Machine Learning Department
School of Computer Science
Carnegie Mellon University

Variational Inference
+
Variational Autoencoders (VAEs)

Matt Gormley & Henry Chai
Lecture 7
Sep. 18, 2024

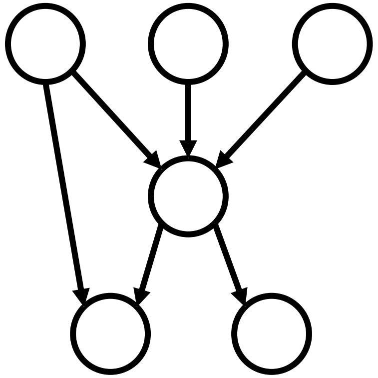
Reminders

- Homework 1: Generative Models of Text
  - Out: Mon, Sep 9
  - Due: Mon, Sep 23 at 11:59pm

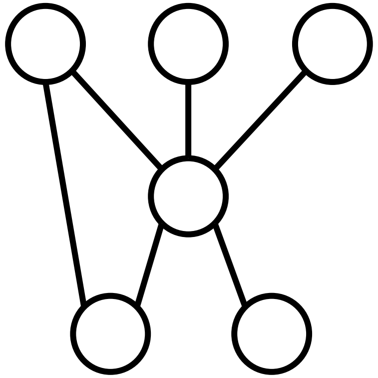
DIRECTED GRAPHICAL MODEL

Three Types of Graphical Models

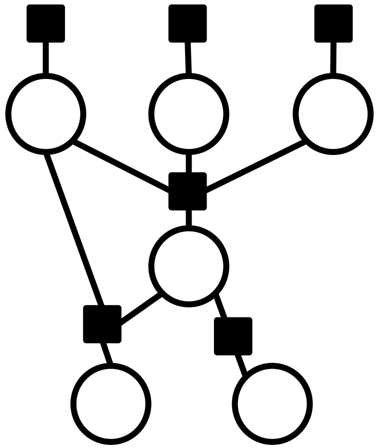
Directed Graphical Model



Undirected Graphical Model



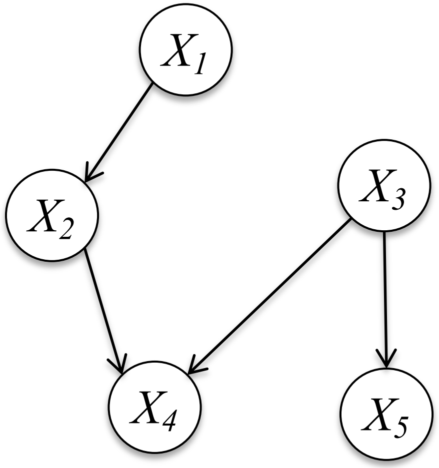
Factor Graph



Directed Graphical Model

$P(X_1, X_2, X_3) = P(X_1)P(X_2 | X_1)P(X_3 | X_2, X_1)$

Example



$$P(X_1, X_2, X_3, X_4, X_5) = \\ P(X_5 \mid X_3)P(X_4 \mid X_2, X_3) \\ P(X_3)P(X_2 \mid X_1)\, P(X_1)$$

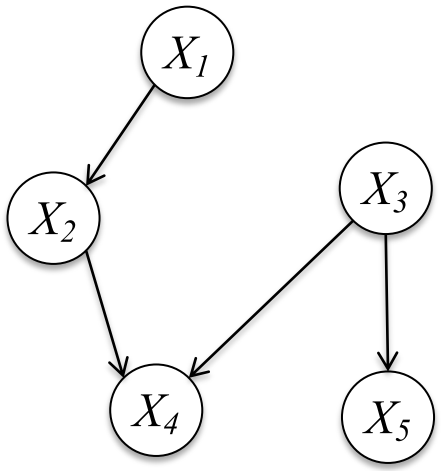
Definition

- A directed graphical model (aka. Bayesian network) is a **directed acyclic graph** that represents the conditional independencies of a set of variables $X_1, \ldots, X_T$
- Each **node** is variable X_t and each **edge** implies a directional influence between a pair of variables
- The DGM factorizes the joint distribution over the variables as a product of conditional probabilities:

$$P(X_1, \ldots, X_T) = \prod_{t=1}^{T} P(X_t \mid \text{parents}(X_t))$$

Directed Graphical Model

Example



the graph (*qualitative specification*) could be:

- specified using domain expertise about causal relationships
- learned from data
- chosen because of nice computational properties

$$P(X_1, X_2, X_3, X_4, X_5) =$$
$$P(X_5 \mid X_3) P(X_4 \mid X_2, X_3)$$
$$P(X_3) P(X_2 \mid X_1) P(X_1)$$

the conditional probabilities (*quantitative specification*) is:

- depends on the types of variables involved
- typically learned from data

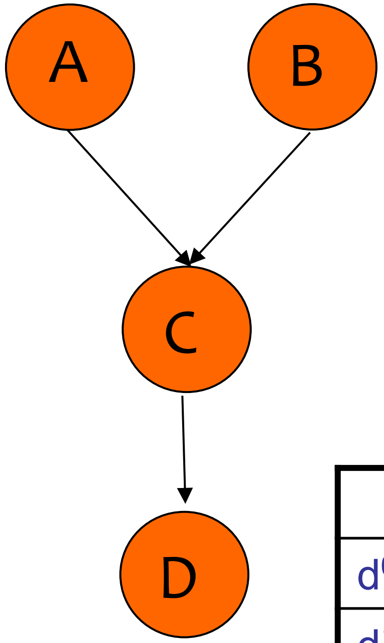
Quantitative Specification

Example: Conditional probability tables (CPTs) for discrete random variables

a^0	0.75
a^1	0.25

b^0	0.33
b^1	0.67

$$P(a,b,c.d) = P(a)P(b)P(c|a,b)P(d|c)$$



	a^0b^0	a^0b^1	a^1b^0	a^1b^1
c^0	0.45	1	0.9	0.7
c^1	0.55	0	0.1	0.3

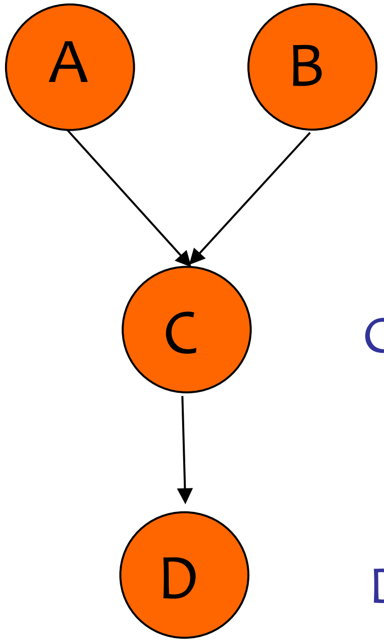
	c^0	c^1
d^0	0.3	0.5
d^1	07	0.5

Quantitative Specification

Example: Conditional probability density functions (CPDs) for continuous random variables

$A \sim N(\mu_a, \Sigma_a)$ $B \sim N(\mu_b, \Sigma_b)$

$$P(a,b,c.d) = P(a)P(b)P(c|a,b)P(d|c)$$



$C \sim N(A+B, \Sigma_c)$

$D \sim N(\mu_d + C, \Sigma_d)$

$P(D|C)$

D

C

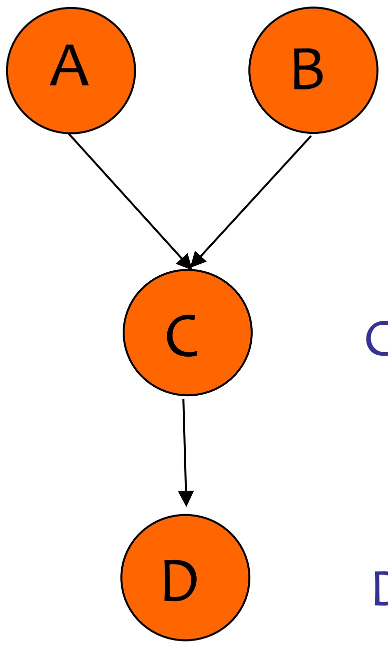
Quantitative Specification

Example: Combination of CPTs and CPDs
for a mix of discrete and continuous variables

a^0	0.75
a^1	0.25

b^0	0.33
b^1	0.67

$P(a,b,c.d) =$
$P(a)P(b)P(c|a,b)P(d|c)$



$C \sim N(A+B, \Sigma_c)$

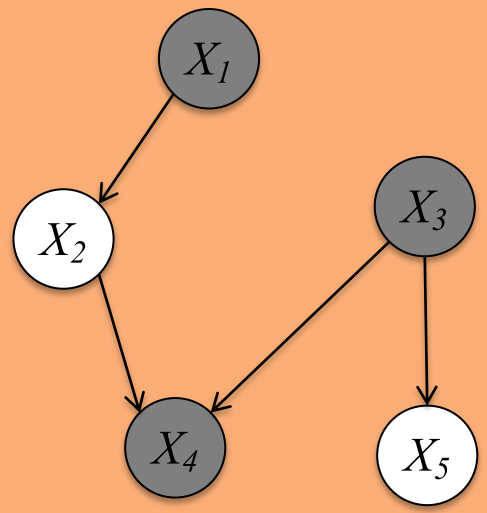
$D \sim N(\mu_d+C, \Sigma_d)$

Observed Variables

- In a graphical model, **shaded nodes** are "**observed**", i.e. their values are given

Example:

$$P(X_2, X_5 \mid X_1 = 0, X_3 = 1, X_4 = 1)$$



MARKOV MODEL

Markov Model

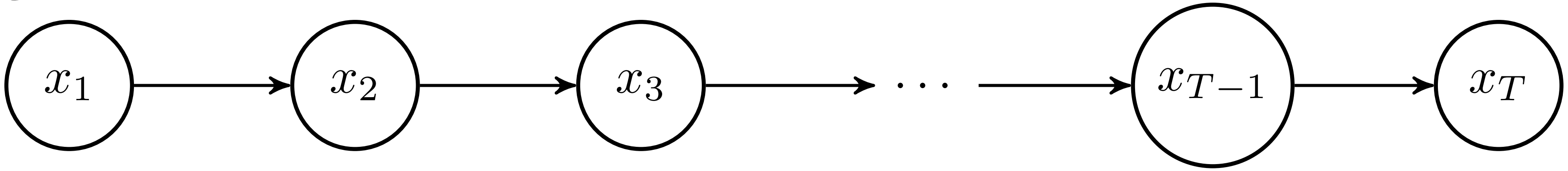
1. **1st-order Markov assumption:** for a sequence of random variables, the probability distribution over x_t random variables is conditionally independent of $x_1, \ldots, x_{t-2}$ given x_{t-1}

$$p(x_t \mid x_1, \ldots, x_{t-1}) = p(x_t \mid x_{t-1})$$

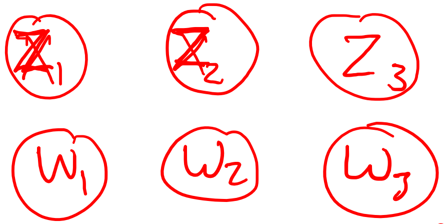
2. **1st-order Markov model:** defines a joint distribution over a sequence of variables using a Markov assumption

$$p(x_1, \ldots, x_T) = p(x_1) \prod_{t=2}^{T} p(x_t \mid x_{t-1})$$

3. We can represent the Markov model as a **directed graphical model**



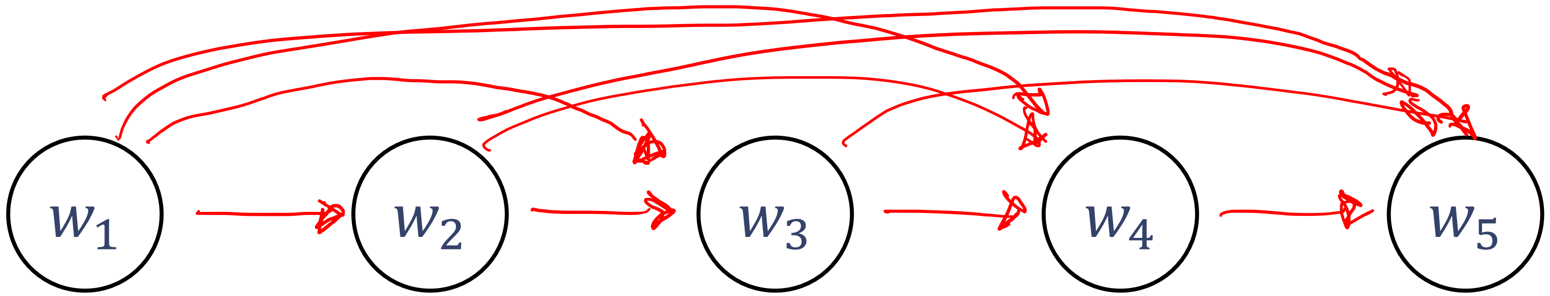
In-class Exercise: RNN as a DGM



$p(w_1, w_2, w_3, z_1, z_2, z_3) =$

Given a five-word sequence, $[w_1, w_2, w_3, w_4, w_5]$, how could we represent the implied probability distributions of an RNN LM as a directed graphical model?

