CS 461: Machine Learning Principles

Class 23: Nov. 25

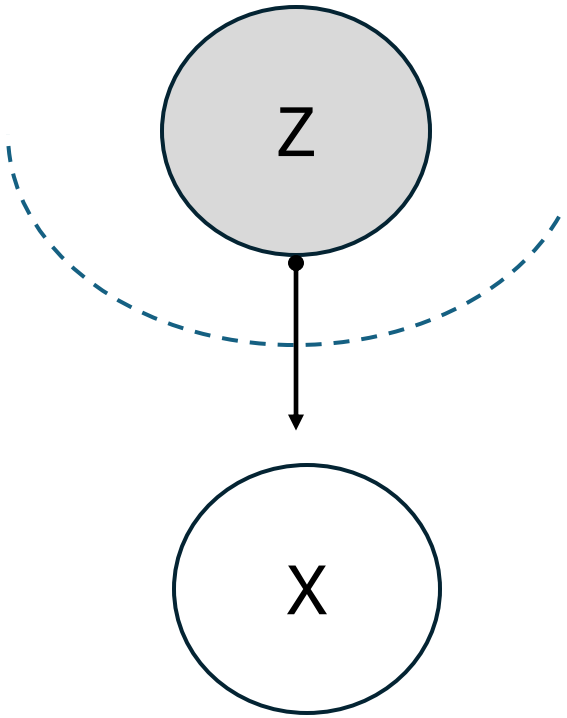
Variational **A**uto**E**ncoder (VAE)

Instructor: Diana Kim

Outline

1. Posterior probability as a compressed representation of data
2. Intractability of posterior Probability
3. Variational Methods
4. Variational Autoencoder (VAE): Architecture
5. Variational Autoencoder (VAE): Loss
6. Autoencoder as a special case of VAE
7. VAE as a generative Modeling

In the last class,
(1) we adopt a hidden variable Z to learn the density to describe X.
(2) The latent value Z can be used as a code representing the data X.

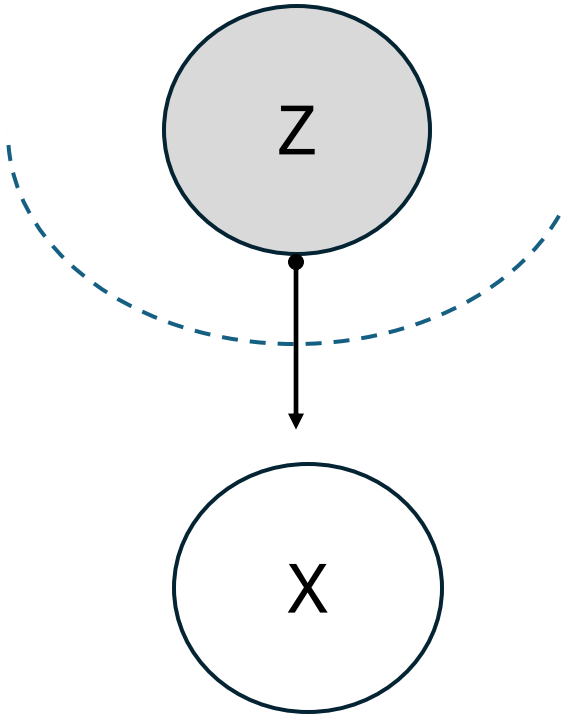


For example) K-mean Clustering

$$k* = \arg\max_k \gamma_{nk} = P(Z = k|x_n)$$

Data points are mapped to one of the K clustering.

The clustering assignment can be interpreted
as an **encoding process. Here, learning the posterior as learning encoder.**



$$k* = \arg\max_k \gamma_{nk} = P(Z = k|x_n)$$

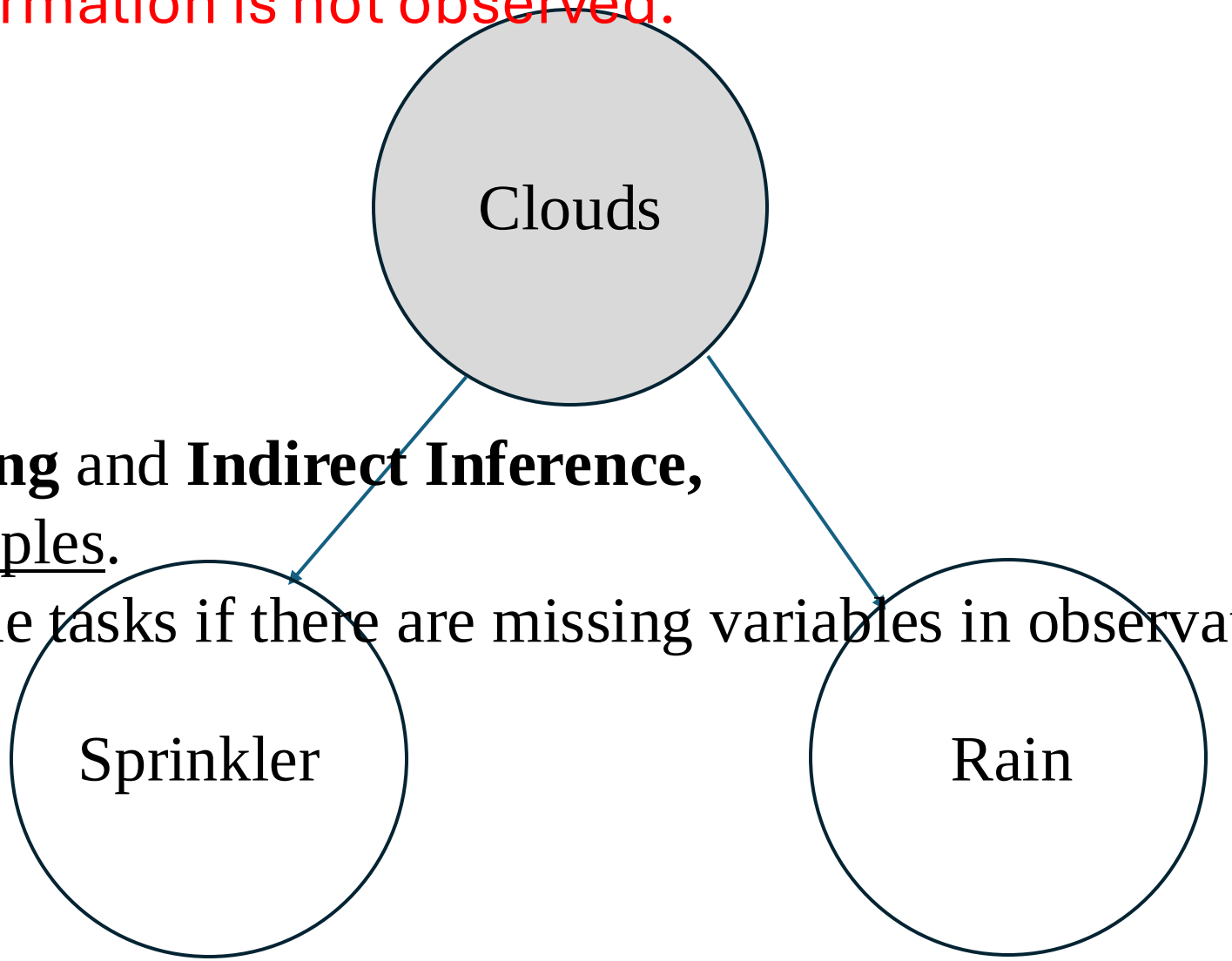
Q: Then, what will be the decoder?
if the graphical model represents GMM?

+ P(x| Z=k), this is is Gaussian in GMM.

(1) Learning the posterior in a latent causal modeling
provides the way to present data a compressed form.

(2) Also. as we learned in the last class,
The posterior enables us
to quantify/learn the probabilities we are interested in. “learning/ inference”

[Example in the last class]
when cloud information is not observed.



For **MLE Learning** and **Indirect Inference,**
We need data samples.
How can we do the tasks if there are missing variables in observation?

+ EM algorithm. First compute responsibility and compute weighted sample mean with responsibility.

