i.e. **E**vidence **L**ower **BO**und)

$$\text{ELBO}(q_\theta) = E_{q_\theta(\mathbf{z})}[\log p_\alpha(\mathbf{x}, \mathbf{z})] - E_{q_\theta(\mathbf{z})}[\log q_\theta(\mathbf{z})]$$

The ELBO for a DGM

Here is why…

$$\begin{aligned}
\theta &= \operatorname*{argmin}_{\theta} \text{KL}(q_\theta(\mathbf{z}) \parallel p_\alpha(\mathbf{z} \mid \mathbf{x})) \\
&= \operatorname*{argmin}_{\theta} E_{q_\theta(\mathbf{z})}[\log q_\theta(\mathbf{z})] - E_{q_\theta(\mathbf{z})}[\log p_\alpha(\mathbf{x}, \mathbf{z})] + \log p_\alpha(\mathbf{x}) \\
&= \operatorname*{argmin}_{\theta} E_{q_\theta(\mathbf{z})}[\log q_\theta(\mathbf{z})] - E_{q_\theta(\mathbf{z})}[\log p_\alpha(\mathbf{x}, \mathbf{z})] \\
&= \operatorname*{argmax}_{\theta} \text{ELBO}(q_\theta)
\end{aligned}$$

dropping the intractable term gives the ELBO

Optimizing KL Divergence

- *Question*: How do we minimize KL?

$$\hat{\theta} = \operatorname*{argmin}_{\theta \in \Theta} \underbrace{\text{KL}(q_\theta(\mathbf{z}) \parallel p_\alpha(\mathbf{z} \mid \mathbf{x}))}$$

- *Answer #2*: We don't need to compute this KL
  We can instead maximize the ELBO (i.e. **E**vidence **L**ower **BO**und)

$$\text{ELBO}(q_\theta) = E_{q_\theta(\mathbf{z})}[\log \tilde{p}_\alpha(\mathbf{z} \mid \mathbf{x})] - E_{q_\theta(\mathbf{z})}[\log q_\theta(\mathbf{z})]$$

The ELBO for a UGM

Here is why…

$$\begin{aligned}
\theta &= \operatorname*{argmin}_{\theta} \text{KL}(q_\theta(\mathbf{z}) \parallel p_\alpha(\mathbf{z} \mid \mathbf{x})) \\
&= \operatorname*{argmin}_{\theta} E_{q_\theta(\mathbf{z})}[\log q_\theta(\mathbf{z})] - E_{q_\theta(\mathbf{z})}[\log \tilde{p}_\alpha(\mathbf{z} \mid \mathbf{x})] + \underbrace{\log Z_\alpha(\mathbf{x})} \\
&= \operatorname*{argmin}_{\theta} E_{q_\theta(\mathbf{z})}[\log q_\theta(\mathbf{z})] - E_{q_\theta(\mathbf{z})}[\log \tilde{p}_\alpha(\mathbf{z} \mid \mathbf{x})] \\
&= \operatorname*{argmax}_{\theta} \text{ELBO}(q_\theta)
\end{aligned}$$

dropping the intractable term gives the ELBO

ELBO as Objective Function

What does maximizing ELBO(q_θ) accomplish?

$$\text{ELBO}(q_\theta) = E_{q_\theta(\mathbf{z})}\left[\log p_\alpha(\mathbf{x}, \mathbf{z})\right] - E_{q_\theta(\mathbf{z})}\left[\log q_\theta(\mathbf{z})\right]$$

1. The first expectation is high if q_θ puts probability mass on the same values of **z** that p_α puts probability mass

2. The second term is the entropy of q_θ and the entropy will be high if q_θ spreads its probability mass evenly

ELBO as lower bound

- *For a DGM*:
  - ELBO(q) is a lower bound for log p(x)
- *For a UGM*:
  - ELBO(q) is a lower bound for log Z(x)

Takeaway: in variational inference, we find the q that gives the **tightest bound** on the normalization constant for p(z | x)

ELBO's relation to log p(x)

Theorem:

for any q, $\log p(x) \geq \text{ELBO}(q)$

i.e. $\text{ELBO}(q)$ is a lower bound on $\log p(x)$

Proof #1:

Recall Jensen's Inequality: $f(E[x]) \geq E[f(x)]$, for concave f

$$\log p(x) = \log \int_z p(x,z)\, dz \quad \text{(marginal)}$$

$$= \log \int_z p(x,z) \frac{q(z)}{q(z)}\, dz \quad \text{(mult. by 1)}$$

$$= \log E_{q(z)}\left[p(x,z) / q(z) \right] \quad \text{(def. of expectation)}$$

$$\geq E_{q(z)}\left[\log \left(\frac{p(x,z)}{q(z)} \right) \right] \quad \text{(by Jensen's Ineq.)}$$

$$= E_{q(z)}[\log p(x,z)] - E_{q(z)}[\log q(z)] = \text{ELBO}(q)$$

$$\Rightarrow \log p(x) \geq \text{ELBO}(q)$$

Proof #2:

① $\log p(x) = KL(q \| p) + \text{ELBO}(q)$

② $KL(q \| p) \geq 0$ (without proof)

③ $\Rightarrow \log p(x) \geq \text{ELBO}(q)$

Key Takeaway:

minimizing KL is the same as finding a tight ELBO(q) lower bound for log p(x)

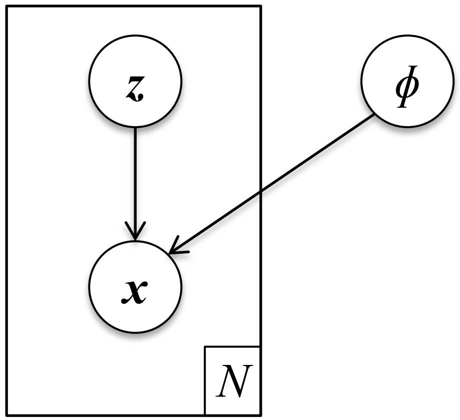
VARIATIONAL AUTOENCODERS

Why VAEs?