$p(w_1, w_2, w_3, w_4, w_5) = p(w_1)\, p(w_2 | w_1)\, p(w_3 | w_2, w_1) \cdots p(w_5 | w_1, \dots w_4)$

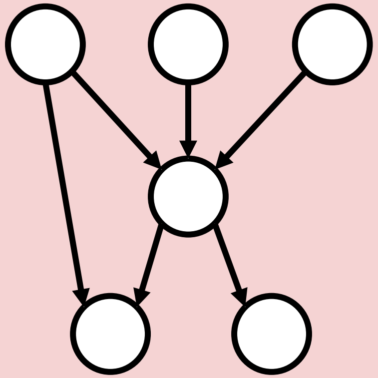


Locally Normalized vs. Globally Normalized

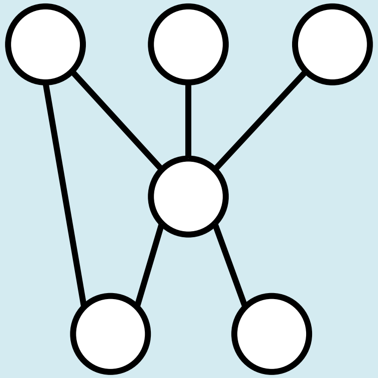
autoregressive LM

encoder-only Transformer

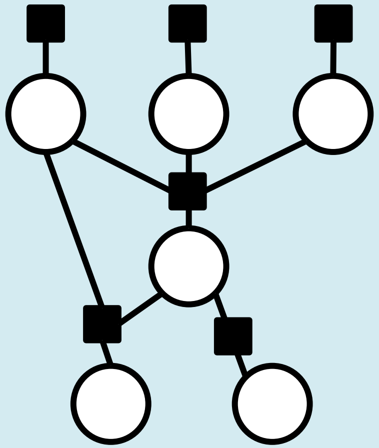
Directed Graphical Model



Undirected Graphical Model



Factor Graph



$$P(X_1, \ldots, X_T) = \prod_{t=1}^{T} P(X_t \mid \text{parents}(X_t))$$

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{\alpha} \psi_\alpha(\boldsymbol{x}_{\boldsymbol{\alpha}})$$

UNSUPERVISED LEARNING

Unsupervised Learning

Assumptions:

1. our data comes from some distribution $p^*(\mathbf{x}_0)$
2. we choose a distribution $p_\theta(\mathbf{x}_0)$ for which sampling $x_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

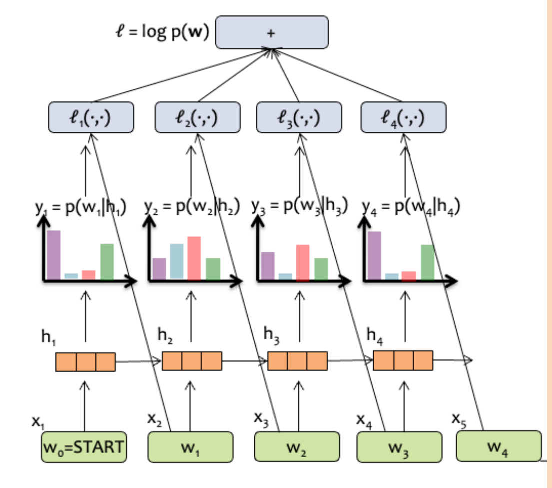
Goal: learn θ s.t. $p_\theta(\mathbf{x}_0) \approx p^*(\mathbf{x}_0)$

Unsupervised Learning

Assumptions:

1. our data comes from some distribution $p^*(\mathbf{x}_0)$
2. we choose a distribution $p_\theta(\mathbf{x}_0)$ for which sampling $x_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

Goal: learn θ s.t. $p_\theta(\mathbf{x}_0) \approx p^*(\mathbf{x}_0)$



Example: autoregressive LMs

- true $p^*(\mathbf{x}_0)$ is the (human) process that produced text on the web
- choose $p_\theta(\mathbf{x}_0)$ to be an autoregressive language model
  - autoregressive structure means that $p(\mathbf{x}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) \sim$ Categorical(.) and ancestral sampling is exact/efficient
- learn by finding

$$\theta \approx \text{argmax}_\theta \log(p_\theta(\mathbf{x}_0))$$

using gradient based updates on

$$\nabla_\theta \log(p_\theta(\mathbf{x}_0))$$

Unsupervised Learning

Assumptions:

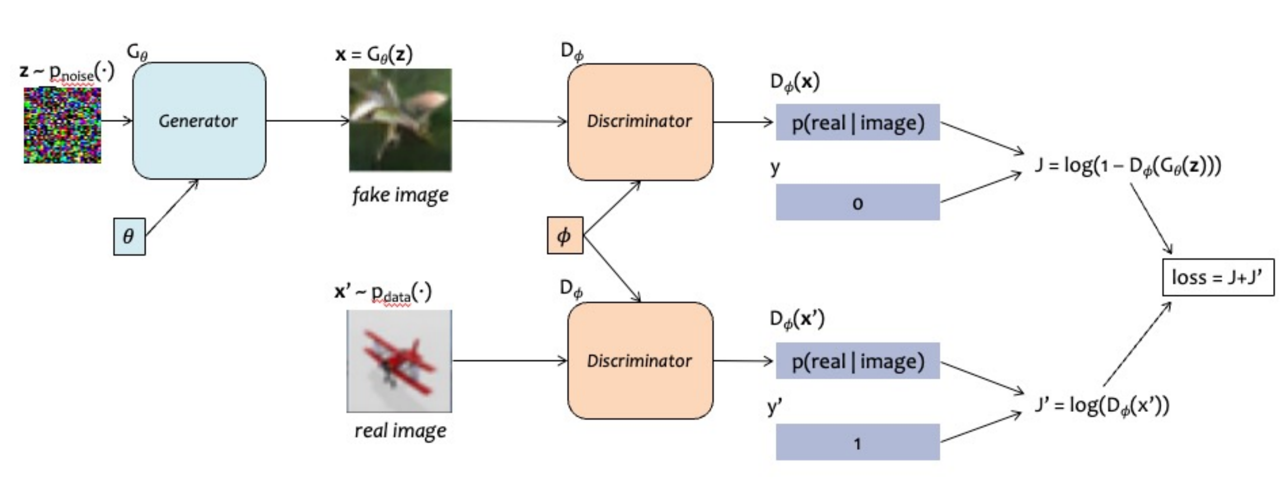
1. our data comes from some distribution $p^*(\mathbf{x}_0)$
2. we choose a distribution $p_\theta(\mathbf{x}_0)$ for which sampling $x_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

Goal: learn θ s.t. $p_\theta(\mathbf{x}_0) \approx p^*(\mathbf{x}_0)$

Example: GANs

- true $p^*(\mathbf{x}_0)$ is distribution over photos taken and posted to Flikr
- choose $p_\theta(\mathbf{x}_0)$ to be an expressive model (e.g. noise fed into inverted CNN) that can generate images
  - sampling is typically easy: $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$ and $\mathbf{x}_0 = f_\theta(\mathbf{z})$
- learn by finding $\theta \approx \text{argmax}_\theta \log(p_\theta(\mathbf{x}_0))$?
  - No! Because we can't even compute $\log(p_\theta(\mathbf{x}_0))$ or its gradient
  - Why not? Because the integral is intractable even for a simple 1-hidden layer neural network with nonlinear activation

$$p(\mathbf{x}_0) = \int_{\mathbf{z}} p(\mathbf{x}_0 \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$



so optimize a minimax loss instead

Unsupervised Learning

Assumptions:

1. our data comes from some distribution $p^*(\mathbf{x}_0)$
2. we choose a distribution $p_\theta(\mathbf{x}_0)$ for which sampling $x_0 \sim p_\theta(\mathbf{x}_0)$ is tractable

Goal: learn θ s.t. $p_\theta(\mathbf{x}_0) \approx p^*(\mathbf{x}_0)$

Example: VAEs / Diffusion Models

- true $p^*(\mathbf{x}_0)$ is distribution over photos taken and posted to Flikr
- choose $p_\theta(\mathbf{x}_0)$ to be an expressive model (e.g. noise fed into inverted CNN) that can generate images
  - sampling is will be easy
- learn by finding $\theta \approx \text{argmax}_\theta \log(p_\theta(\mathbf{x}_0))$?
  - Sort of! We can't compute the gradient $\nabla_\theta \log(p_\theta(\mathbf{x}_0))$
  - So we instead optimize a variational lower bound (more on that later)

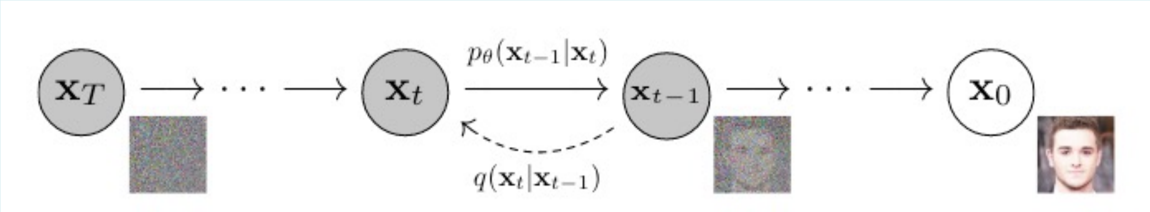


Figure from Ho et al. (2020)

Latent Variable Models

- For GANs and VAEs, we assume that there are (unknown) **latent variables** which give rise to our observations
- The **vector z** are those latent variables
- After learning a GAN or VAE, we can **interpolate** between images in latent **z** space

Figure 4: Top rows: Interpolation between a series of 9 random points in Z show that the space learned has smooth transitions, with every image in the space plausibly looking like a bedroom. In the 6th row, you see a room without a window slowly transforming into a room with a giant window. In the 10th row, you see what appears to be a TV slowly being transformed into a window.

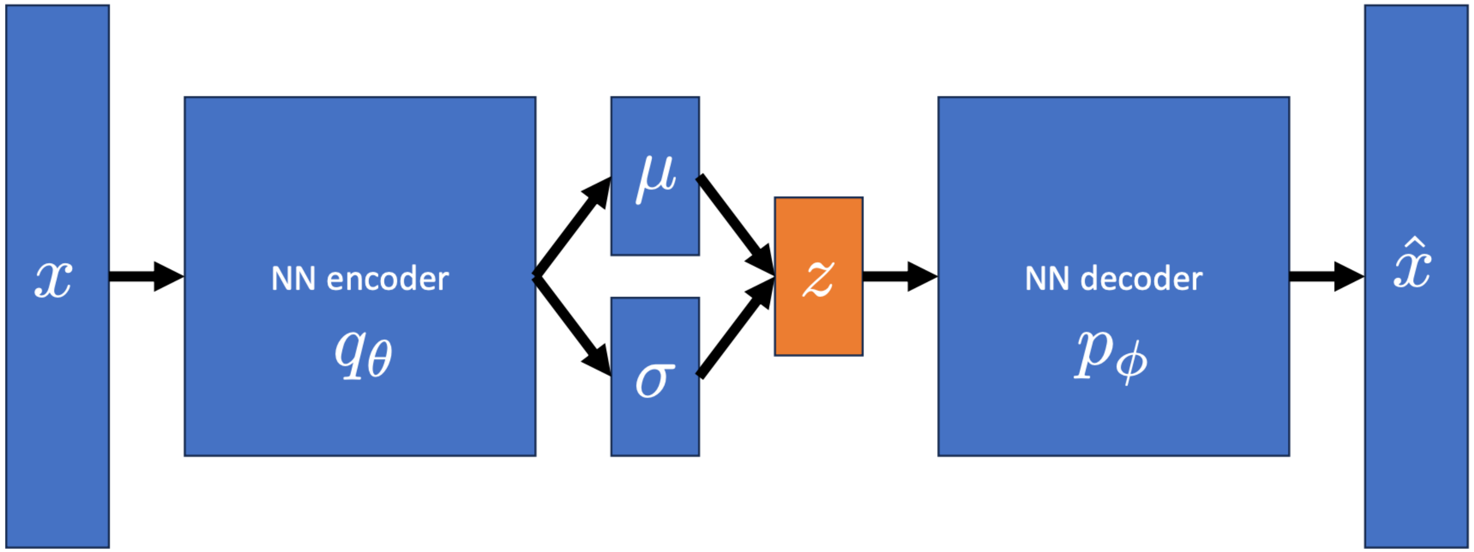
Figure from Radford et al. (2016)

From VAEs to Diffusion Models

- Next we will consider (1) **diffusion models** and (2) **variational autoencoders (VAEs)**
- 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 Autoencoder: Network Perspective