[1] The Example of Learning the Parameters

- E step : compute $\gamma_{nk}(t) = P[Cloud(n) = k \mid Rain(n) \ and \ S(n)]$

- M step: update the parameters
  - $P(C)(t+1) = \frac{\sum_{n=1}^{N} \gamma_{nk}(t)}{N}$
  - $P(S:+, R:+|C:+)(t+1) = \frac{\sum_{n=1}^{N} \gamma_{nk} \delta(x_n = S:+, R:+)}{\sum_{n=1}^{N} \gamma_{nk}}$

[2] Example of Inference for Probability Queries

- compute $\gamma_{nk} = P[Cloud\ (n) = k \mid Rain\ (n)\ \ and\ S\ (n)\ ]$
- compute weighted sampling

$$P[C+|S+] = \frac{P[C+, S+]}{P[S+]}$$
$$= \frac{N_{C+S+}}{N_{S+}}$$
$$= \frac{\sum_n^N \gamma_{nk}\delta(x_n = S:+)}{N_{S+}}$$

This is the case of
indirect inference through sampling and counting,
but what if not all samples are observed?

+ EM algorithm. First compute responsibility and compute weighted sample mean with responsibility.

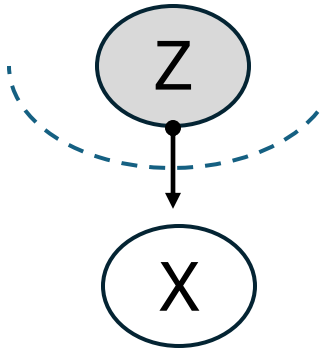
However,
It is often hard to define posterior density/ probability as a closed form.
or very complex. Especially the hidden variables are continuous.

It is often hard to compute responsibility.

Posterior distribution $P[Z_n|X_n]$ is often not well-defined.
(no closed / analytical form)

For example,

- $f(z) \sim \mathcal{N}(0,1)$
- $f(x|z) \sim \mathcal{N}(f(z),1)$
- $f(z|x) = \dfrac{f(z)f(x|z)}{\int f(z)f(x|z)dz} = \dfrac{f(z)f(x|z)}{\int \frac{1}{\sqrt{2\pi}}\exp -\frac{z^2}{2}\frac{1}{\sqrt{2\pi}}\exp -\frac{(x-f(z))^2}{2}dz}$

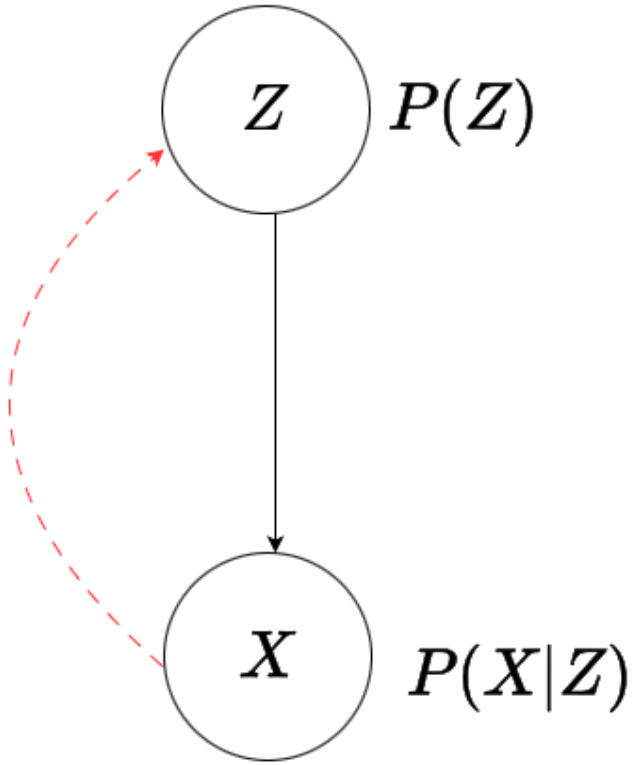


+ No closed form!

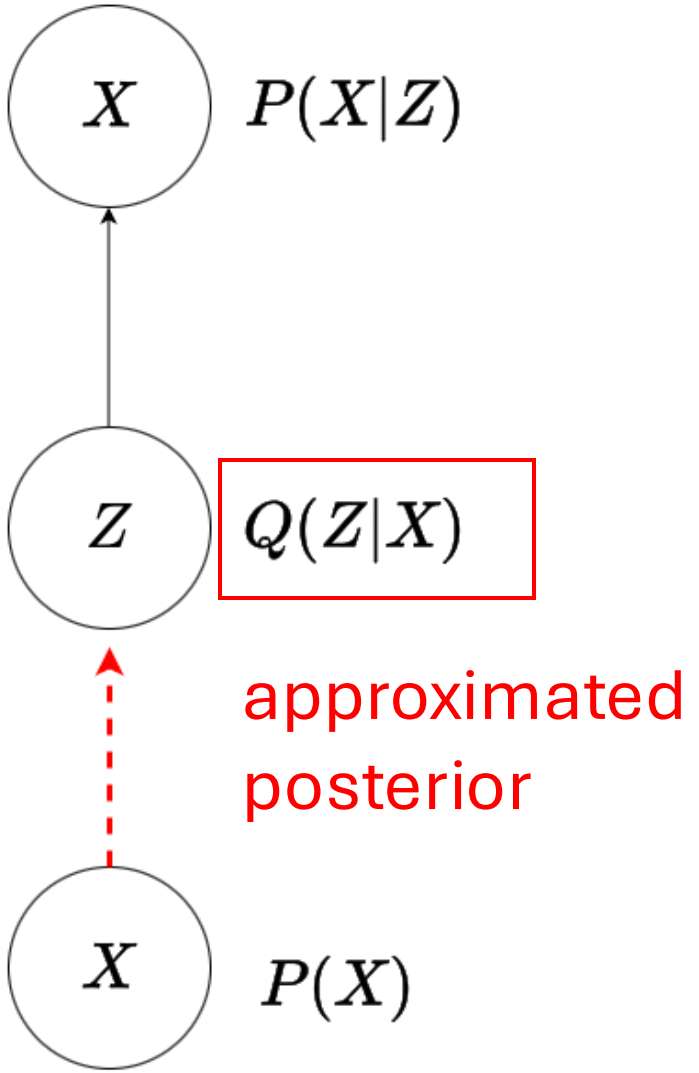
How can we handle the intractability of posterior density?

Use variational inference. Use posterior which is tractable.

Suppose a graphical model and forward direction CPTs are given as below.
And computing the posterior (reverse direction) is intractable.
How about using an inexact but tractable posterior rich enough to the original one.?



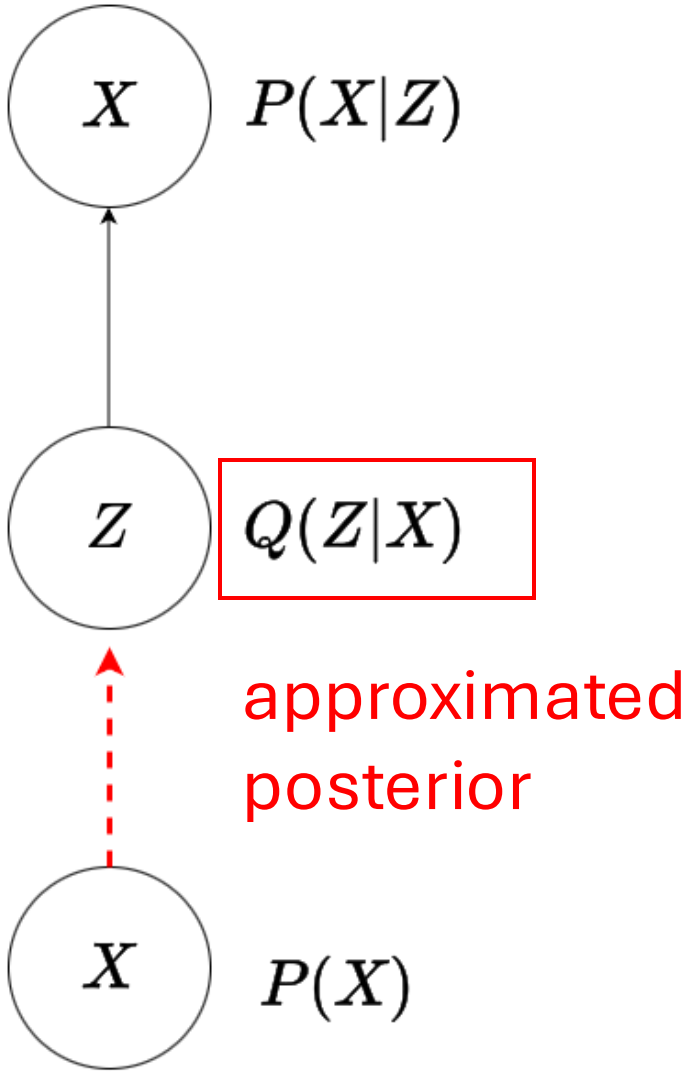
- Could we build a new graphical modeling including the approximated posterior?
- Could we learn the posterior $Q(Z|X)$ whiling learning $P(X)$?