- **Autoencoders:**
  - learn a low dimensional representation of the input, but hard to work with as a generative model
  - one of the key limitations of autoencoders is that we have no way of **sampling** from them!
- **Variational autoencoders (VAEs)**
  - by contrast learn a continuous latent space that is **easy to sample from!**
  - can **generate** new data (e.g. images) by sampling from the learned generative model

Variational Autoencoders

$p_\phi(\mathbf{x}, \mathbf{z})$



$z \sim \text{Gaussian}(0, I)$

Graphical Model Perspective

- The DGM diagram shows that the VAE model is quite simple as a graphical model (ignoring the neural net details that give rise to **x**)
- Sampling from the model is easy:
  - Consider a DGM where $\mathbf{x} = g_\phi(\mathbf{z}/10 + \mathbf{z}/||\mathbf{z}||)$ (i.e. we don't use parameters ϕ)
  - Then we can draw samples of **z** and directly convert them to values **x**
- **Key idea of VAE:** define $g_\phi(z)$ as a neural net and learn ϕ from data

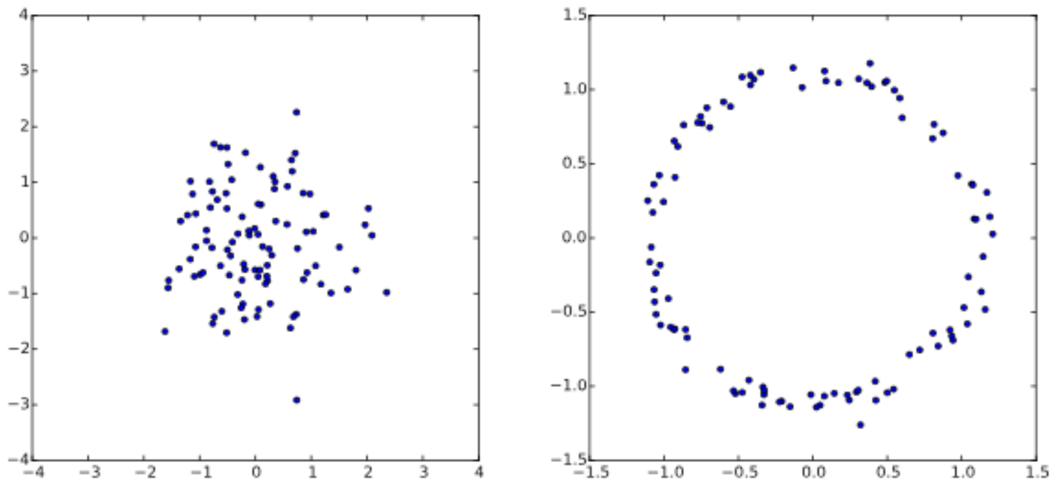


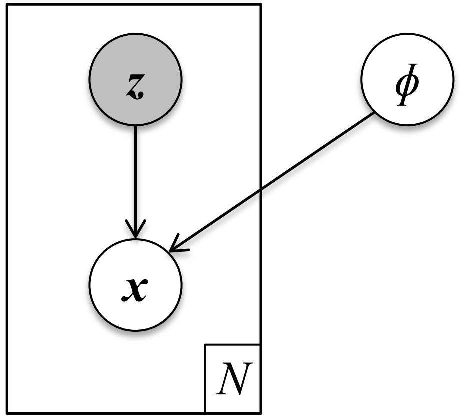
Figure from Doersch (2016)

Variational Autoencoders

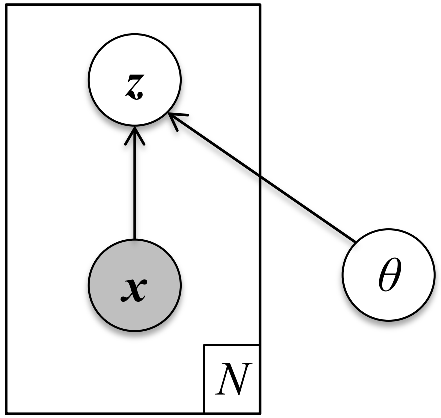
Neural Network Perspective

- We can view a variational autoencoder (VAE) as an autoencoder consisting of two neural networks
- VAEs (as encoders) define two distributions:
  - **encoder**: $q_\theta(\mathbf{z} \mid \mathbf{x})$
  - **decoder**: $p_\phi(\mathbf{x} \mid \mathbf{z})$
- Parameters θ and ϕ are neural network parameters (i.e. θ are not the variational parameters)

$p_\phi(\mathbf{x} \mid \mathbf{z})$



$q_\theta(\mathbf{z} \mid \mathbf{x})$

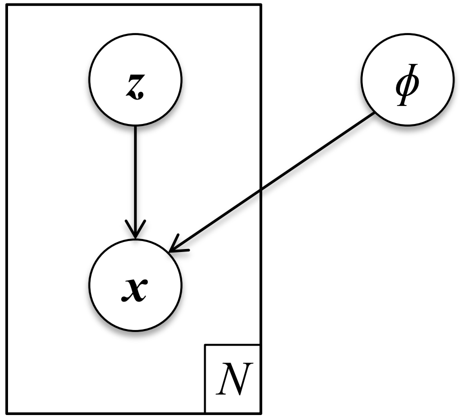


Variational Autoencoders

Graphical Model Perspective

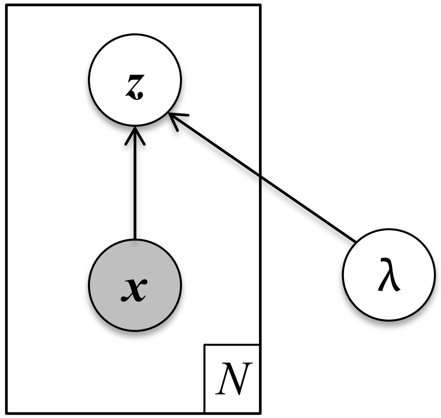
- We can also view the VAE from the perspective of variational inference
- In this case we have two distributions:
  - **model**: $p_\phi(\mathbf{z} \mid \mathbf{x})$
  - **variational approximation**: $q_{\lambda=f(x;\,\theta)}(\mathbf{z} \mid \mathbf{x})$
- We have the same model parameters ϕ
- The variational parameters λ are a function of NN parameters θ