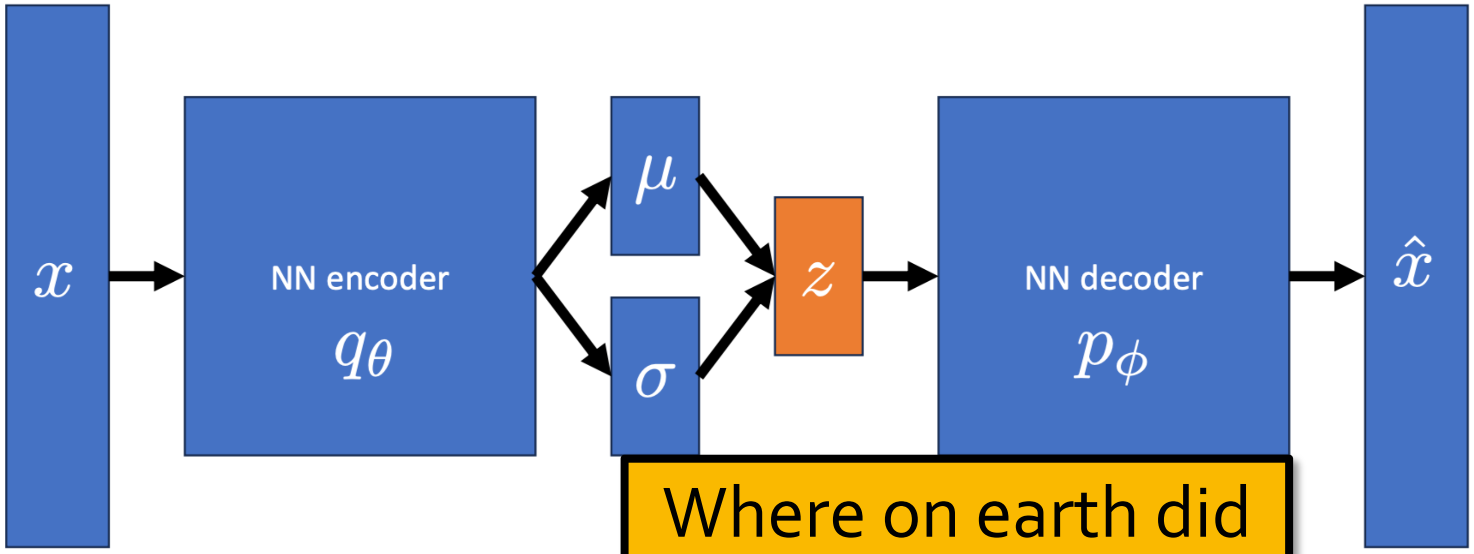


- Objective: minimize the negative log-likelihood of the dataset plus a *regularization term* that encourages a dense latent space

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

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

Variational Autoencoder: Network Perspective

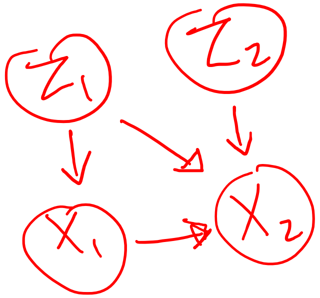


Where on earth did this objective function come from?

- Objective: minimize the nega… dataset plus a *regularization term* tha… nt space

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

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

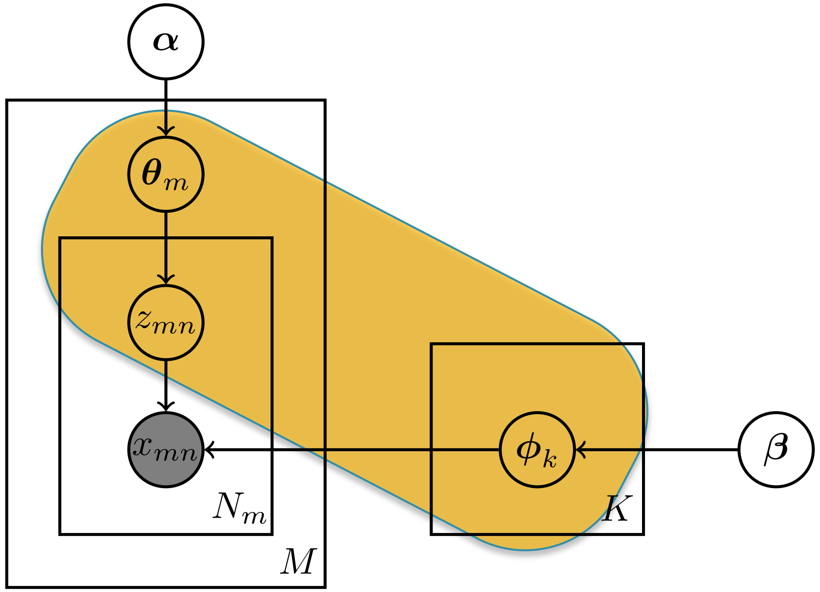


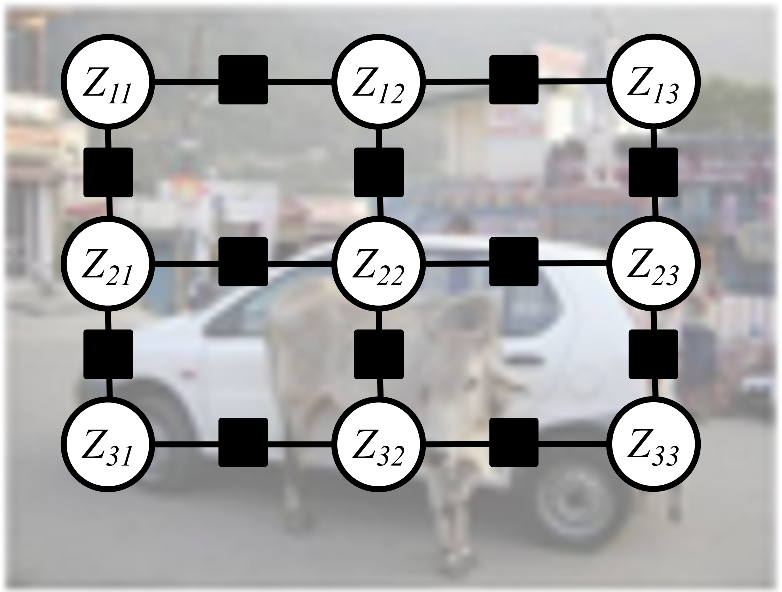
Variational Inference

$p(x_1, x_2, z_1, z_2) = p(z_1)\, p(z_2)\, p(x_1 | z_1)\, p(x_2 | x_1, z_1, z_2)$

A Common Problem:

- Suppose we have an interesting distribution p(**x, z**) and we wish to work with its posterior p(**z** | **x**)
- For training data **x** and latent variables **z,** estimating the posterior p(**z** | **x**) is usually intractable!





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

Variational Inference

A Common Problem:

- Suppose we have an interesting distribution p(**x**, **z**) and we wish to work with its posterior p(**z** | **x**) or the marginal p(**x**)
- For training data **x** and latent variables **z**, estimating the posterior p(**z** | **x**) or the marginal p(**x**) is usually intractable!

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

we assume p(**x**) is intractable

Question: Why is p(**x**) often intractable to compute?

Answer:

$$p(\vec{x}) = \int_{\vec{z}} p(\vec{x}, \vec{z}) \, d\vec{z}$$

$$p(\vec{x}) = \sum_{\vec{z}} p(\vec{x}, \vec{z})$$

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

Variational Inference

A Common Problem:

- Suppose we have an interesting distribution p(**x**, **z**) and we wish to work with its posterior p(**z** | **x**)
- For training data **x** and latent variables **z**, estimating the posterior p(**z** | **x**) is usually intractable!

Solution:

- Approximate p(**z** | **x**) with a simpler q(**z** | **x**)
- Typically q(**z** | **x**) has more independence assumptions than p(**z** | **x**), which is fine b/c q(**z** | **x**) is tuned for a specific **x**
- **Key idea**: pick a single q(**z** | **x**) 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:

- $q(\mathbf{z} \mid \mathbf{x})$: the **variational approximation**
- Q: the **variational family**
- Usually $q_\theta(\mathbf{z} \mid \mathbf{x})$ 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

- **Question:** Do we learn a single distribution $q_\theta(\mathbf{z} \mid \mathbf{x})$ for all $\mathbf{x}$'s?
- **Answer:** Not necessarily, it's quite common to infer a separate q_θ for each $\mathbf{x}$!
  - Consider the sampling equivalent of this:
    - you could draw samples $z^{(i)} \sim p(\mathbf{z} \mid \mathbf{x}')$
    - then train some simple $q_\theta(\mathbf{z} \mid \mathbf{x}')$ on $z^{(1)}, z^{(2)}, \ldots, z^{(N)}$
    - hope that the sample adequately represents the posterior for the given $\mathbf{x}'$
  - 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 $q_\theta(\mathbf{z} \mid \mathbf{x})$ such that V.I. is efficient

$\boxed{\text{VAE}}$

$$p(x,z) = p(x|z)\,p(z)$$

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

$$p(x|z) = \begin{cases} 1 & \text{if } x = \text{MLP}(z) \\ 0 & \text{otherwise} \end{cases}$$

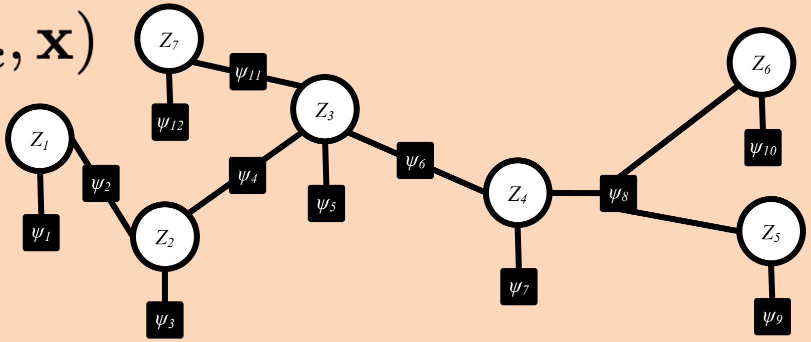
Narrative adapted from Jason Eisner's High-Level Explanation of VI: https://www.cs.jhu.edu/~jason/tutorials/variational.html

THE MEAN FIELD APPROXIMATION

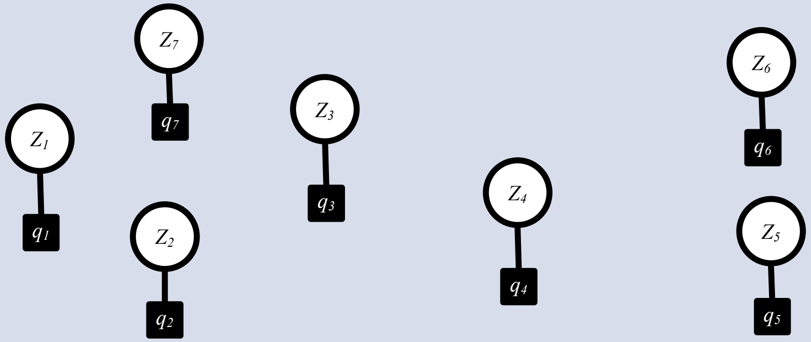
Mean Field Approximation

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

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



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

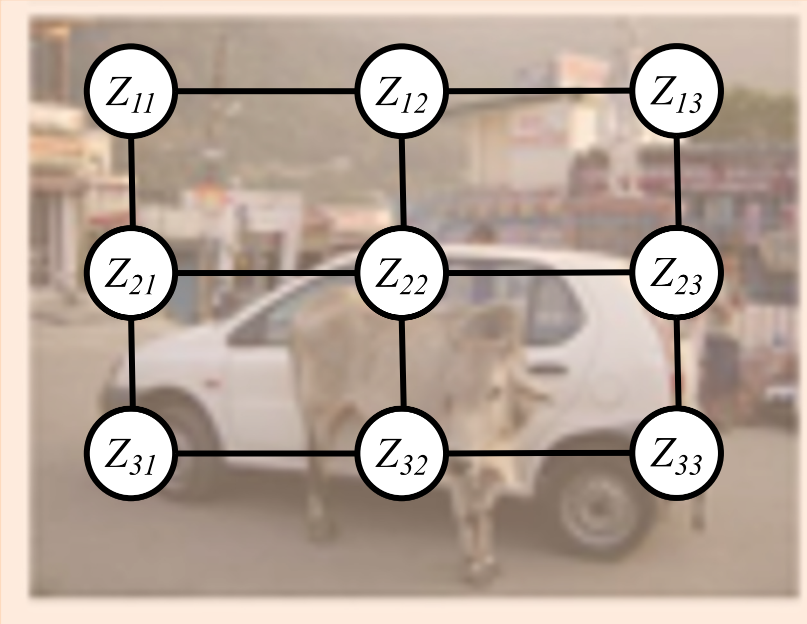


Mean Field Approximation

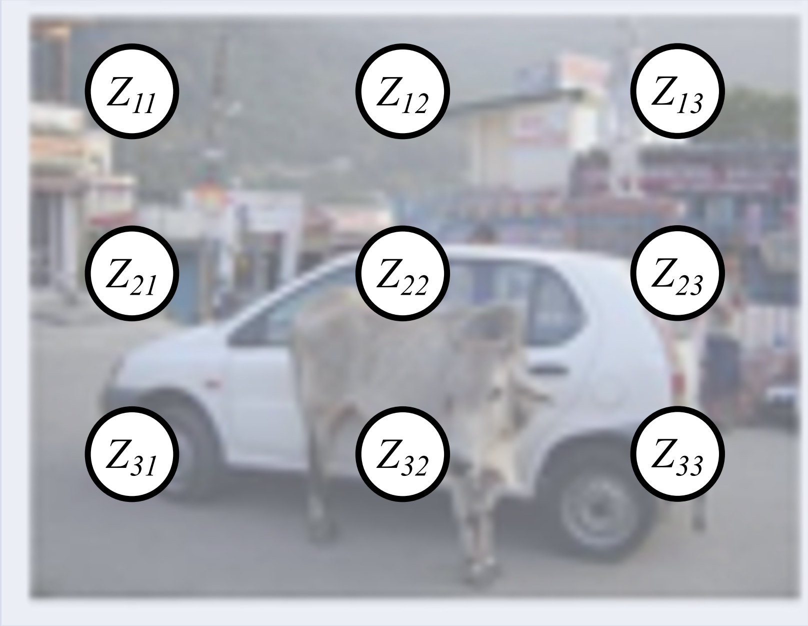
The **mean field approximation** assumes our variational approximation $q_\theta(\mathbf{z})$ treats each variable as independent