- Feedforward network

- $Q(Z|X)$ became a part of process of reproducing X.

- $Q(Z|X)$ can be used to compute the lower bounds for Log P(X)



q(C_i |R_i,S_i θ_t) $\neq$ p (C_i |R_i,S_i θ_t) would not provide a tight bound.
But it will be tractable. We will choose a reasonable model
that is sufficiently rich, providing a good bound. (reflecting observation x well)

$$\log P(D|\theta) = \sum_{n=1}^{N} \log \sum_{C_n} P(S_n, R_n, C_n|\theta)$$

$$= \sum_{n=1}^{N} \log \sum_{C_n} \frac{P(S_n, R_n, C_n|\theta) q(C_n)}{q(C_n)}$$

$$= \sum_{n=1}^{N} \log E[\frac{P(S_n, R_n, C_n|\theta)}{q(C_n)}]$$

$$\geq \sum_{n=1}^{N} E[\log(\frac{P(S_n, R_n, C_n|\theta)}{q(C_n)})]$$

$$\geq \sum_{n=1}^{N} E[\log P(S_n, R_n, C_n|\theta)] + H(q(C_n))$$

Any arbitrary q probability
Provides a lower bound.

How can we learn an approximated posterior Q(Z|X)?

Using VAE

VAE (**V**ariational **A**uto-**E**ncoder)

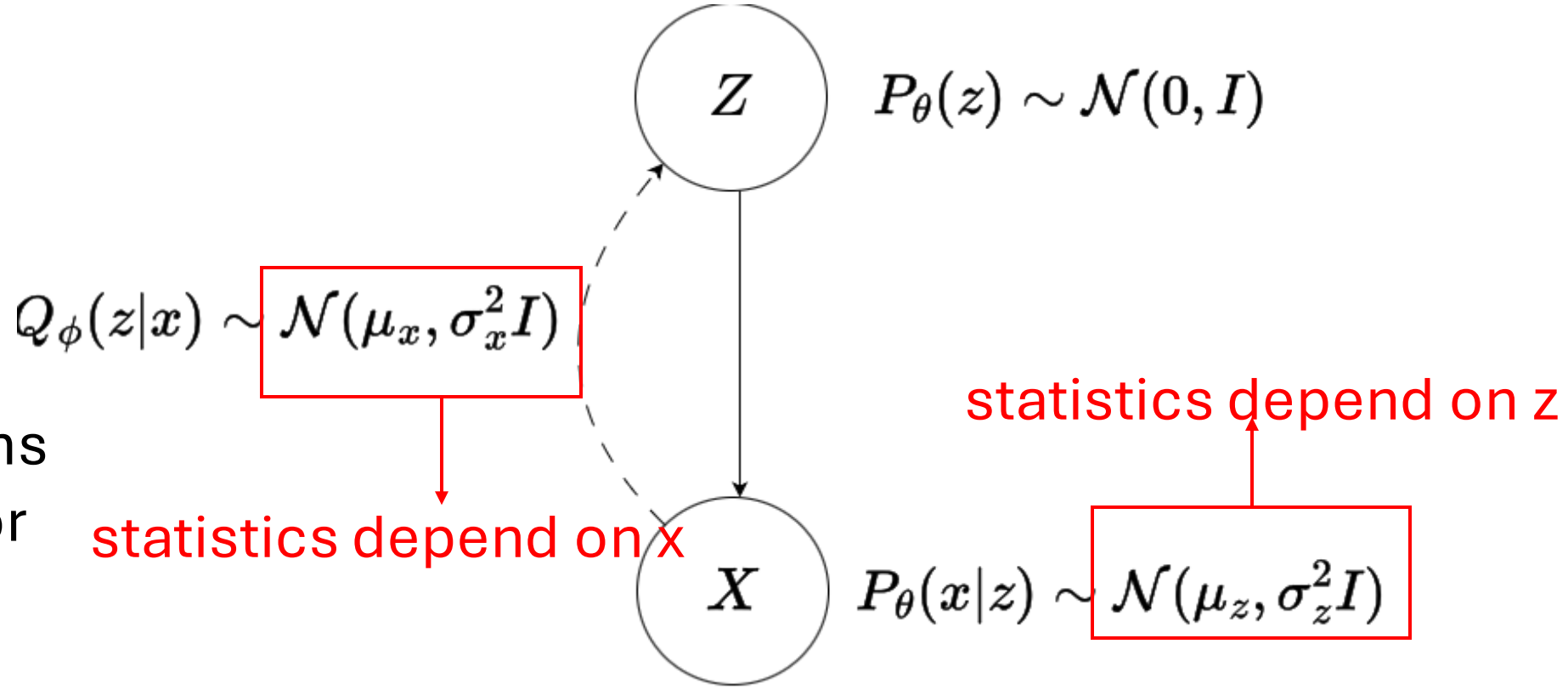
indirectly learn the approximated $Q(Z|X,\theta')$

while learning $P(X)$ from the observations $x_1, x_2, x_3, \ldots\ x_N$.

Bayesian Network for VAE

VAE assumes three probabilities.

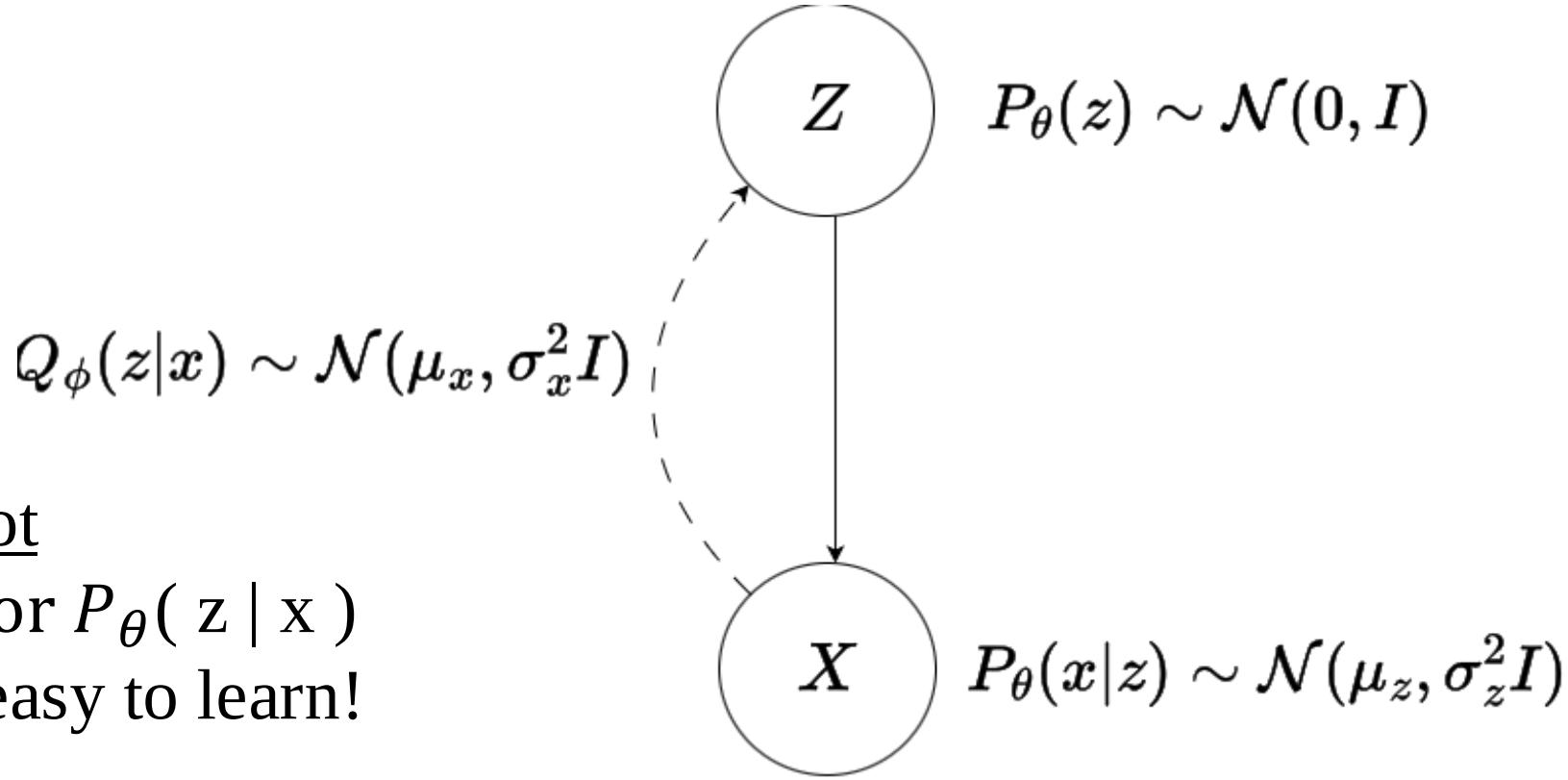
(1) assumptions
about the density of hidden space