$p_\phi(\mathbf{x}, \mathbf{z})$



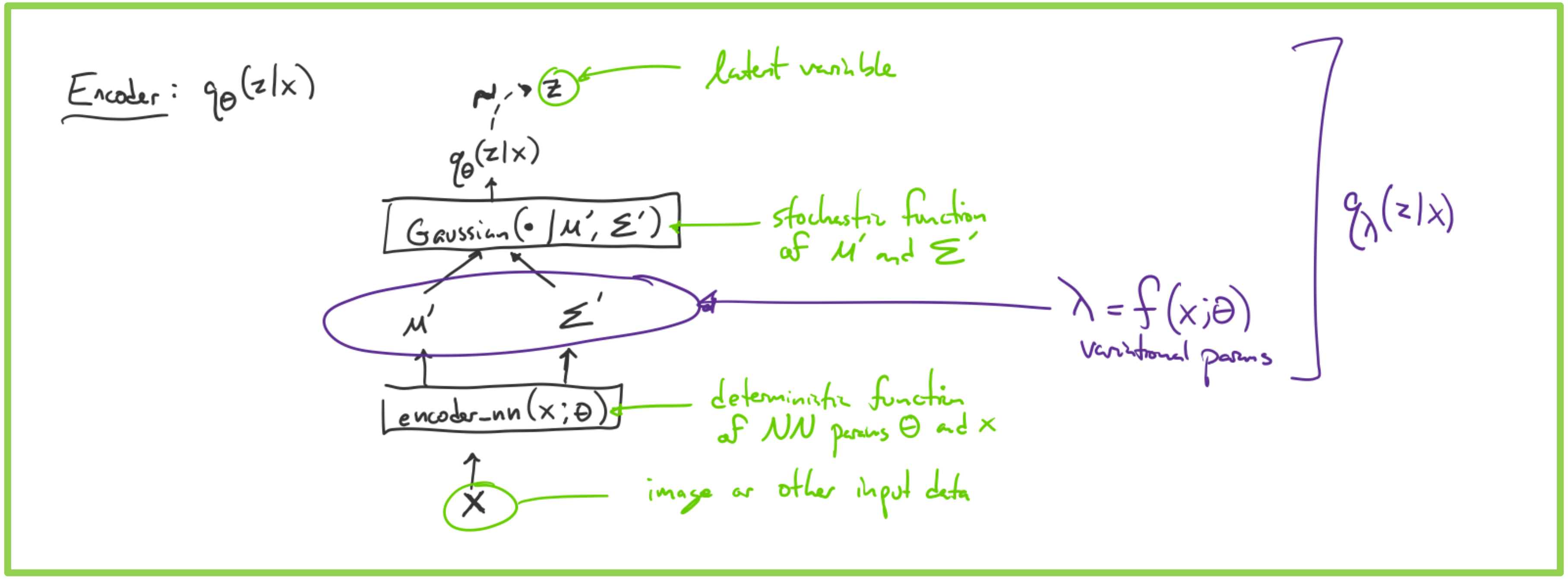
$z \sim \text{Gaussian}(0, I)$

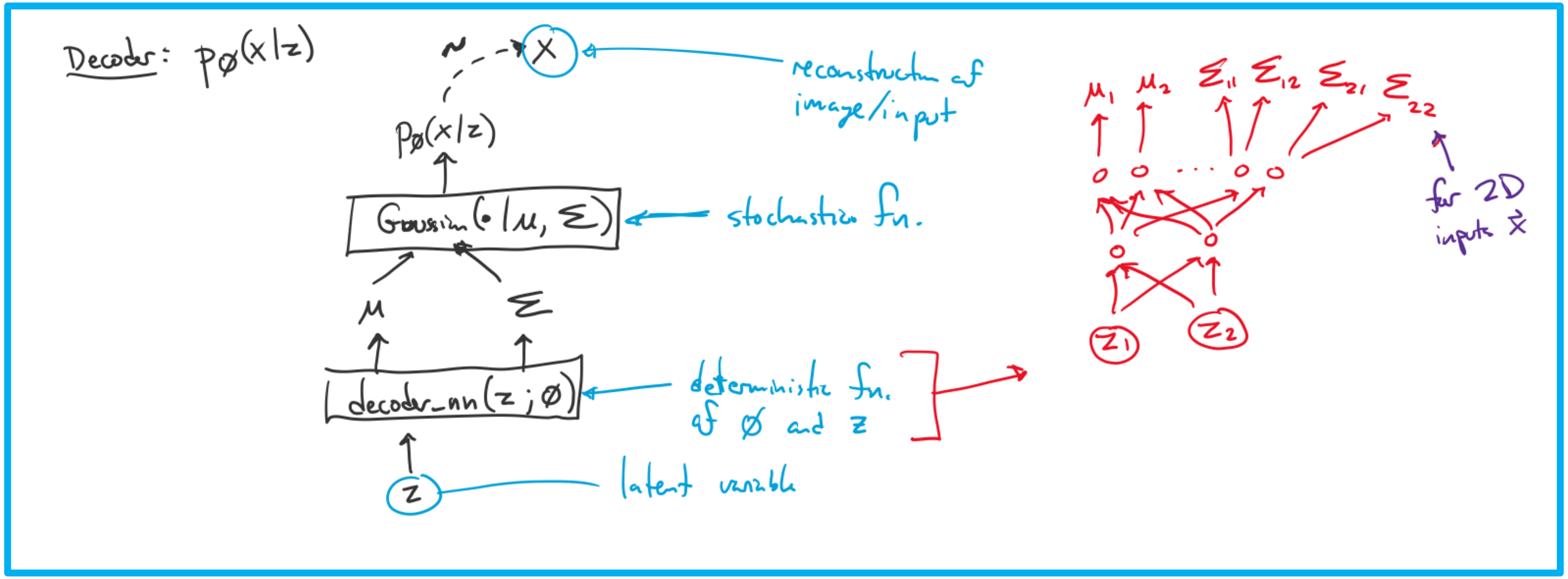
$q_\lambda(\mathbf{z} \mid \mathbf{x})$



$\lambda = f(x;\, \theta)$

VAEs: Neural Network View





VAEs: Neural Network View

Training VAE

Dataset : $D = \{\vec{x}^{(i)}\}_{i=1}^{N}$ unlabeled data (eg. images)

Loss Fn :

$$\ell(\theta,\phi) = \sum_{i=1}^{N} \ell_i(\theta,\phi)$$

$$\ell_i(\theta,\phi) = -\mathbb{E}_{q_\theta(z|x^{(i)})}\left[\log p_\phi(x^{(i)}|z)\right] + KL\left(q_\theta(z|x^{(i)}) \,\|\, p(z)\right)$$

reconstruction loss (just like an autoencoder)