Ising Model

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



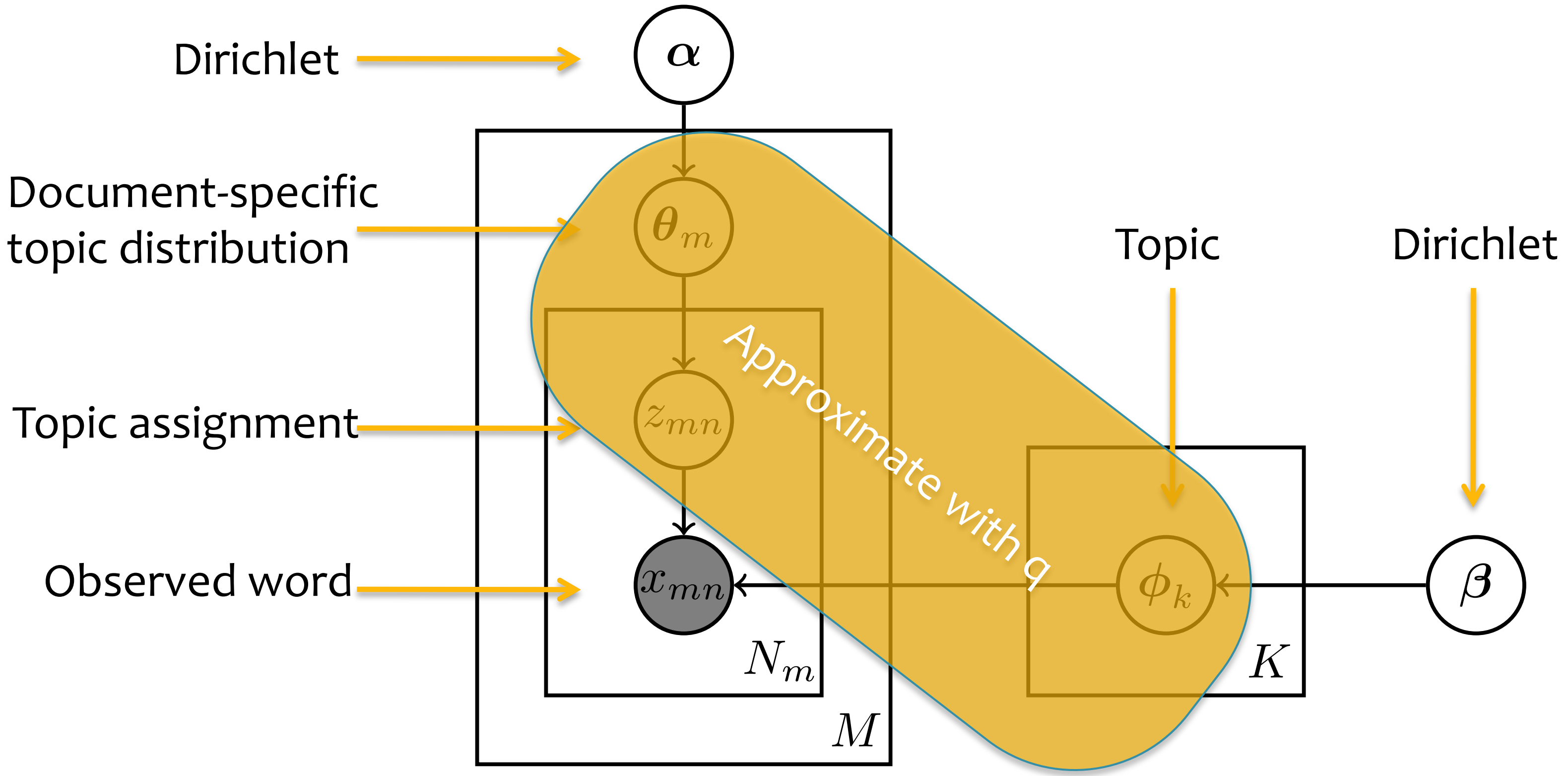
$$q_\theta(\mathbf{z}) = \prod_{t=1}^{T} q_t(z_t)$$



Mean Field Approximation

Latent Dirichlet Allocation (LDA)

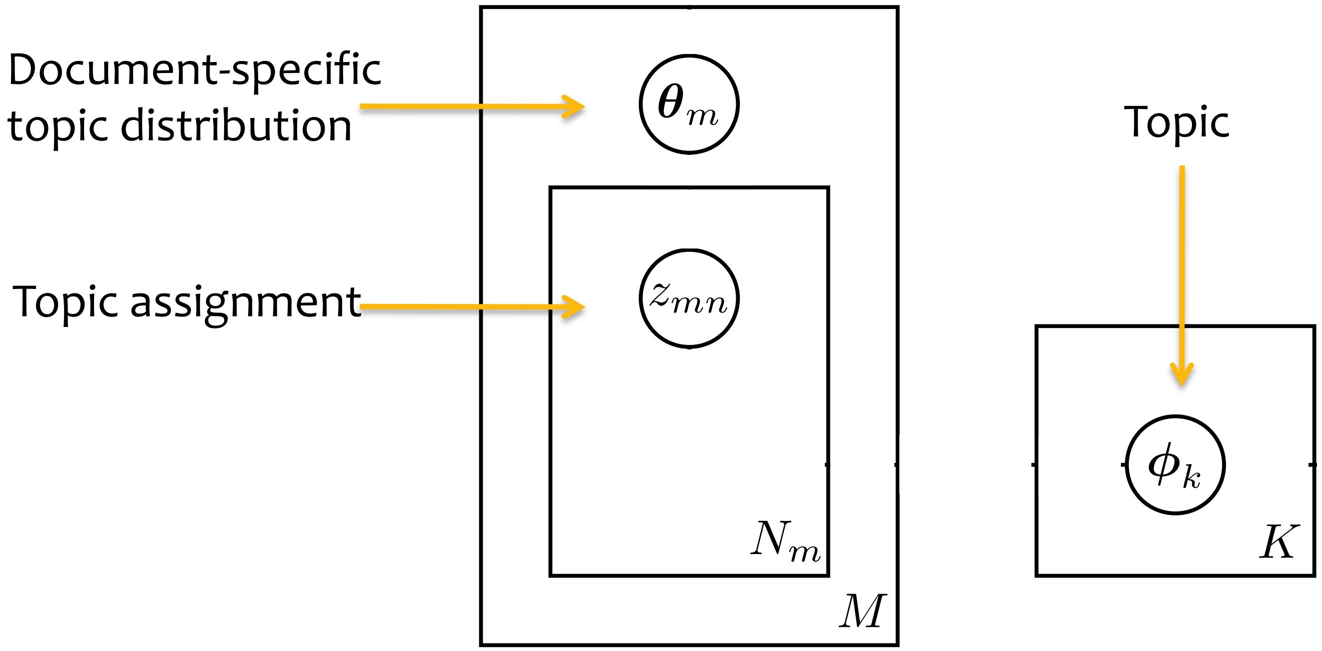
- Uncollapsed Variational Inference, aka. Explicit V.I. (original distribution)



Mean Field Approximation

Latent Dirichlet Allocation (LDA)

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



MEAN FIELD VARIATIONAL INFERENCE

KL Divergence

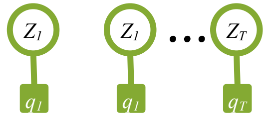
Recall…

- *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 ∈ $\mathcal{X}$
  - KL(q || p) ≥ 0

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 $q_\theta(\mathbf{z} \mid \mathbf{x}) \approx p_\alpha(\mathbf{z} \mid \mathbf{x})$ for each $\mathbf{x}$
3. *Mean Field*: assume $q_\theta(\mathbf{z} \mid \mathbf{x}) = \prod_t q_t(z_t \mid x; \theta)$
   

   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)

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

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

equivalent

5. *Optimization Algorithm*: various options
   – e.g. coordinate descent repeatedly picks the best $q_t(z_t \mid x)$ based on the other $\{ q_s(z_s \mid x) \}_{s \neq t}$ being fixed
   – e.g. gradient descent optimizes a surrogate objective ELBO(q_θ) to find θ

Optimizing KL Divergence

- *Question*: How do we minimize KL?

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

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

Why we can't compute KL…

$$\mathsf{KL}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x})) = E_{q(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{q(\mathbf{z} \mid \mathbf{x})}{p(\mathbf{z} \mid \mathbf{x})}\right)\right]$$

$$= E_{q(\mathbf{z}|\mathbf{x})}[\log q(\mathbf{z} \mid \mathbf{x})] - E_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z} \mid \mathbf{x})]$$

$= p(x,z)/p(x)$

$$= E_{q(\mathbf{z}|\mathbf{x})}[\log q(\mathbf{z} \mid \mathbf{x})] - E_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}, \mathbf{z})] + \boxed{E_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x})]}$$

$$= E_{q(\mathbf{z}|\mathbf{x})}[\log q(\mathbf{z} \mid \mathbf{x})] - E_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}, \mathbf{z})] + \underbrace{\log p(\mathbf{x})}$$

this expectation does not depend on q

we assumed this is intractable to compute!

Optimizing KL Divergence

- *Question*: How do we minimize KL?

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

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

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

The ELBO for a DGM

Here is why…

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

dropping the
intractable term
gives the ELBO

ELBO as Objective Function

What does maximizing ELBO(q_θ) accomplish?

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

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

Theorem: For any q, $\log p(\mathbf{x}) \geq \text{ELBO}(q)$

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

Note:

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

$$\text{KL}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x})) = E_{q(\mathbf{z}|\mathbf{x})}\left[\log q(\mathbf{z} \mid \mathbf{x})\right] - E_{q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x},\mathbf{z})\right] + \cancel{E_{q(\mathbf{z}|\mathbf{x})}}\left[\log p(\mathbf{x})\right]$$

Proof #1:

1. $\log p(x) = KL(q \| p) + ELBO(q)$
2. $KL(q \| p) \geq 0$
3. $\log(p(x)) \geq ELBO(q)$

Takeaways:

1. in variational inference, we find the q that gives the **tightest bound** on the normalization constant for p(z | x)
2. maximizing the ELBO is equivalent to minimizing KL
3. maximizing the ELBO is maximizing a lower bound on the likelihood p(**x**)

ELBO's relation to log p(x)

Theorem:

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

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

Proof #2:

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

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

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

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

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

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

$$\Rightarrow \log p(x) \geq ELBO(q)$$

(Through the Lens of Variational Inference)

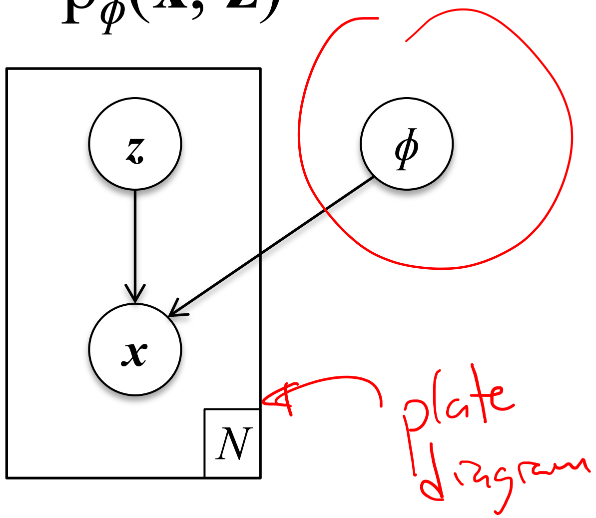
VARIATIONAL AUTOENCODERS

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

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



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

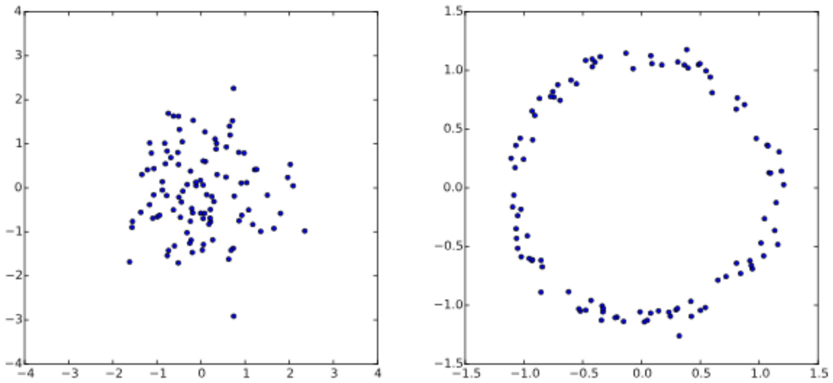


Figure from Doersch (2016)