(3) assumptions
about posterior

(2) assumptions
about the prob of generative part

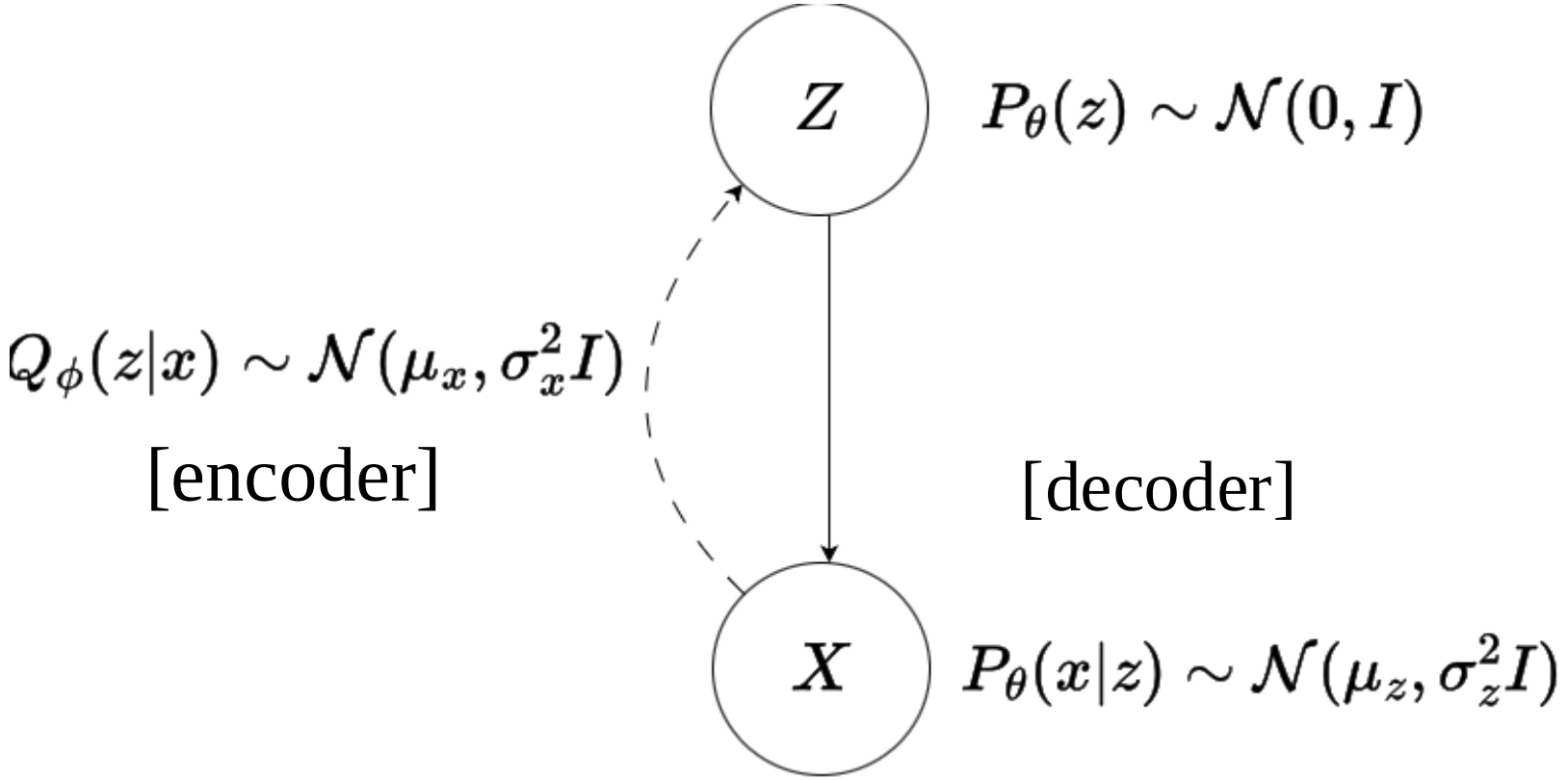
Bayesian Network for VAE
Variational Methods!



+ the posterior <u>is not</u>
  an exact posterior $P_\theta(\ z \mid x\ )$
  but this make it easy to learn!

+ this is the reason for why
  it is called variational method / variational inference.

The Bayesian Network for VAE



VAE encodes the Gaussian parameters (ϕ, θ) using a feed forward neural net and learn them together toward the direction maximizing $\prod_{n=1}^{N} P_\theta\ (x_n)$

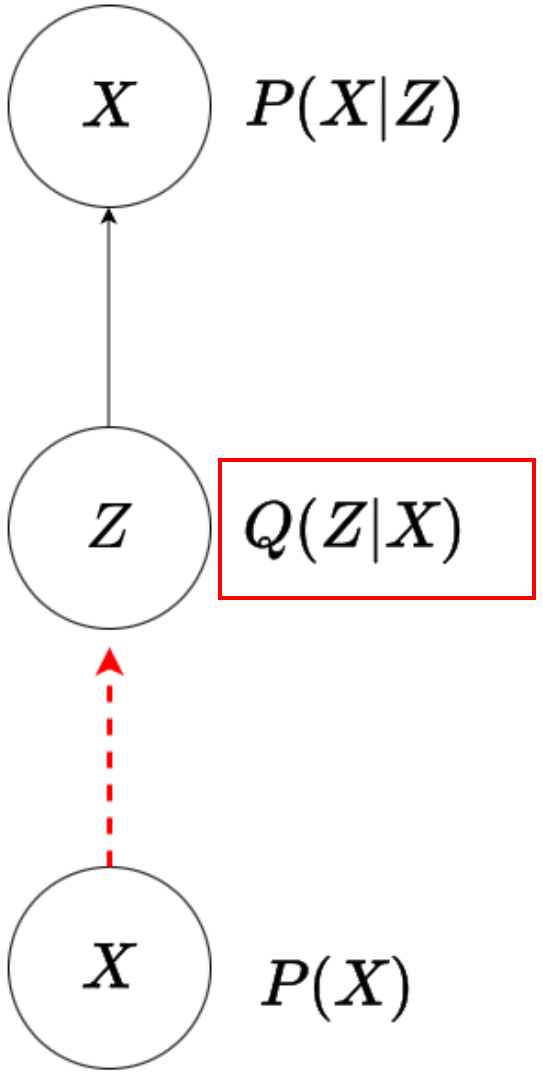
VAE is a frameworks that

- Defines a unified architecture for forward (decoder) and backward (encoder) operations.