Q1: what does this term accomplish?
A: regularizer

$$= -\sum_{s=1}^{S} q_\theta(z^{(s)}|x^{(i)}) \log p_\phi(x^{(i)}|z^{(s)}) + KL\left(q_\theta(z|x^{(i)}) \,\|\, p(z)\right)$$

where $z^{(s)} \sim q_\theta(\cdot|x^{(i)}) \ \forall s$

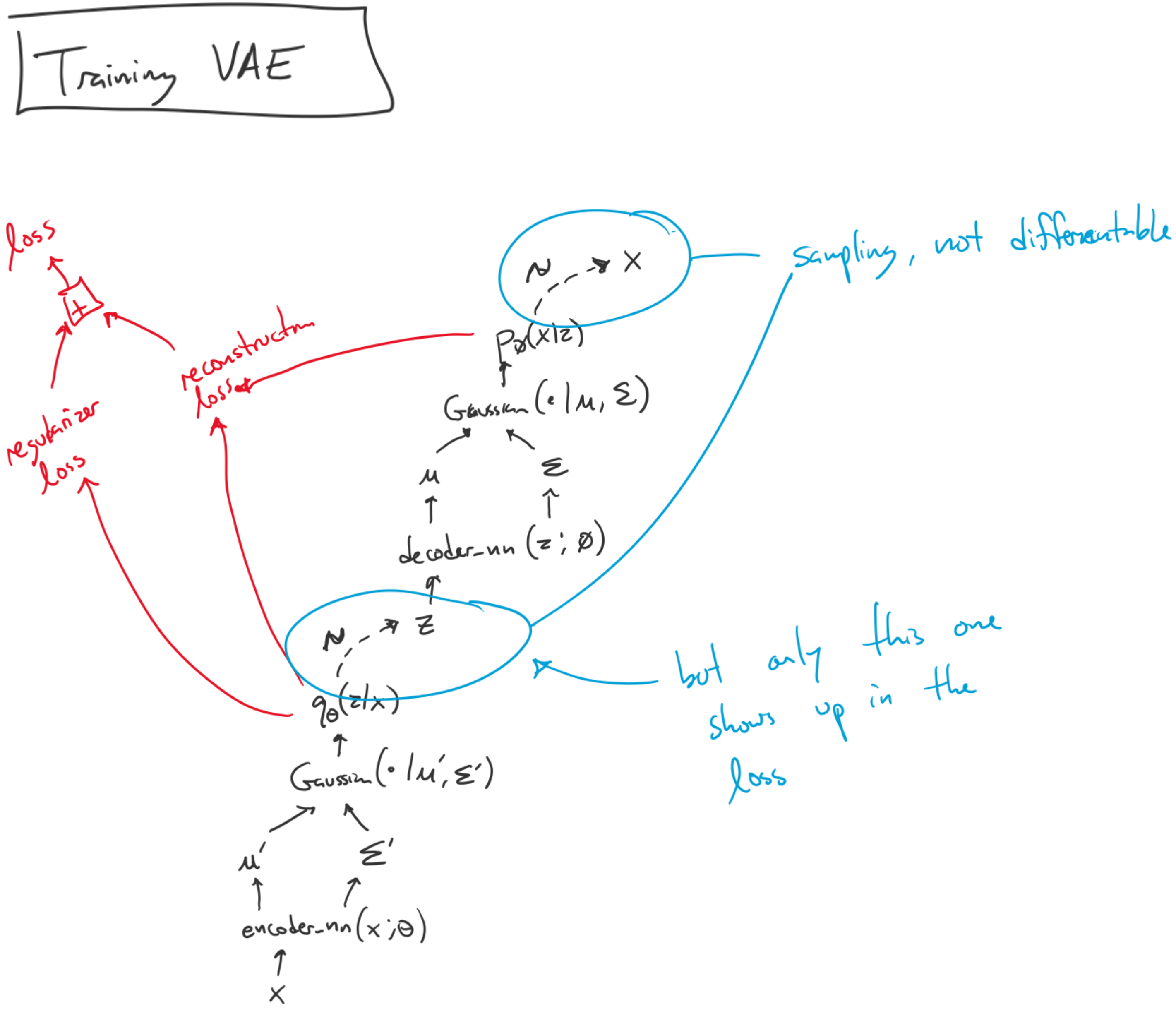
Monte Carlo Approx.

NN Perspective :