The Something-like-a-VAE Model

- Consider a model p(**x**, **z**) = p(**x** | **z**) p(**z**)
  - where p(**z**) is a N(0, **I**)
  - where **x** = $\cancel{g}$(**z**/10 + **z**/||**z**||) $= g_\phi(\vec{z})$
    i.e. we don't use parameters ϕ
- Trivially, we can draw samples of **z** and directly convert them to values **x**

The VAE Model

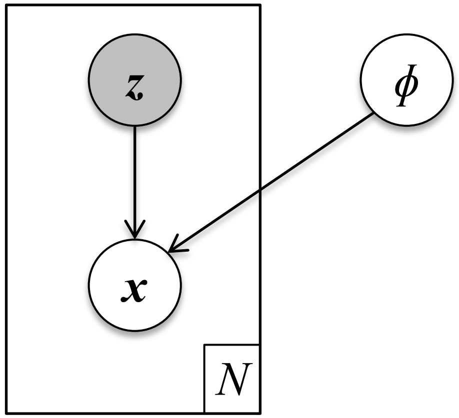
- The directed graphical model for VAE is the same as for the silly model above, and it's quite simple (ignoring the neural net details that give rise to **x**)
- **Key idea of VAE:** define $g_\phi(z)$ as a neural net and learn ϕ from data

Variational Autoencoders

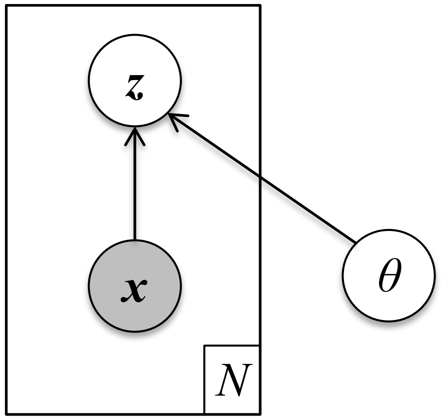
Neural Network Perspective

- 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)

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



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



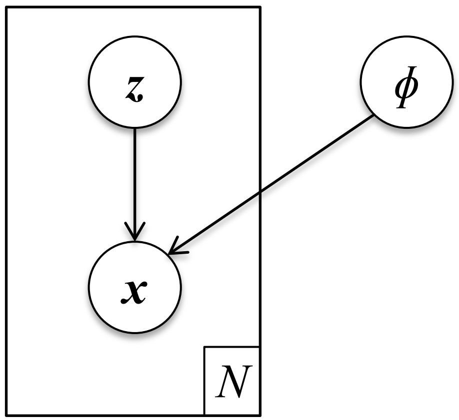
Variational Autoencoders

Graphical Model Perspective

- We can also view the VAE from the perspective of variational inference
- In this case we have two distributions:
  - **model:** $p_\phi(\mathbf{x}, \mathbf{z}) = p_\phi(\overset{x|z}{\cancel{\mathbf{z} \mid \mathbf{x}}})\, p(\mathbf{z})$
  - **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

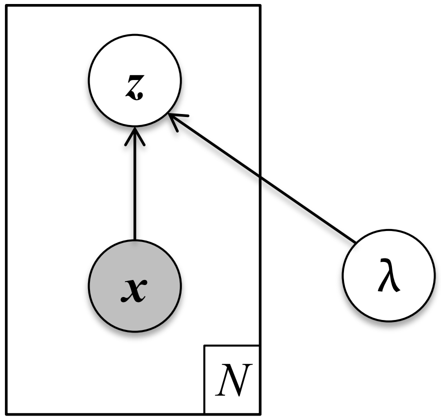
$$p_\phi(\mathbf{x}, \mathbf{z})$$

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

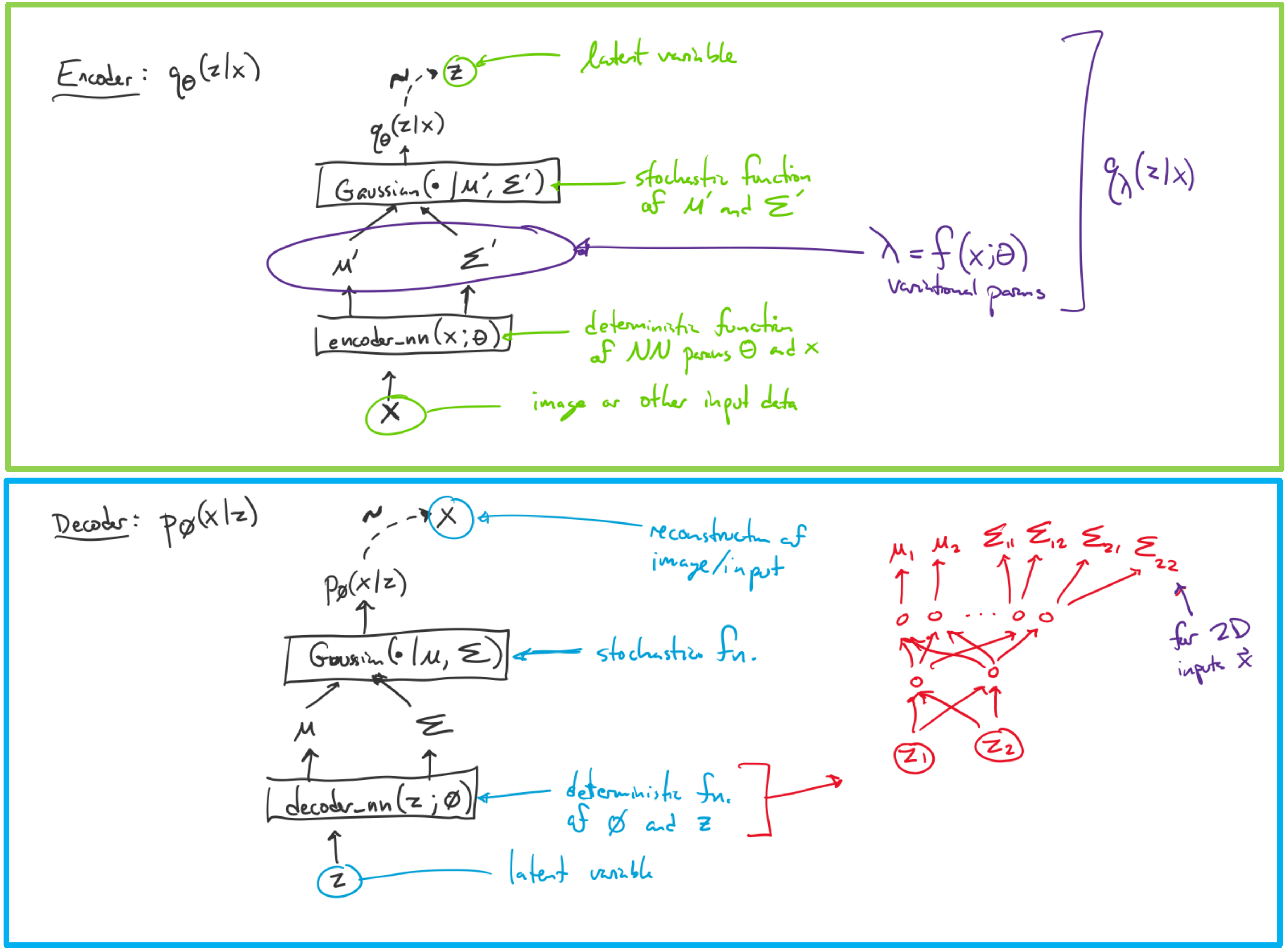


$$q_\lambda(\mathbf{z} \mid \mathbf{x})$$

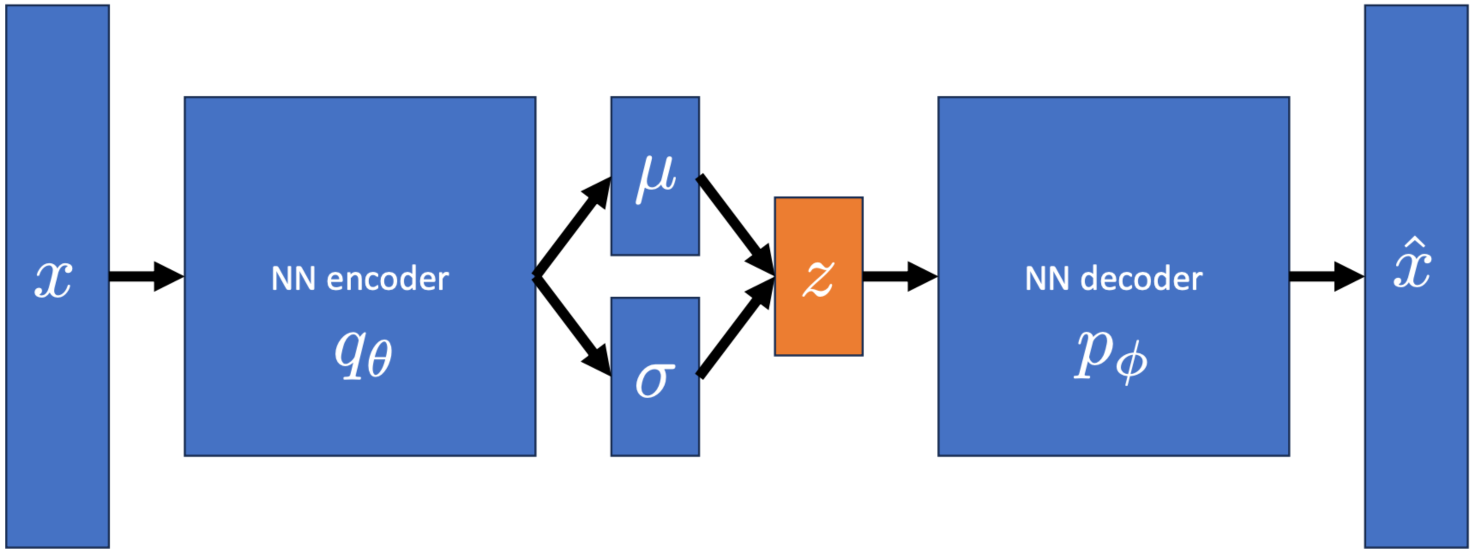
$$\lambda = f(x;\ \theta)$$



VAEs: Neural Network View



Variational Autoencoder: Network Perspective

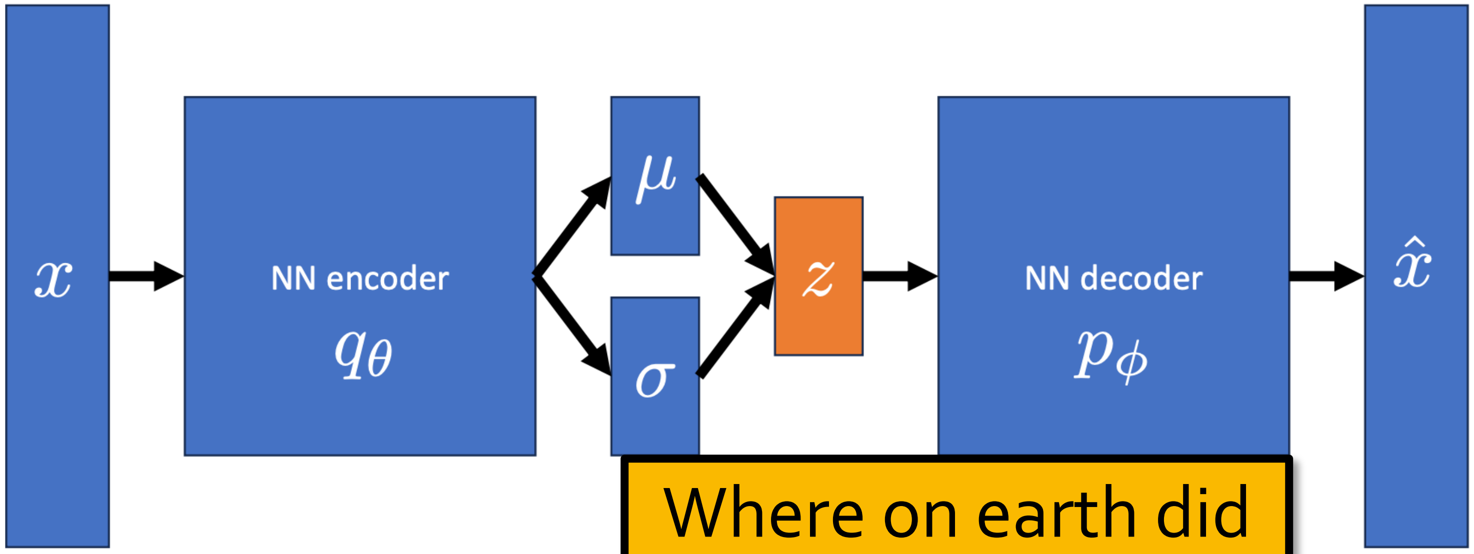


- Objective: minimize the negative log-likelihood of the dataset plus a *regularization term* that encourages a dense latent space

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

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

Variational Autoencoder: Network Perspective



Where on earth did this objective function come from?