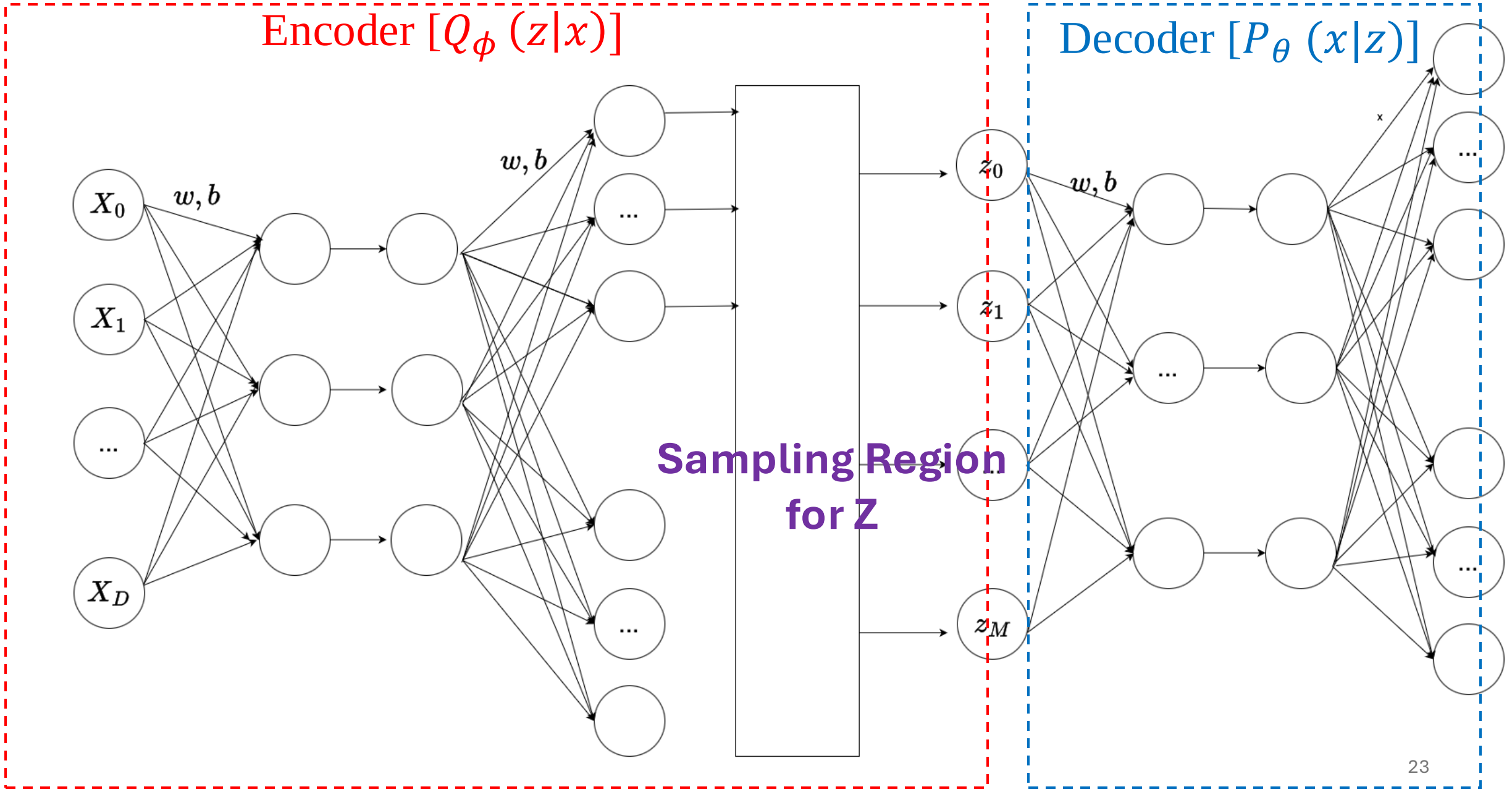
- Defines loss functions based on the lower bounds.

- Provides reparameterization trick for end-to-end optimization through backpropagation.

VAE Architecture



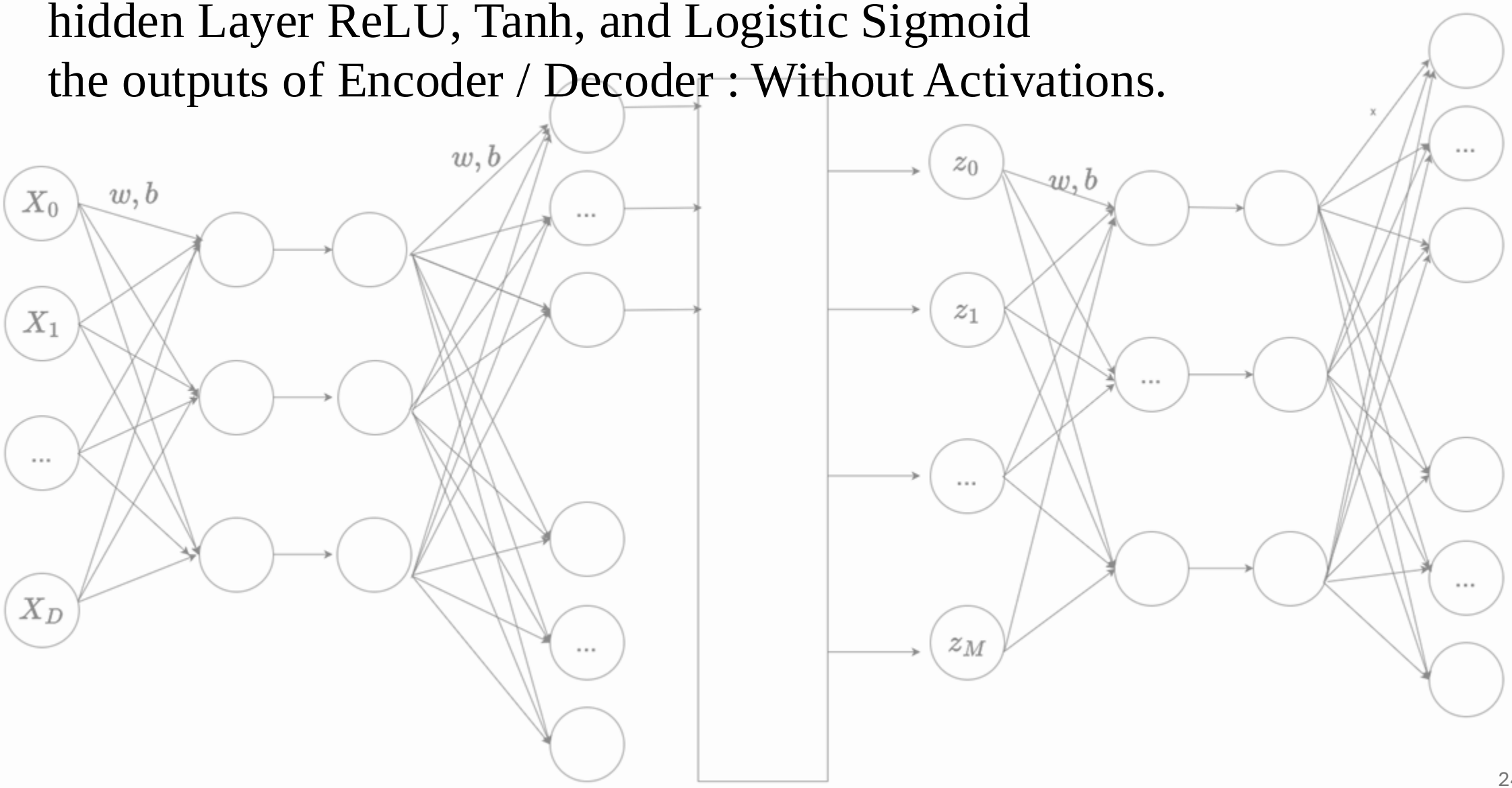
VAE Architecture [Encoder – Z – Decoder]



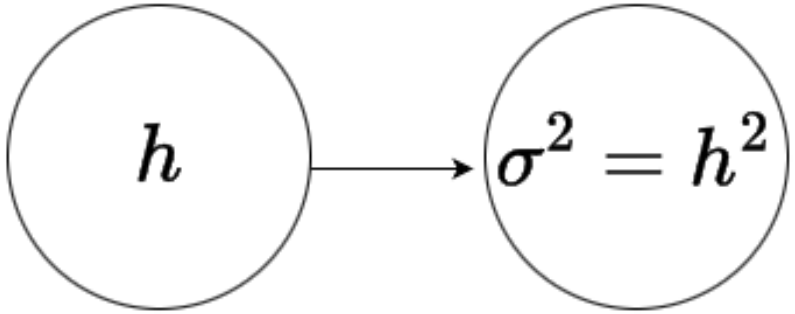
VAE Architecture [Multiple Layer MLP + Activations]

Activation functions:

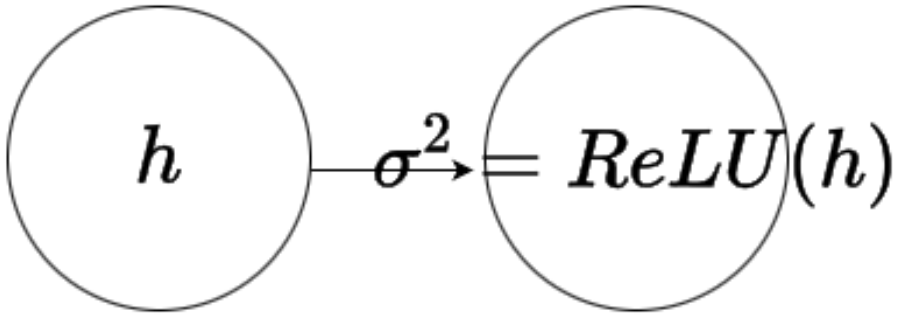
- hidden Layer ReLU, Tanh, and Logistic Sigmoid
- the outputs of Encoder / Decoder : Without Activations.