just backpropagation with a loss consisting of two terms

a) reconstruction loss
b) regularizer

VAEs: Neural Network View



Reparameterization Trick

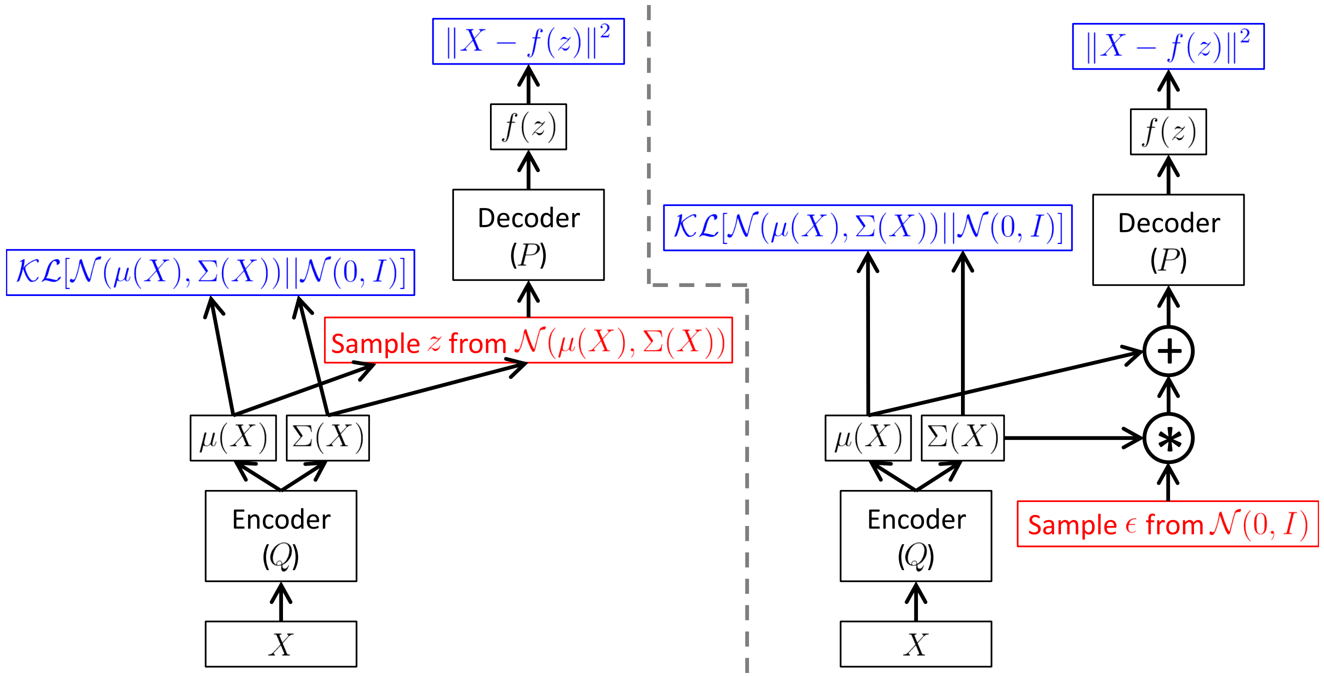


Figure 4: A training-time variational autoencoder implemented as a feed-forward neural network, where $P(X|z)$ is Gaussian. Left is without the "reparameterization trick", and right is with it. Red shows sampling operations that are non-differentiable. Blue shows loss layers. The feedforward behavior of these networks is identical, but backpropagation can be applied only to the right network.

Figure from Doersch (2016)

VAE RESULTS

VAEs for Image Generation

Kingma & Welling (2014)

- introduced VAEs
- applied to image generation

Model

- $p_\phi(\mathbf{z}) \sim N(\mathbf{z}; 0, \mathbf{I})$
- $p_\phi(\mathbf{x} \mid \mathbf{z})$ is a multivariate Gaussian with mean and variance computed by an MLP, fully connected neural network with a single hidden layer with parameters ϕ
- $q_\theta(\mathbf{z} \mid \mathbf{x})$ is a multivariate Gaussian with diagonal covariance structure and with mean and variance computed by an MLP with parameters θ

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

Figure from Kingma & Welling (2014)

VAEs for Image Generation

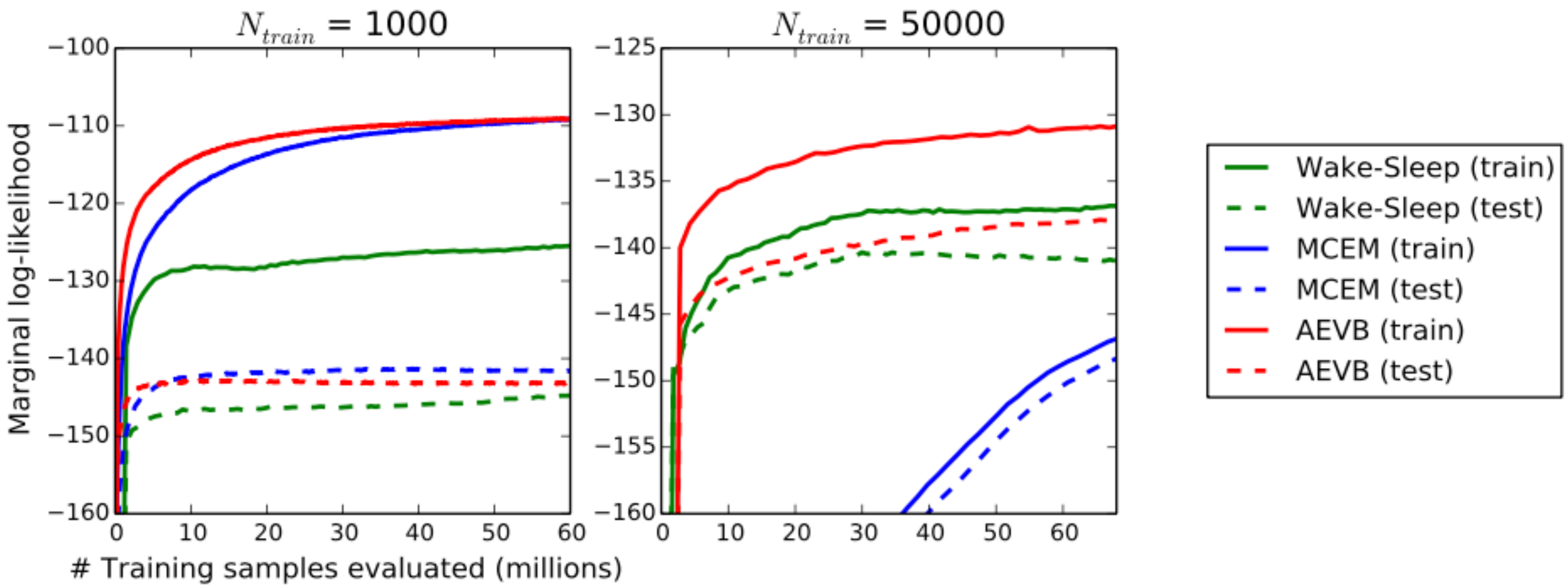
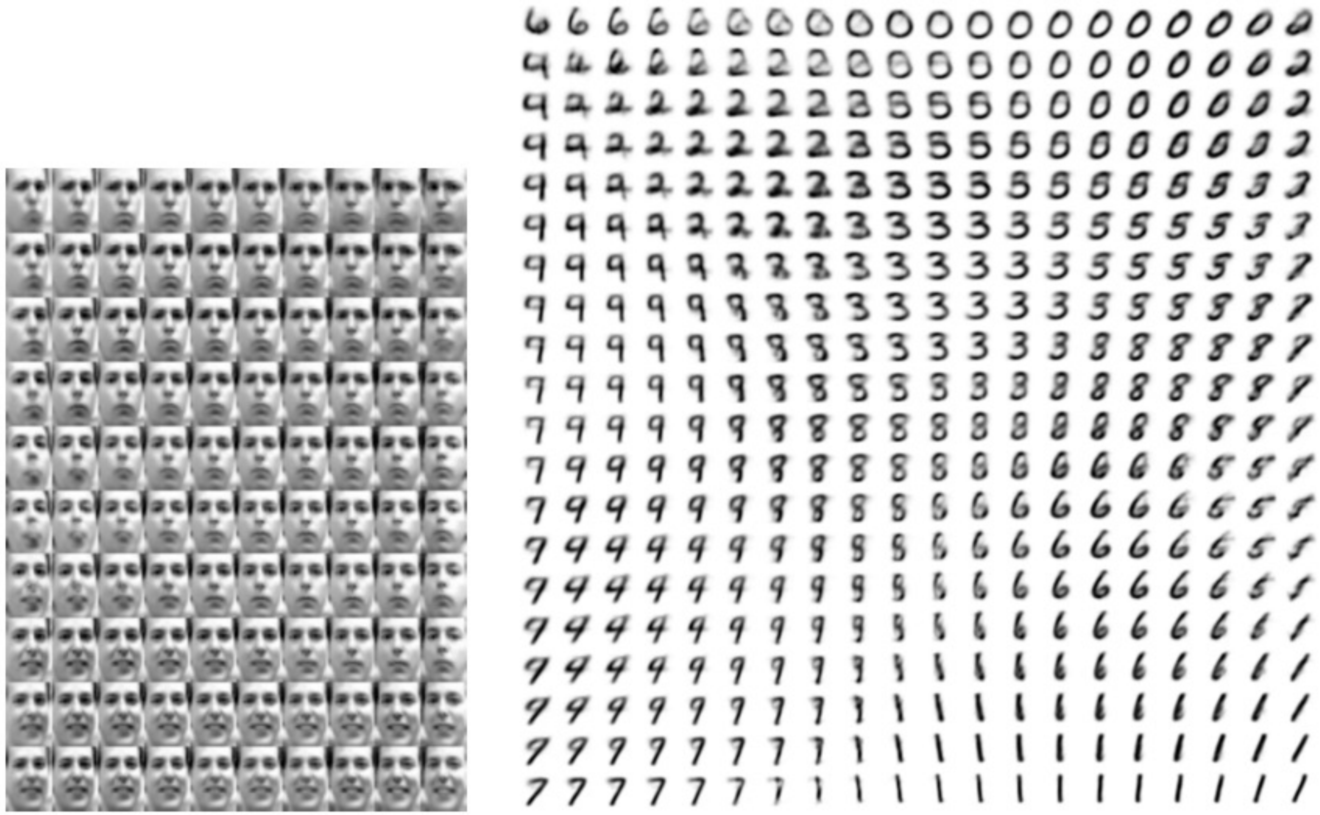