- Objective: minimize the nega… dataset plus a *regularization term* tha… nt space

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

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

VAE Objective Function

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

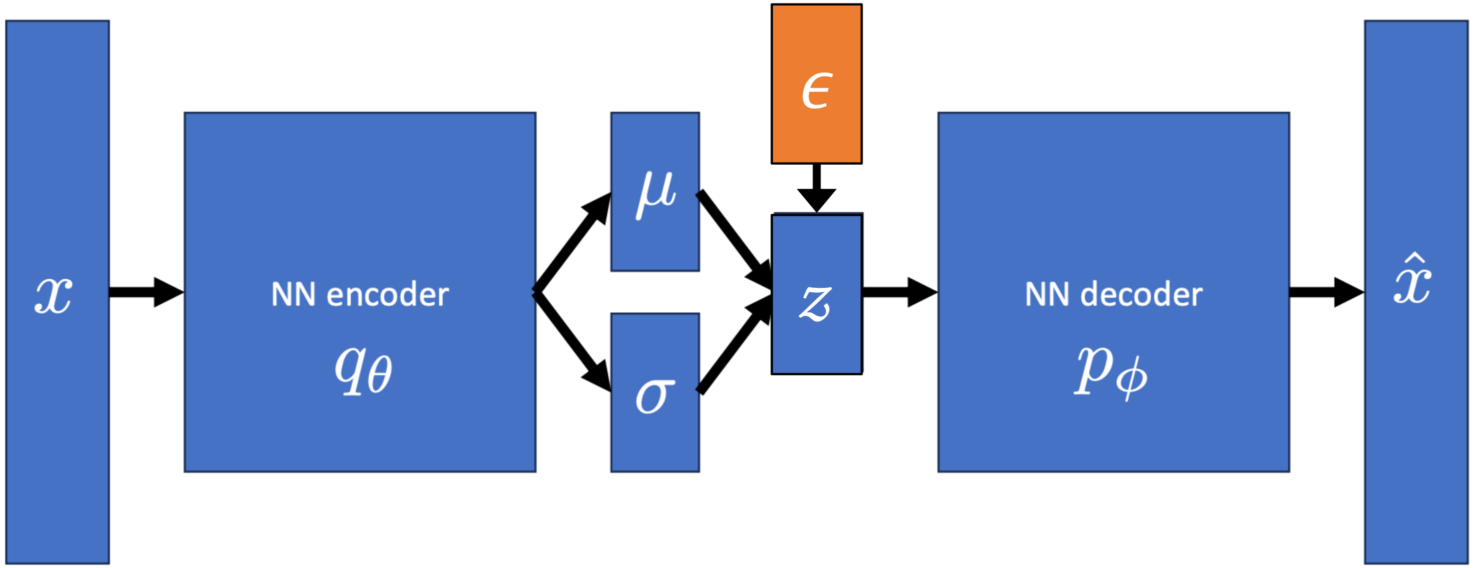
$$= -\Big(E_q[\log p(x|z)] + \underline{E_q[\log p(z)] - E_q[\log q(z|x)]}\Big)$$

$$= -\Big(E_q[\log p(x|z)] + -E_q\left[\frac{\log q(z|x)}{\log p(z)}\right]$$

$$=$$

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

Reparameterization Trick



- Objective: minimize the negative log-likelihood of the dataset plus a *regularization term* that encourages a dense latent space

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

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

for $\boldsymbol{z}_s(\boldsymbol{\theta}) = \mu_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) + \boldsymbol{\sigma}_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) \odot \boldsymbol{\epsilon}_s$ where $\boldsymbol{\epsilon}_s \sim N(\boldsymbol{0}, I)$

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

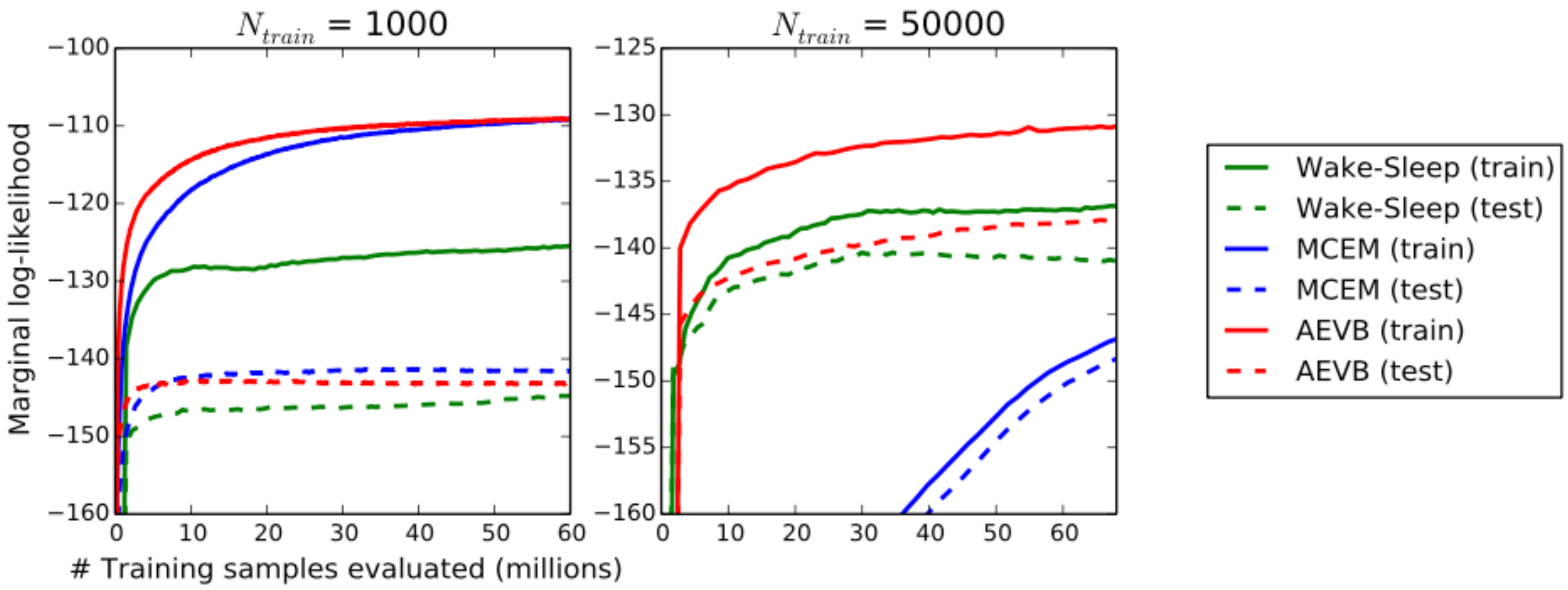
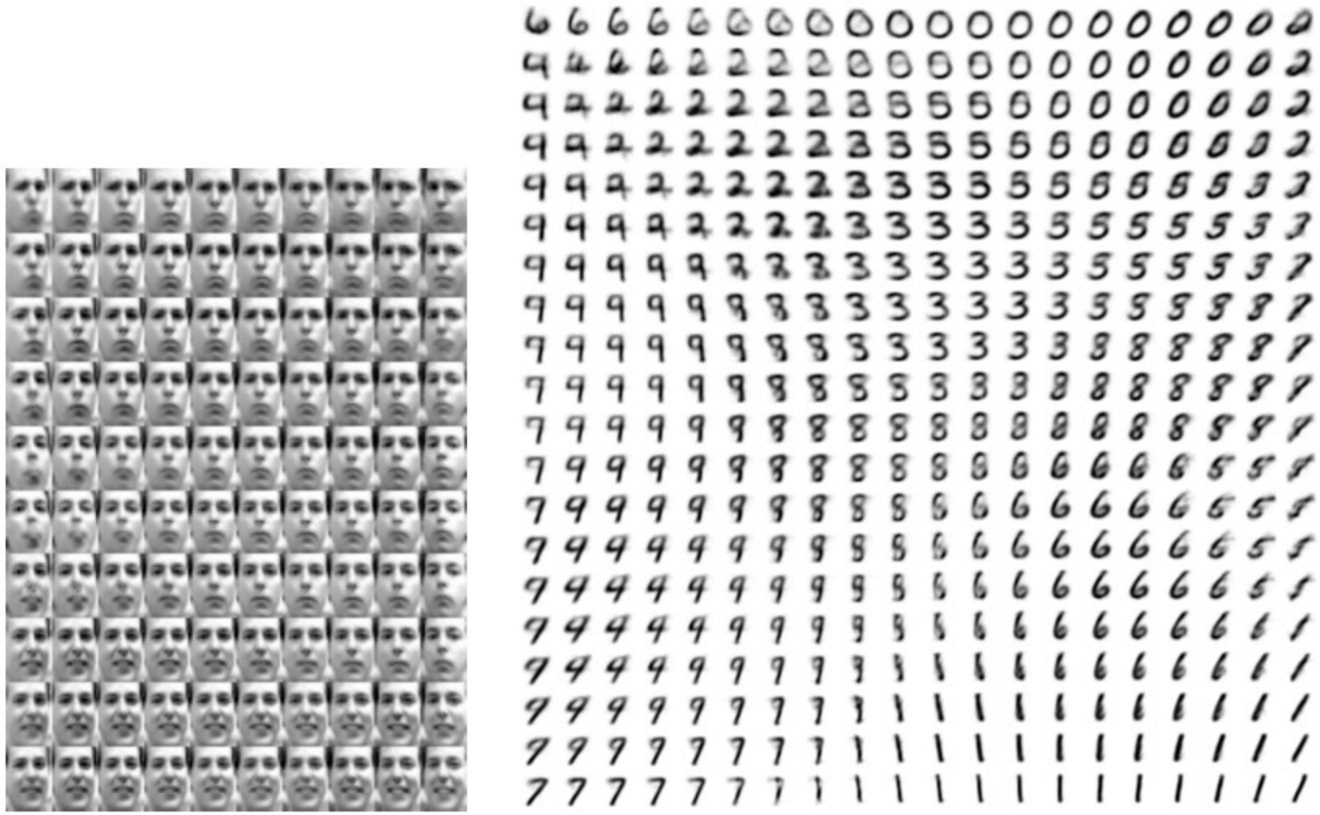


Figure 3: Comparison of AEVB to the wake-sleep algorithm and Monte Carlo EM, in terms of the estimated marginal likelihood, for a different number of training points. Monte Carlo EM is not an on-line algorithm, and (unlike AEVB and the wake-sleep method) can't be applied efficiently for the full MNIST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

Figure from Kingma & Welling (2014)

VAEs for Image Generation

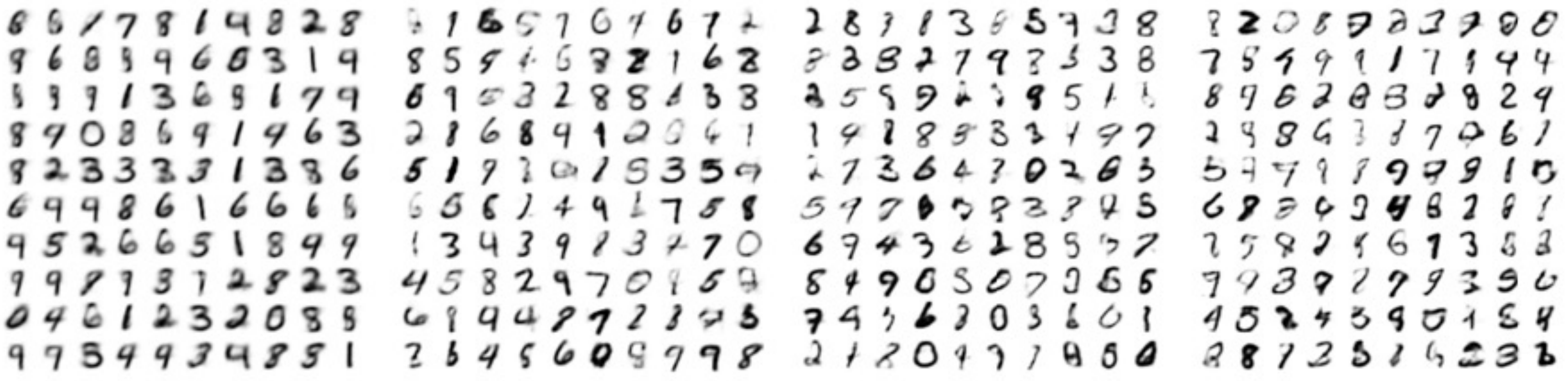


(a) Learned Frey Face manifold (b) Learned MNIST manifold

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 ϕ
- $q_\theta(\mathbf{z} \mid \mathbf{x})$ is a multivariate Gaussian with mean and variance computed by an LSTM with parameters θ

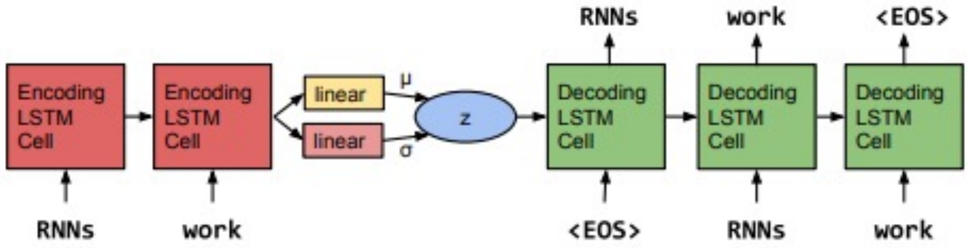


Figure 1: The core structure of our variational autoencoder language model. Words are represented using a learned dictionary of embedding vectors.