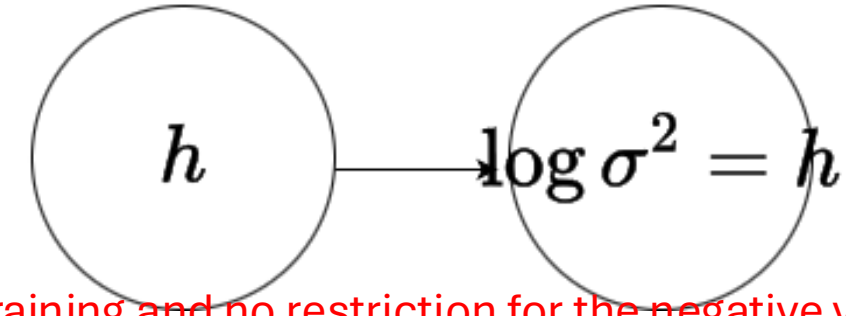
Possible Implementation for σ^2
but we generally use …



+ possibility of gradient explosion for the square term.



+ for the negative values becomes zero



+ stable for training and no restriction for the negative values.

VAE Loss (ϕ: $encoder$, θ: $decoder$)

$$\log p(x_i|\theta) \geq \underbrace{E_{Q(Z|x_i,\phi)}[\log P(x_i,z)] + H_{Q(Z|x_i,\phi)}(\log Q(Z|x_i,\phi))}_{\text{Function of } \phi \text{ and } \theta}$$

$$E_{Q(Z|x_i,\phi)}[\log P(x_i,z)] + H_{Q(Z|x_i,\phi)}(\log Q(Z|x_i,\phi))$$

$$\simeq 1/L \sum_{l=1}^{L} \{\log p(z_{i,l}|\theta) + \log p(x_i|z_{i,l},\theta) - \log Q(z_{i,l}|x_i,\phi)\}$$

[Monte Carlo Estimates of Expectations]

Monte Carlo Estimates of Expectation

$$E_{f(x)}[Q(x)] = \int Q(x) f(x) dx$$

$$\simeq 1/L \sum_{l=1}^{L} Q(x_i)$$

VAE Loss
defined by $Q_\phi(z|x)$, $P_\theta(x|z)$, $P_\theta(x) \sim N(0, I)$

- We know this lower bound:
  This is called "ELBO: Evidence Lower Bound"

$$\log p(x_i|\theta) \geq E_{Q(Z|x_i,\phi)}[\log P(x_i, z)] + H_{Q(Z|x_i,\phi)}(\log Q(Z|x_i, \phi))$$

ELBO

- We can compute the lower bound by using the multiple samples z given x_i in the middle of VAE. (z is a vector of dimension M)

collect L samples of z, given x_i

$$E_{Q(Z|x_i,\phi)}[\log P(x_i, z)] + H_{Q(Z|x_i,\phi)}(\log Q(Z|x_i, \phi))$$

$$\simeq 1/L \sum_{l=1} \{\log p(z_{i,l}|\theta) + \log p(x_i|z_{i,l}, \theta) - \log Q(z_{i,l}|x_i, \phi)\}$$

[1] VAE Loss: using data and hidden values

$$E_{Q(Z|x_i,\phi)}[\log P(x_i,z)] + H_{Q(Z|x_i,\phi)}(\log Q(Z|x_i,\phi))$$

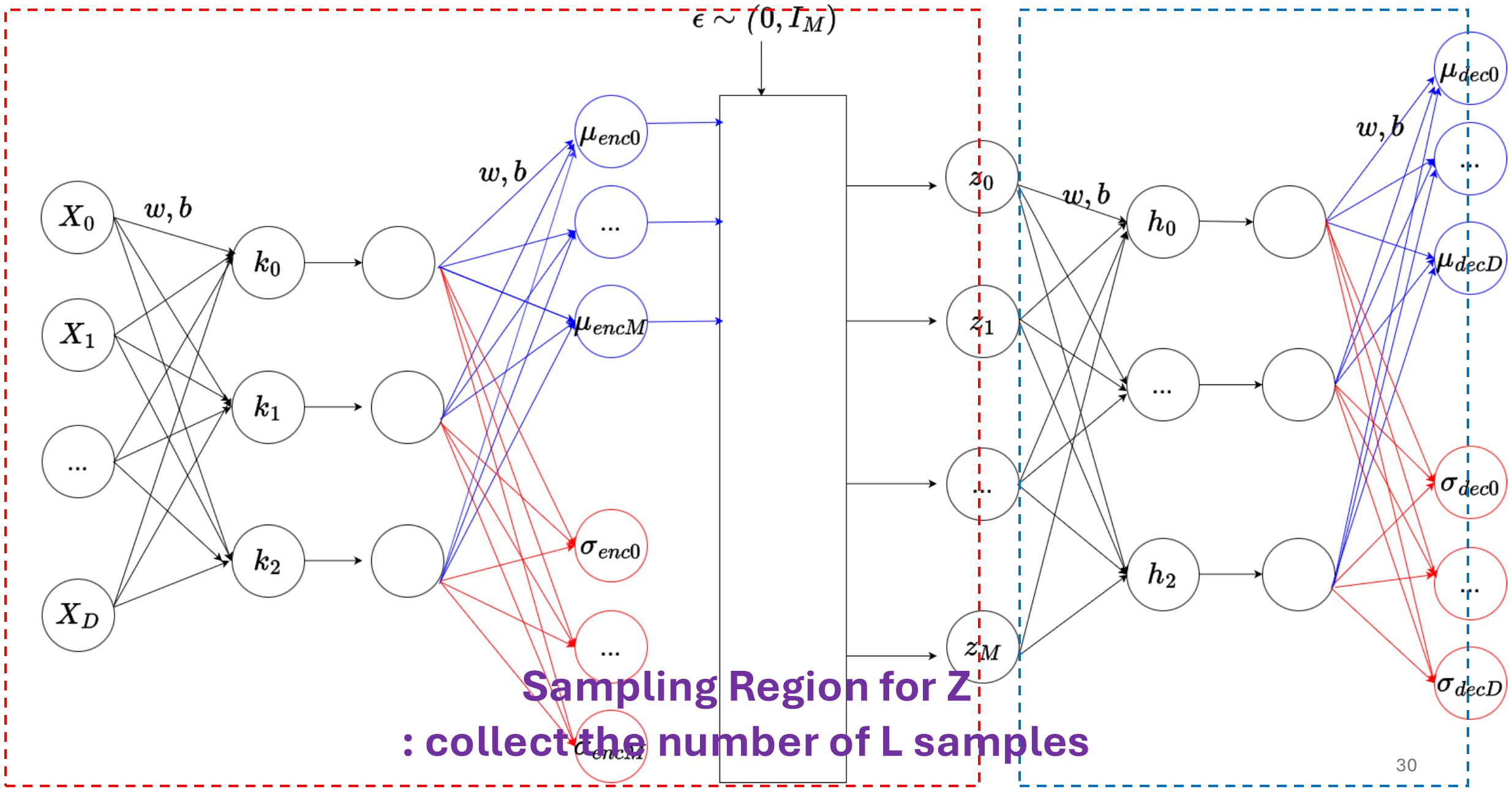
$$\simeq 1/L \sum_{l=1}^{L} \{\log p(z_{i,l}|\theta) + \log p(x_i|z_{i,l},\theta) - \log Q(z_{i,l}|x_i,\phi)\}$$

$$= 1/L \sum_{l=1}^{L} \{-1/2||z_{i,l}||^2 - \frac{1}{2\sigma_{dec}(i,l)}||x_i - \mu_{dec}(i,l)||^2 + \frac{1}{2\sigma_{enc}(i)}||z_{i,l} - \mu_{enc}(i)||^2\} + C$$

$$= 1/L \sum_{l=1}^{L} \{-||z_{i,l}||^2 - \frac{1}{\sigma_{dec}(i,l)}||x_i - \mu_{dec}(i,l)||^2 + \frac{1}{\sigma_{enc}(i)}||z_{i,l} - \mu_{enc}(i)||^2\}$$