Figure 3: Comparison of AEVB to the wake-sleep algorithm and Monte Carlo EM, in terms of the estimated marginal likelihood, for a different number of training points. Monte Carlo EM is not an on-line algorithm, and (unlike AEVB and the wake-sleep method) can't be applied efficiently for the full MNIST dataset.

Figure from Kingma & Welling (2014)

VAEs for Image Generation

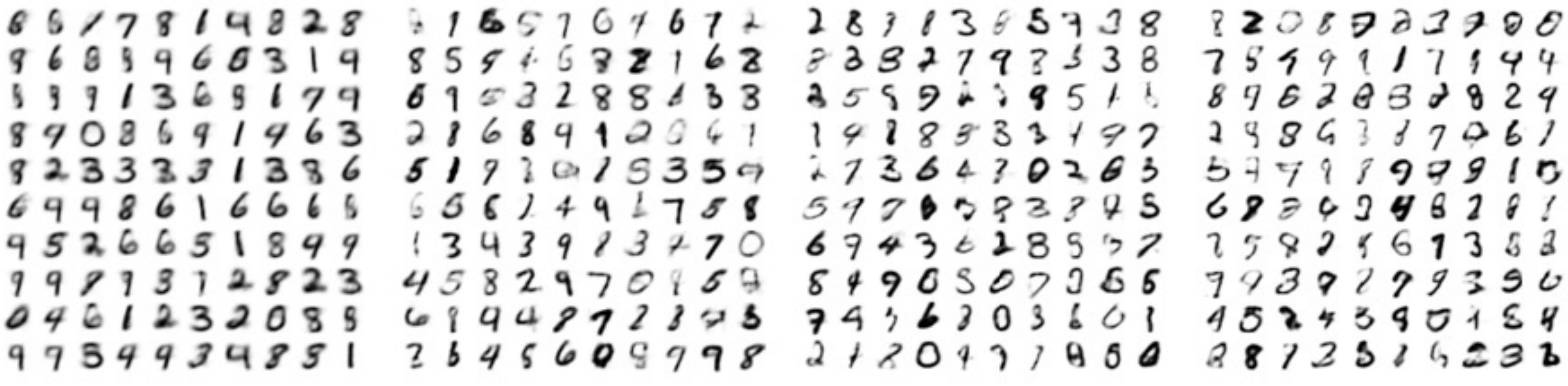


(a) Learned Frey Face manifold (b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

Figure from Kingma & Welling (2014)

VAEs for Image Generation



(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

Figure from Kingma & Welling (2014)

VAEs for Text Generation

Bowman et al. (2015)

- example of an application of VAEs to discrete data
- built on the sequence-to-sequence framework:
  - input is read in by an LSTM
  - output is generated by an LSTM-LM

Model

- $p_\phi(\mathbf{z}) \sim N(\mathbf{z}; 0, \mathbf{I})$
- $p_\phi(\mathbf{x} \mid \mathbf{z})$ is an LSTM Language Model with parameters $\boldsymbol{\phi}$
- $q_\theta(\mathbf{z} \mid \mathbf{x})$ is a multivariate Gaussian with mean and variance computed by an LSTM with parameters θ

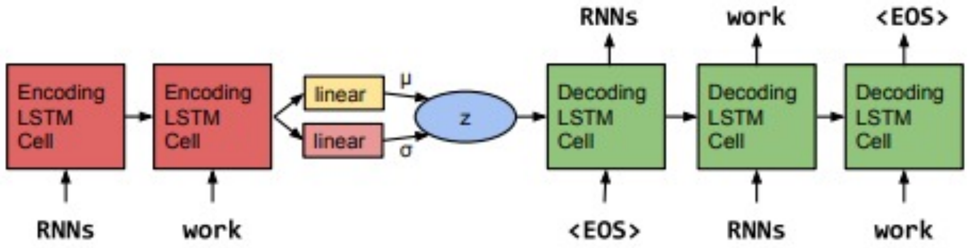


Figure 1: The core structure of our variational autoencoder language model. Words are represented using a learned dictionary of embedding vectors.