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

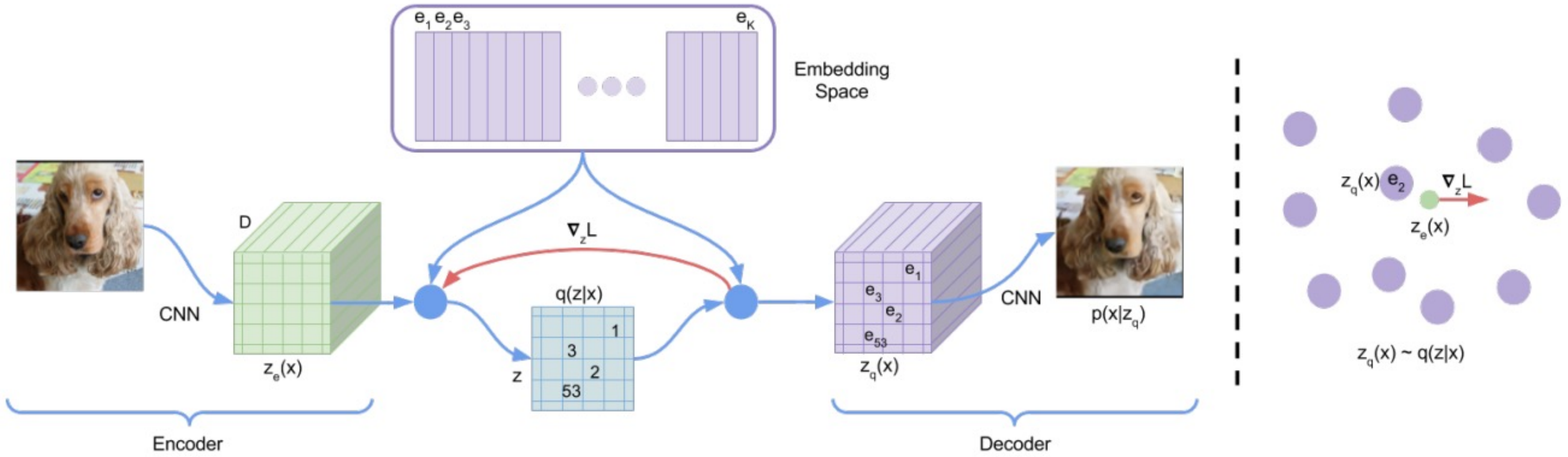


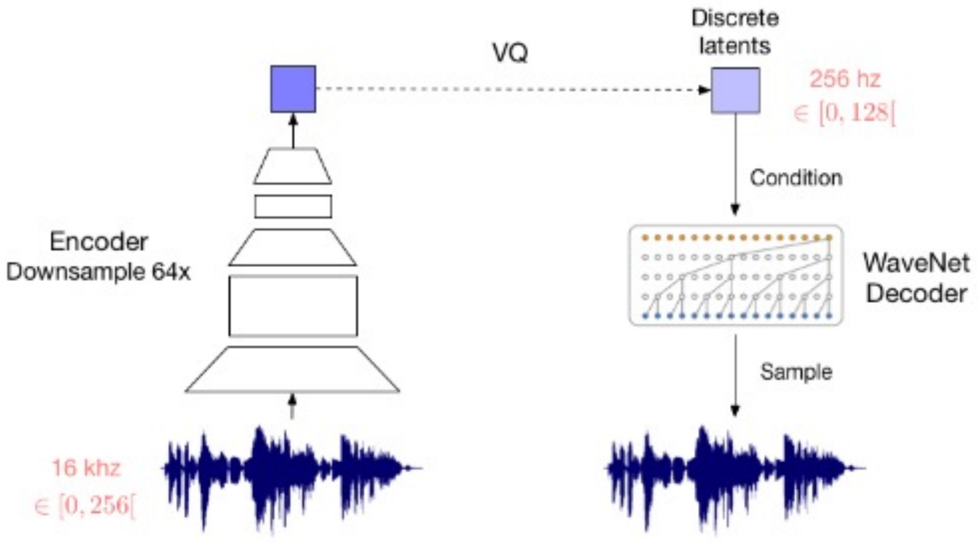
Figure 1: Left: A figure describing the VQ-VAE. Right: Visualisation of the embedding space. The output of the encoder $z(x)$ is mapped to the nearest point e_2. The gradient $\nabla_z L$ (in red) will push the encoder to change its output, which could alter the configuration in the next forward pass.

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

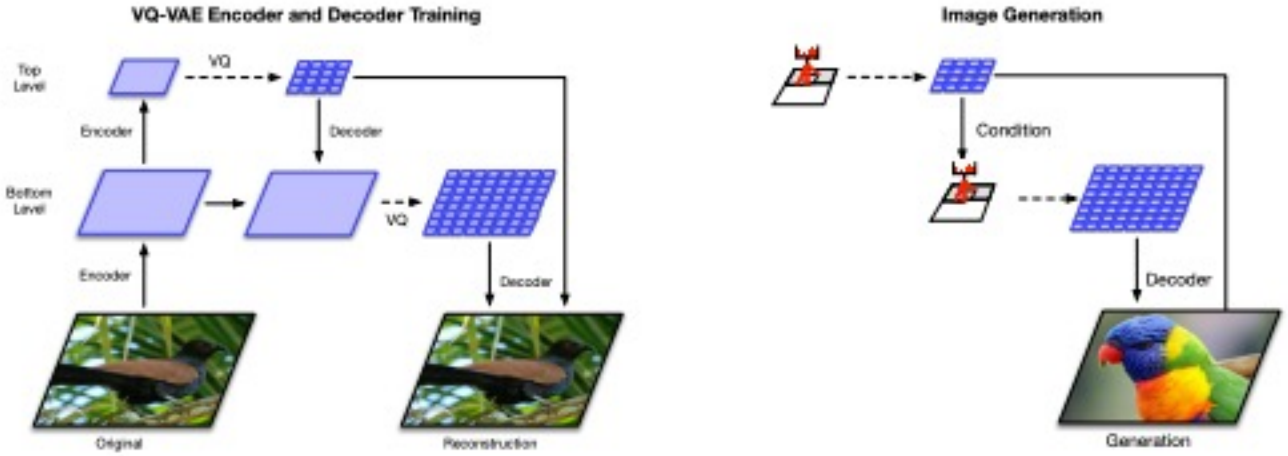


https://avdnoord.github.io/homepage/vqvae

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

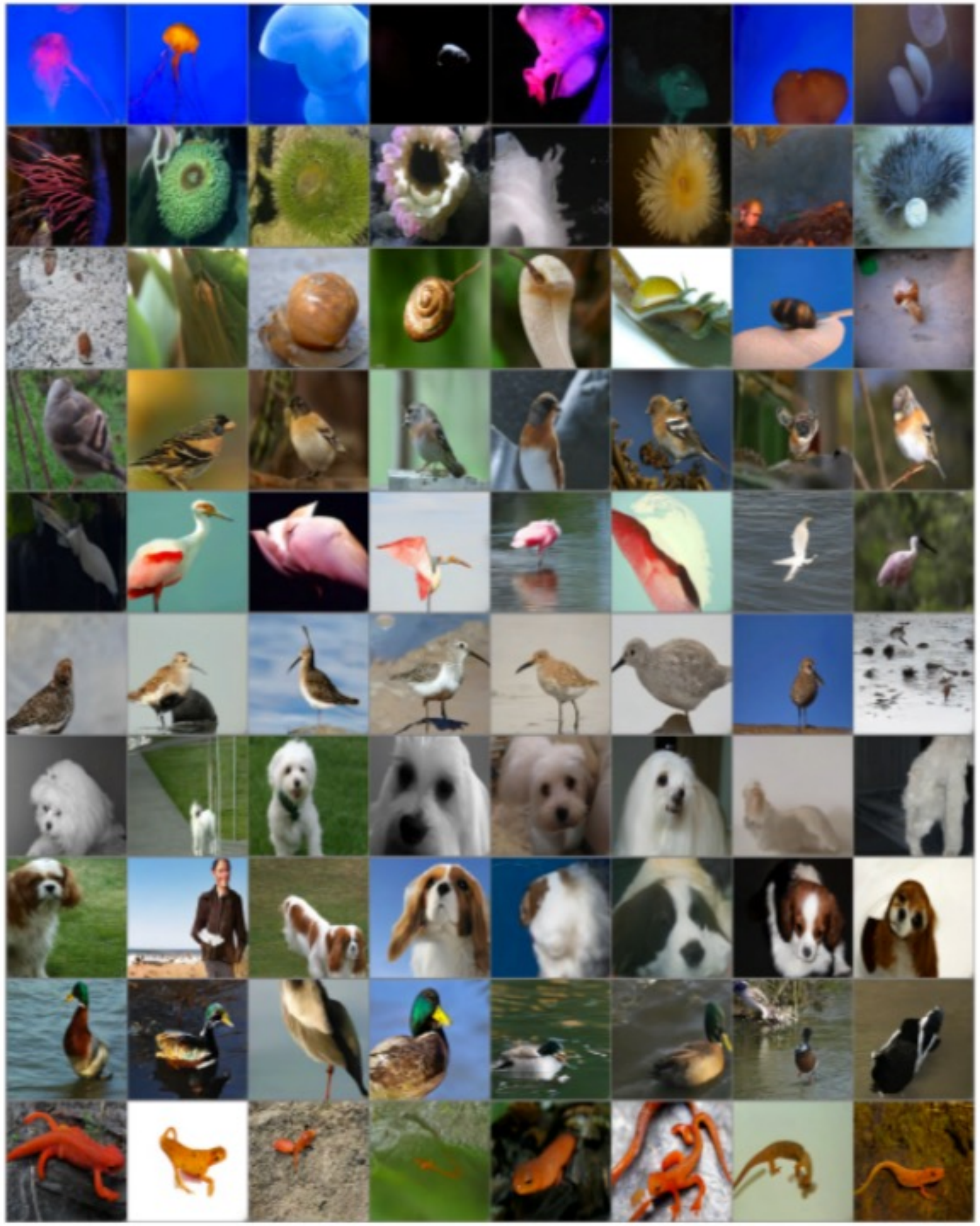


Figure 4: Class conditional random samples. Classes from the top row are: 108 sea anemone, 109 brain coral, 114 slug, 11 goldfinch, 130 flamingo, 141 redshank, 154 Pekinese, 157 papillon, 97 drake, and 28 spotted salamander.