1

VAE Architecture (Sampling Process: $x_n \rightarrow \#\ of\ L\ \ samples\ of\ Z$)



Sampling Region for Z

: collect the number of L samples

[2] VAE Loss (General Form)

defined by $Q_\phi(z|x)$, $P_\theta(x|z), P_\theta(x) \sim N(0, I)$

$$E_{Q(Z|x_i,\phi)}[\log P(x_i, z)] + H_{Q(Z|x_i,\phi)}(\log Q(Z|x_i, \phi))$$

$$\simeq 1/L \sum_{l=1}^{L} \{-\lambda||z||^2 - (x_i - \mu_{dec}(i,l))^t \Sigma^{-1}_{dec(i,l)} (x_i - \mu_{dec}(i,l)) + (z_{i,l} - \mu_{enc}(i))^t \Sigma^{-1}_{end(i)} (z_{i,l} - \mu_{enc}(i))\}$$

+ depending on prior assumption, we can have different lambda values. This can be interpreted as regularization.

$$\Sigma_{enc(i)} = \begin{bmatrix} \sigma_0(i) & \dots & 0 & 0 \\ 0 & \sigma_1(i) & \dots & 0 \\ 0 & 0\dots & \dots & \sigma_M(i) \end{bmatrix}$$

$$\Sigma_{dec(i,l)} = \begin{bmatrix} \sigma_0(i,l) & \dots & 0 & 0 \\ 0 & \sigma_1(i,l) & \dots & 0 \\ 0 & 0\dots & \dots & \sigma_D(i,l) \end{bmatrix}$$

+ Covariance matrix is diagonal.
it can be isotropic
for the case of single unit variance.

[2] VAE Loss

(For Gaussian Assumption, an analytical solution is also possible)

$$E_{Q(Z|x_i,\phi)}[\log P(x_i,z)] + H_{Q(Z|x_i,\phi)}(\log Q(Z|x_i,\phi))$$
$$= -KL(Q(z|x_i)||P_\theta(z)) + E_{Q(Z|x_i,\phi)}[\log p(x_i|Z,\theta)]$$
$$\simeq 1/2 \sum_{m=1}^{M}(1 + \log(\sigma_{enc}(m|i))^2 - \mu_{enc}(m|i)^2 - \sigma_{enc}(m|i)^2) - 1/L \sum_{l=1}^{L} \frac{1}{\sigma_{dec}(i,l)}||x_i - \mu_{dec}(i,l)||^2$$

Check Reference : Appendix B!
https://arxiv.org/abs/1312.6114

$$\int Q(Z|x_i,\phi)\log P(Z)dz = \int Q(Z|x_i,\phi)\log 1/\sqrt{(2\pi)^M}\exp -1/2||z||^2$$
$$= -M/2\log(2\pi) - 1/2\sum_{m=1}^{M} E_{Q(Z|x_i,\phi)}E[z_m^2]$$
$$= -M/2\log(2\pi) - 1/2\sum_{m=1}^{M}(\mu_{enc}(m|i)^2 + \sigma_{enc}(m|i)^2)$$

VAE uses a trick
that integrates the sampling process into
end-to-end learning, enabling backpropagation.

Reparameterization Trick:

Generates Z through the deterministic block $g_\phi(\epsilon, x_i)$ where $\epsilon \sim N(0, I)$.

$$z = \Sigma^{1/2}\epsilon + \mu$$

$$Z = \begin{bmatrix} \sigma_{enc}(0|i) & \dots & 0 & 0 \\ 0 & \sigma_{enc}(1|i) & \dots & 0 \\ 0 & 0\dots & \dots & \sigma_{enc}(M|i) \end{bmatrix} \epsilon + \begin{bmatrix} \mu_{enc}(0|i) \\ \mu_{enc}(1|i) \\ \dots \\ \mu_{enc}(M|i) \end{bmatrix}$$

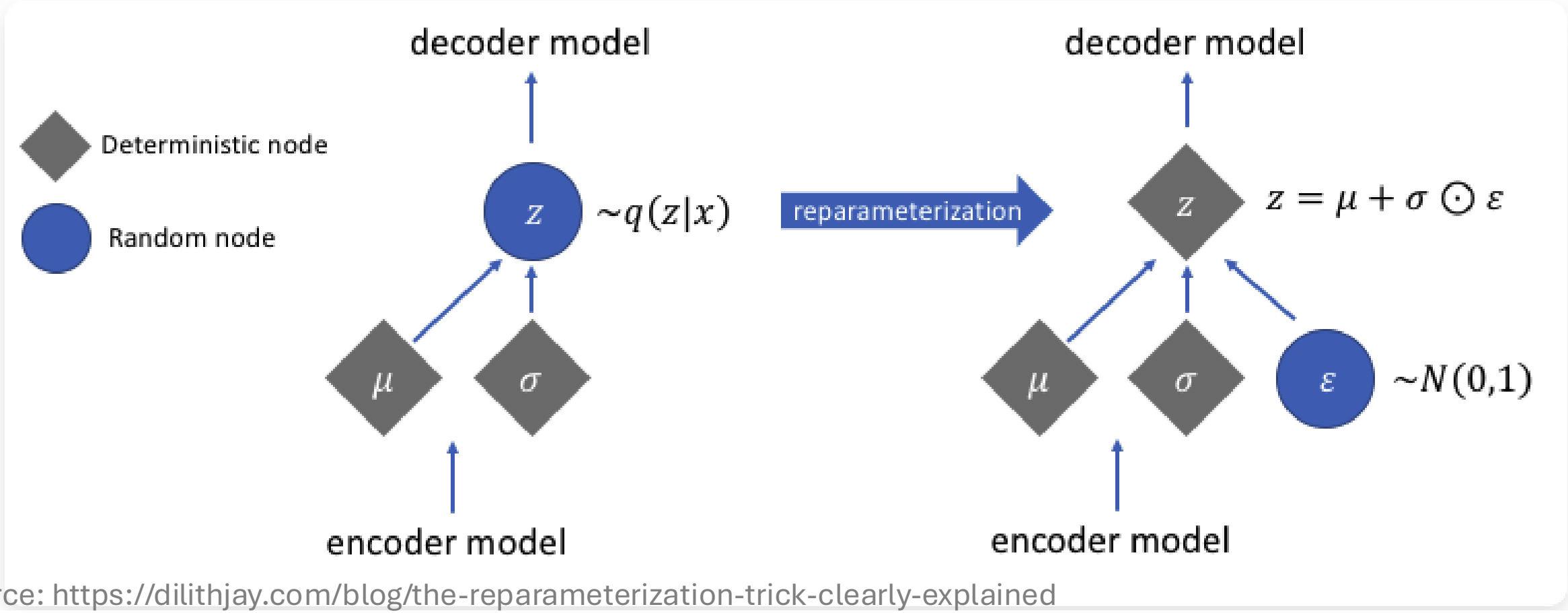
Backpropagation cannot pass through random block!