Figure from Bowman et al. (2015)

VAEs for Text Generation

INPUT	**we looked out at the setting sun .**	**i went to the kitchen .**	**how are you doing ?**
MEAN	*they were laughing at the same time .*	*i went to the kitchen .*	*what are you doing ?*
SAMP. 1	*ill see you in the early morning .*	*i went to my apartment .*	*" are you sure ?*
SAMP. 2	*i looked up at the blue sky .*	*i looked around the room .*	*what are you doing ?*
SAMP. 3	*it was down on the dance floor .*	*i turned back to the table .*	*what are you doing ?*

Table 7: Three sentences which were used as inputs to the VAE, presented with greedy decodes from the mean of the posterior distribution, and from three samples from that distribution.

" i want to talk to you . "
"i want to be with you . "
"i do n't want to be with you . "
i do n't want to be with you .
she did n't want to be with him .
he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .

Table 8: Paths between pairs of random points in VAE space: Note that intermediate sentences are grammatical, and that topic and syntactic structure are usually locally consistent.

Figure from Bowman et al. (2015)

VQ-VAE

- Vector Quantized VAE (VQ-VAE) learns a continuous codebook, but the encoder outputs discrete codes
- Decoder takes a code and generates a sample conditioned on it

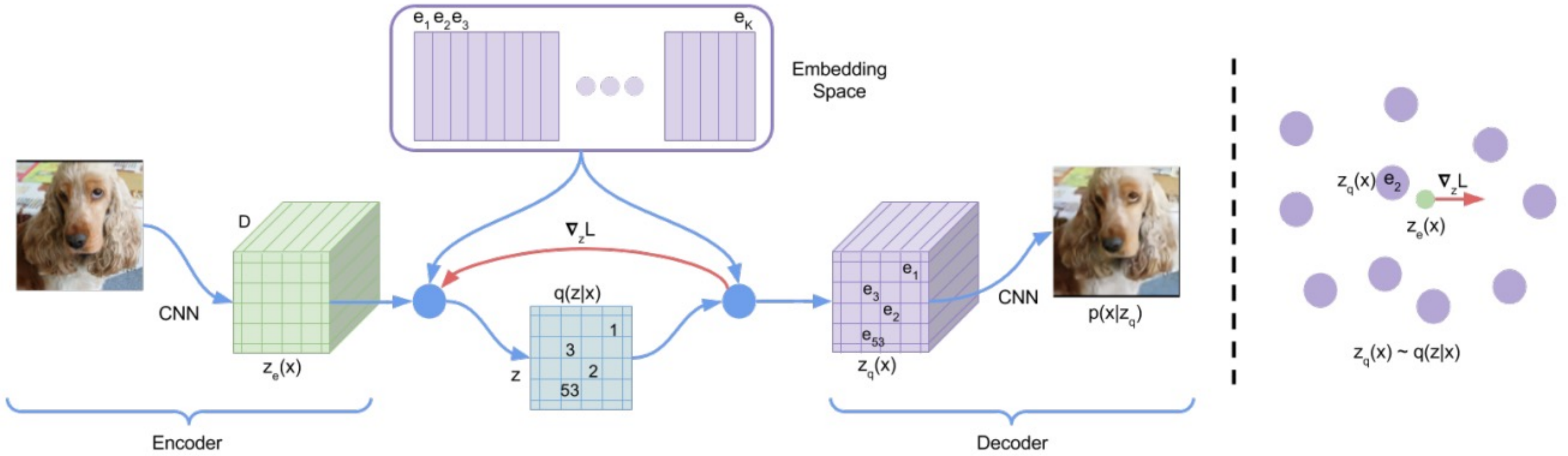


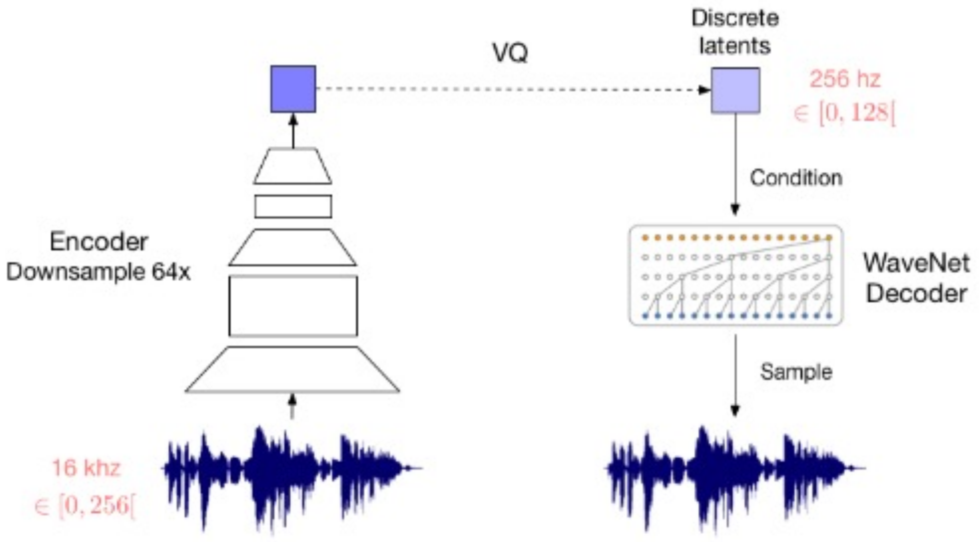
Figure 1: Left: A figure describing the VQ-VAE. Right: Visualisation of the embedding space. The output of the encoder $z(x)$ is mapped to the nearest point e_2. The gradient $\nabla_z L$ (in red) will push the encoder to change its output, which could alter the configuration in the next forward pass.