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)

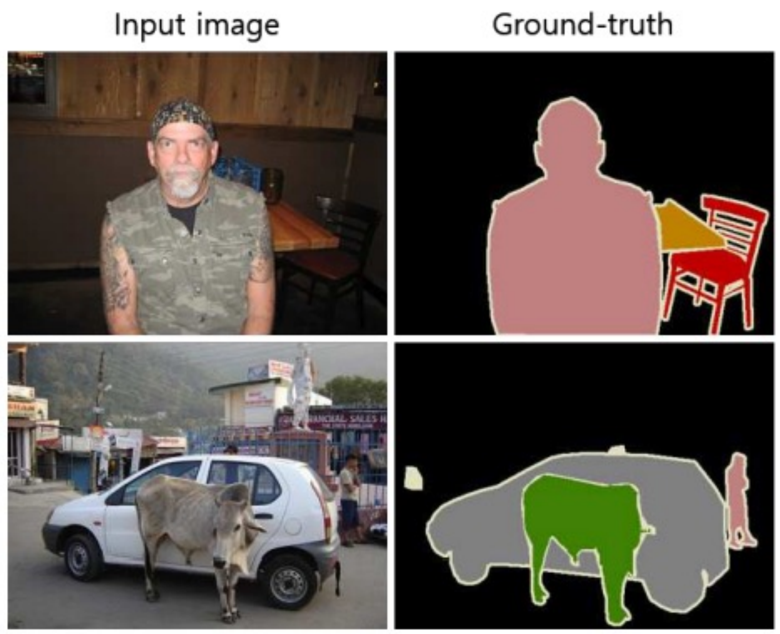
- VQ-VAE-2 extended the original idea by learning two levels (bottom and top) and a strong prior over the latent space
- Samples from this new model can be convincing even at high-fidelity

Figure from Razavi et al. (2019)

U-NET

Semantic Segmentation

- Given an image, predict a label for every pixel in the image
- Not merely a classification problem, because there are strong correlations between pixel-specific labels



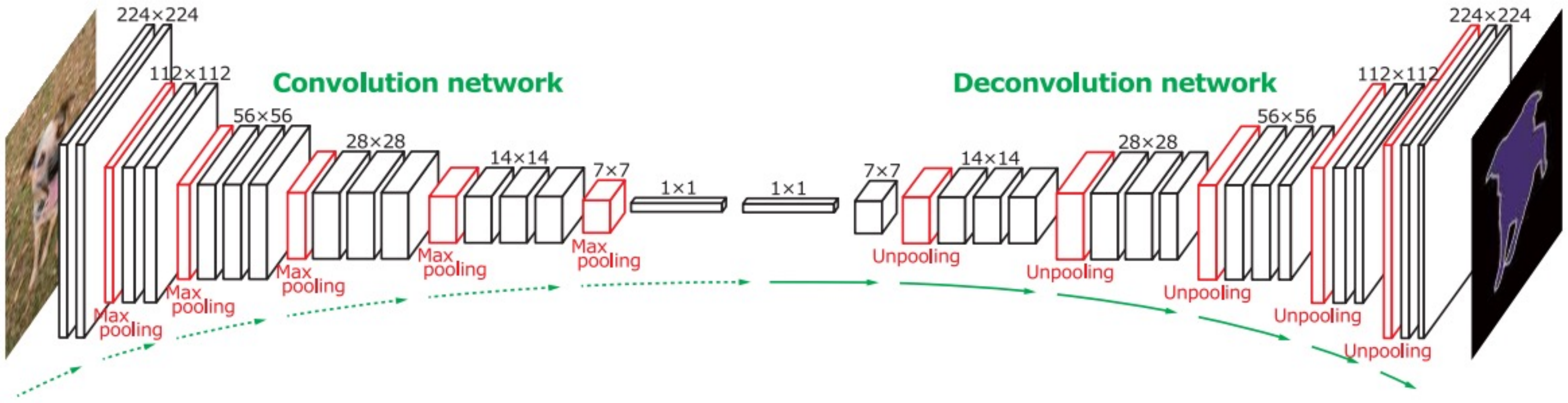


Figure from https://openaccess.thecvf.com/content_iccv_2015/papers/Noh_Learning_Deconvolution_Network_ICCV_2015_paper.pdf

Instance Segmentation

- Predict per-pixel labels as in semantic segmentation, but differentiate between different instances of the same label
- *Example*: if there are two people in the image, one person should be labeled **person-1** and one should be labeled **person-2**

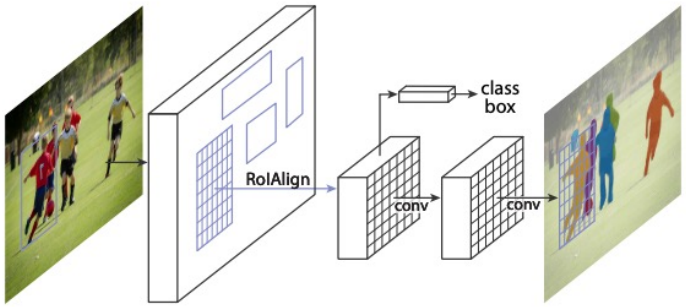
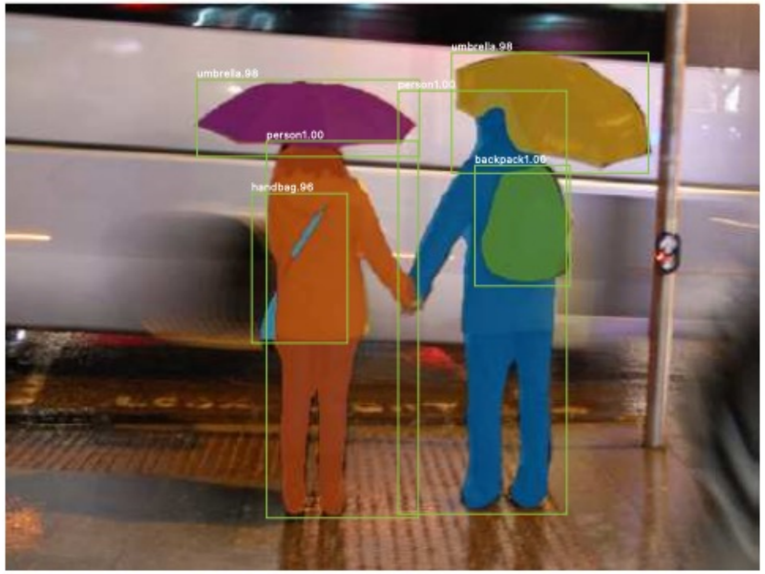


Figure 1. The **Mask R-CNN** framework for instance segmentation.



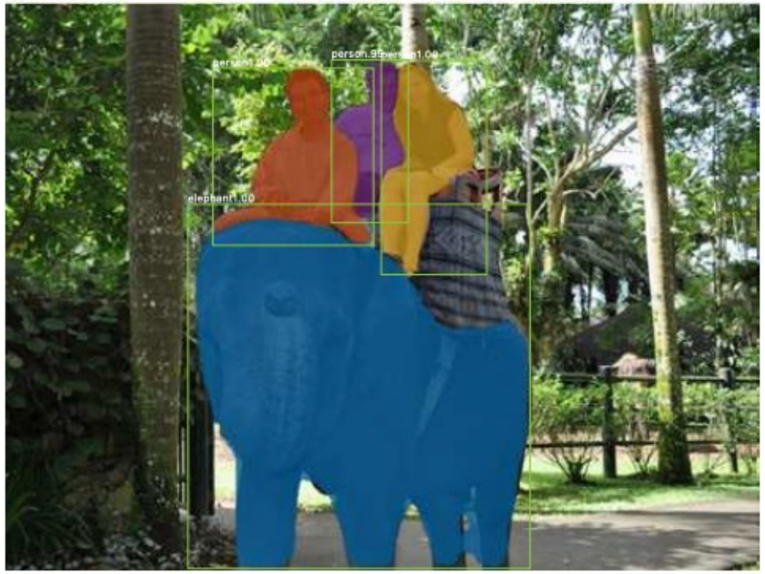


Figure from https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf

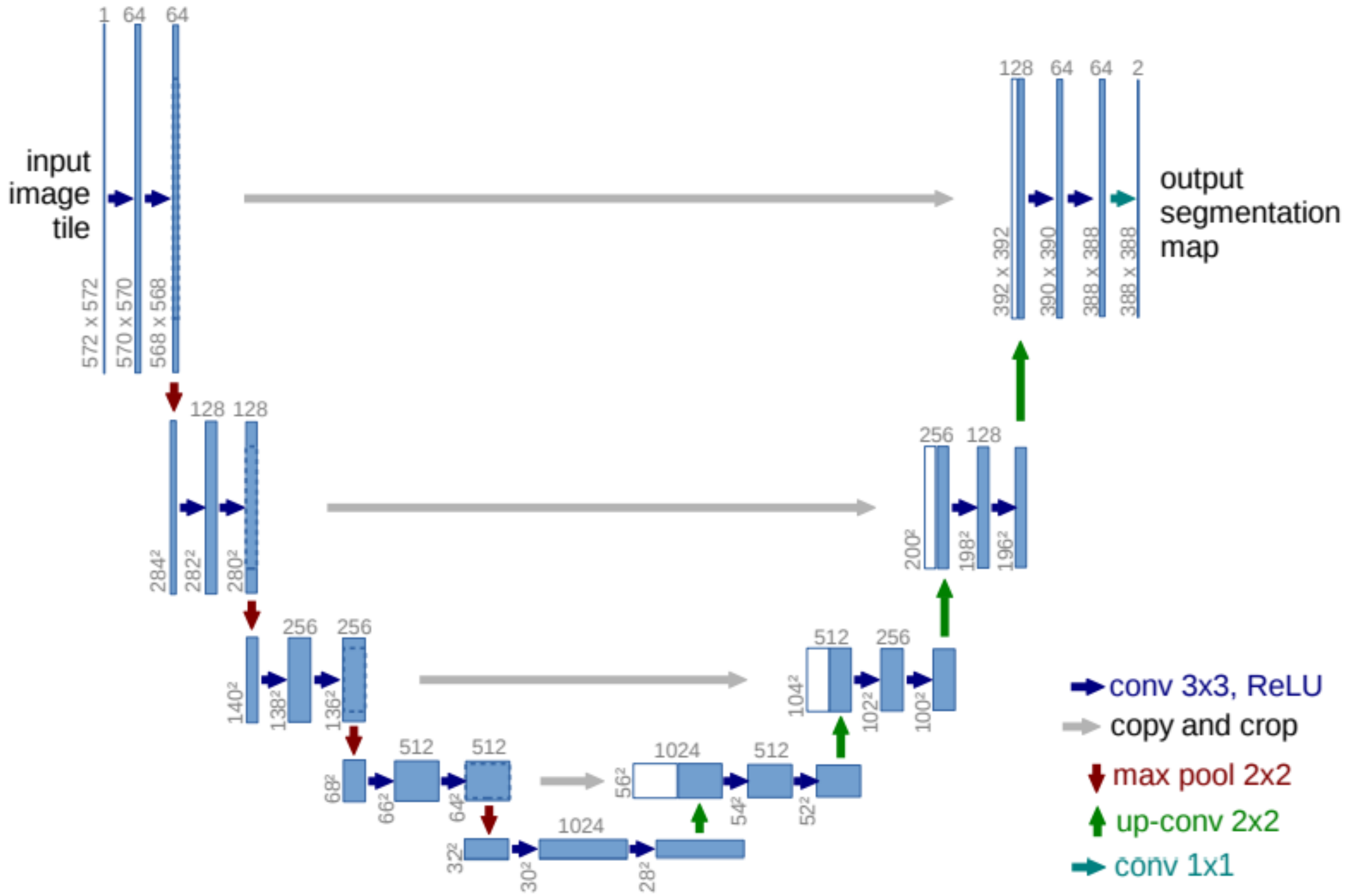
U-Net

Contracting path

- block consists of:
  - 3x3 convolution
  - 3x3 convolution
  - ReLU
  - max-pooling with stride of 2 (downsample)
- repeat the block N times, doubling number of channels

Expanding path

- block consists of:
  - 2x2 convolution (upsampling)
  - concatenation with contracting path features
  - 3x3 convolution
  - 3x3 convolution
  - ReLU
- repeat the block N times, halving the number of channels



U-Net

- Originally designed for applications to biomedical segmentation
- Key observation is that the output layer has the **same** dimensions as the input image (possibly with different number of channels)

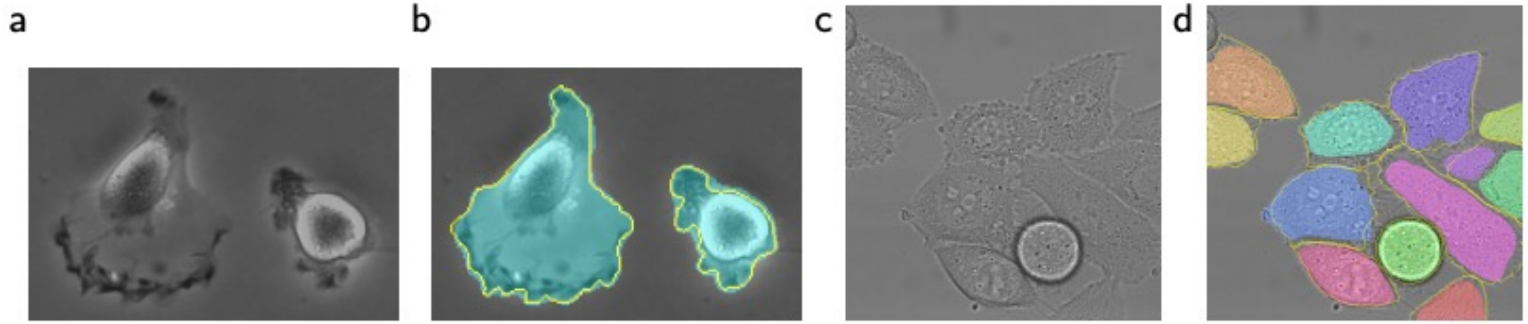


Fig. 4. Result on the ISBI cell tracking challenge. (**a**) part of an input image of the "PhC-U373" data set. (**b**) Segmentation result (cyan mask) with manual ground truth (yellow border) (**c**) input image of the "DIC-HeLa" data set. (**d**) Segmentation result (random colored masks) with manual ground truth (yellow border).