Figure from van den Oord et al. (2018)

VQ-VAE

- Vector Quantized VAE (VQ-VAE) learns a continuous codebook, but the encoder outputs discrete codes
- Decoder takes a code and generates a sample conditioned on it

Example: Generating Audio

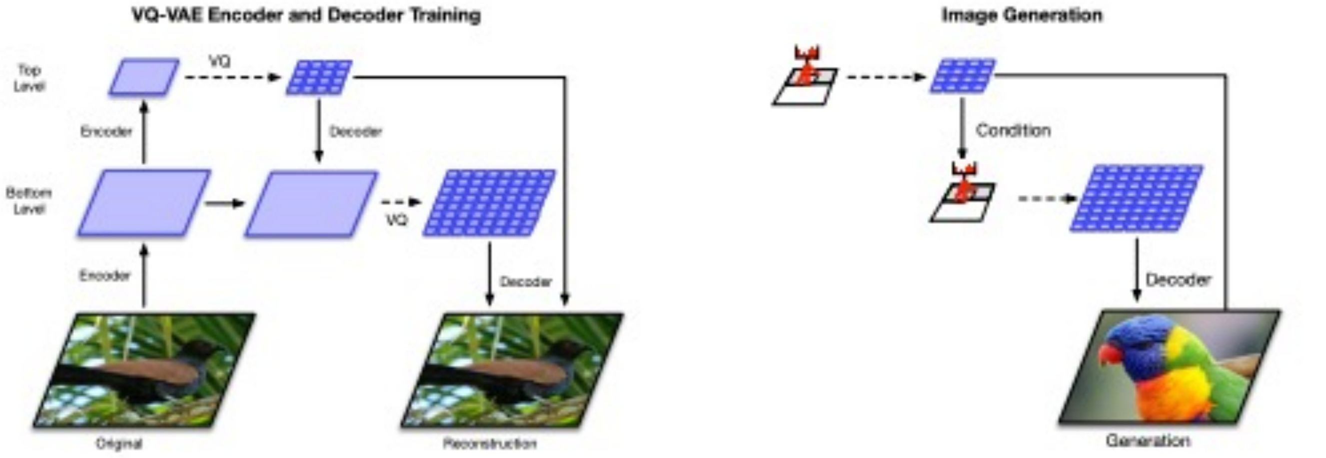


https://avdnoord.github.io/homepage/vqvae

Figure from van den Oord et al. (2018)

VQ-VAE

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity



(a) Overview of the architecture of our hierarchical VQ-VAE. The encoders and decoders consist of deep neural networks. The input to the model is a 256×256 image that is compressed to quantized latent maps of size 64×64 and 32×32 for the *bottom* and *top* levels, respectively. The decoder reconstructs the image from the two latent maps.

(b) Multi-stage image generation. The top-level PixelCNN prior is conditioned on the class label, the bottom level PixelCNN is conditioned on the class label as well as the first level code. Thanks to the feed-forward decoder, the mapping between latents to pixels is fast. (The example image with a parrot is generated with this model).

Figure from Razavi et al. (2019)

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity

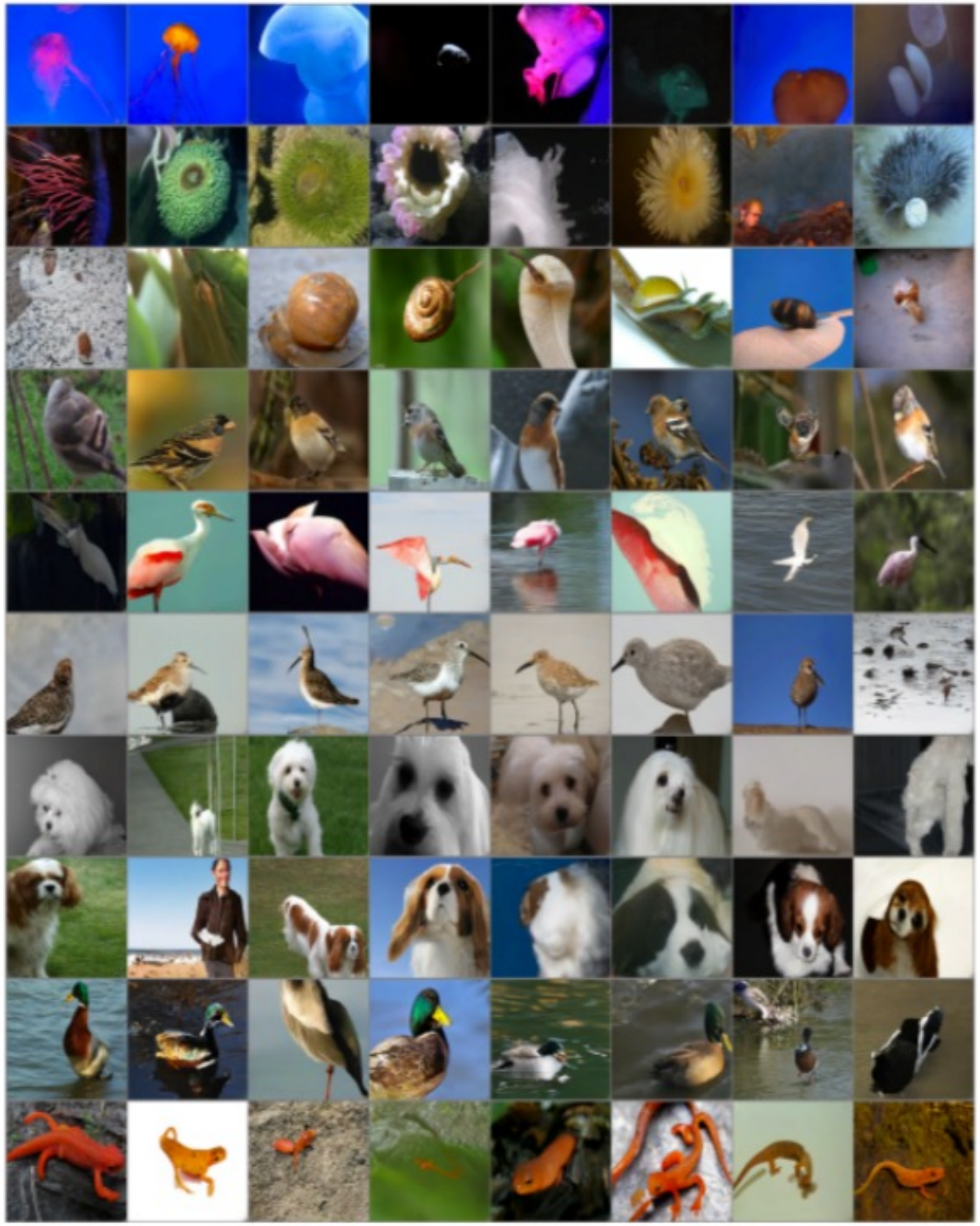


Figure 4: Class conditional random samples. Classes from the top row are: 108 sea anemone, 109 brain coral, 114 slug, 11 goldfinch, 130 flamingo, 141 redshank, 154 Pekinese, 157 papillon, 97 drake, and 28 spotted salamander.

Figure from Razavi et al. (2019)

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity

Figure from Razavi et al. (2019)

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity

Figure from Razavi et al. (2019)

- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity

Figure from Razavi et al. (2019)