Reparameterization:

(1) uses external random source (no need to compute gradient) and

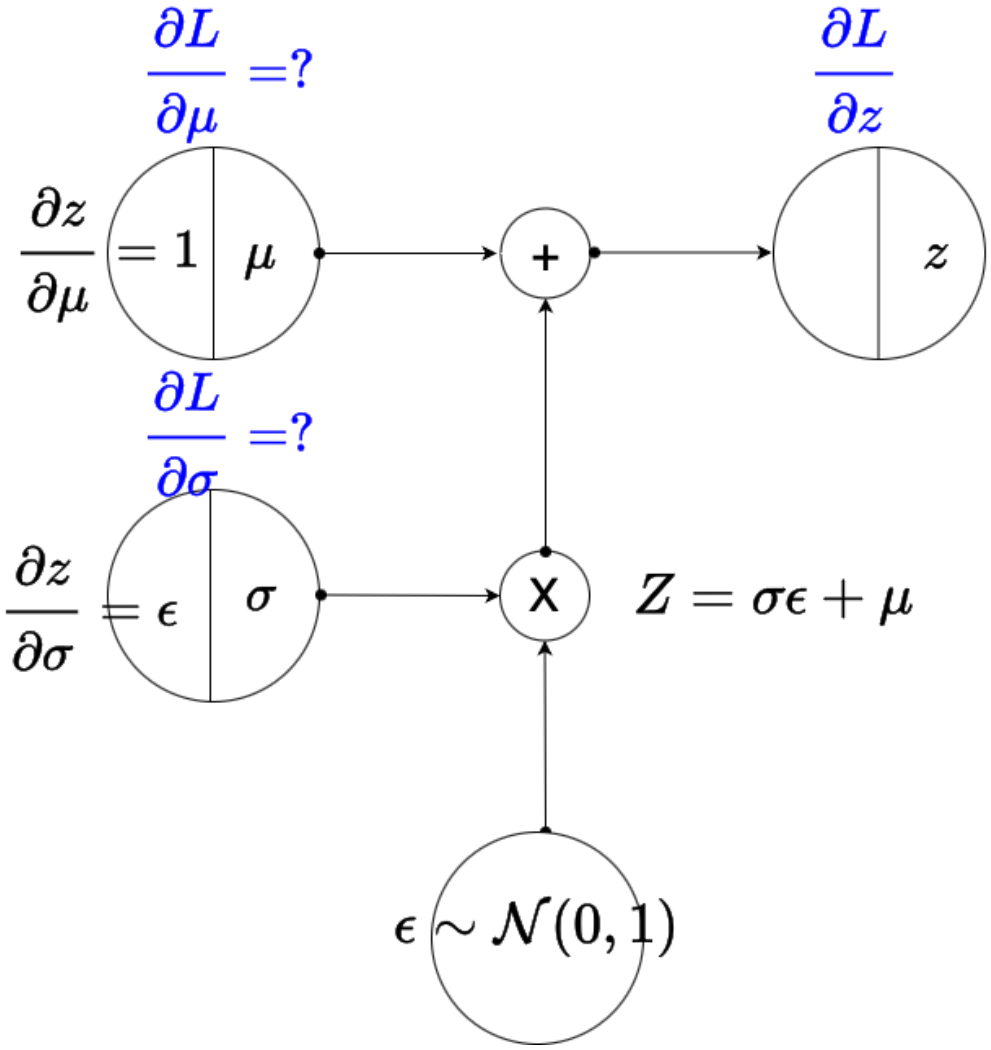
(2) defines a deterministic function that generates z

having same statistical properties.



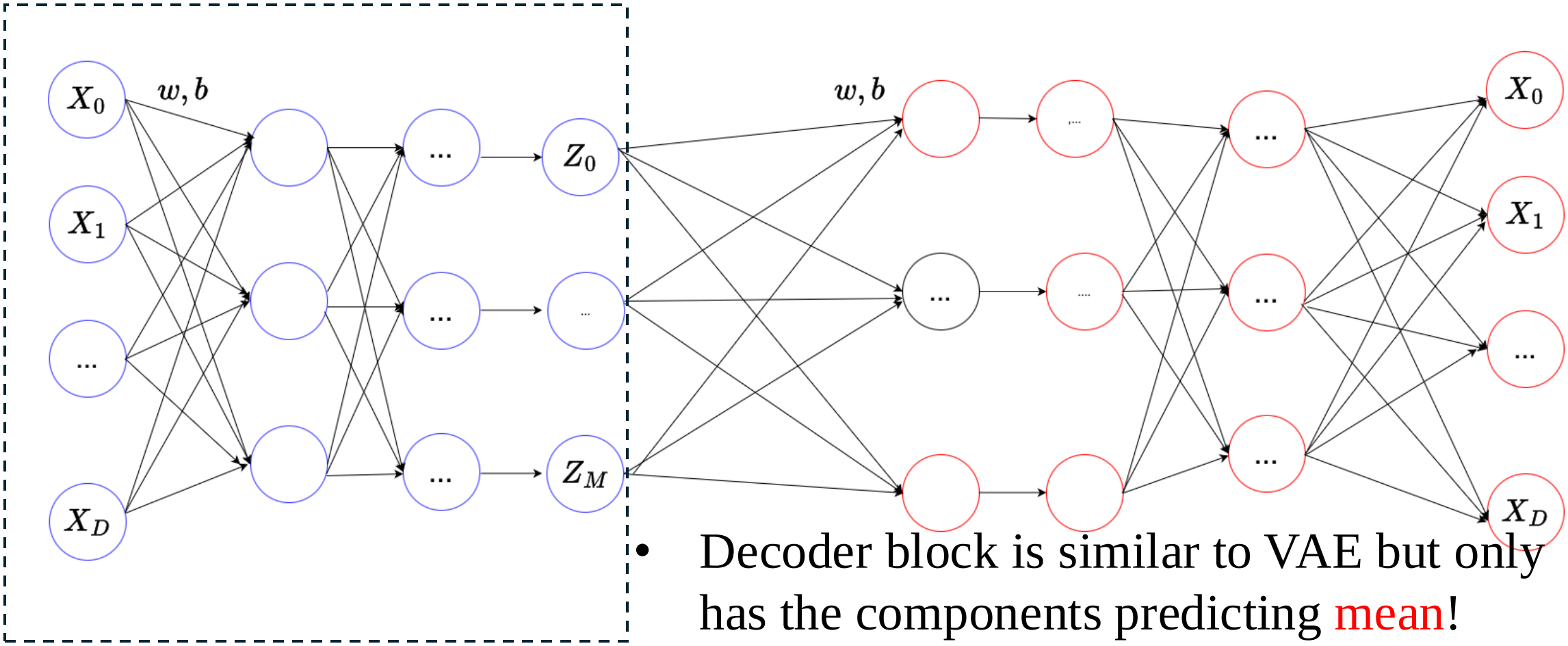
Source: https://dilithjay.com/blog/the-reparameterization-trick-clearly-explained

Reparameterization Trick makes the sampling process
Integrated as a part of end-to-end learning by backpropagation.



Connection of VAE to Autoencoder

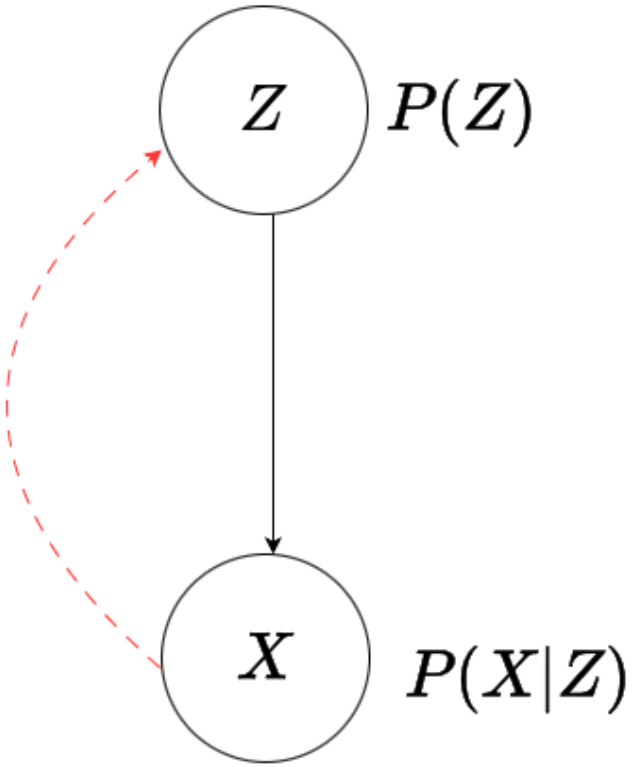
Autoencoder Architecture:
No probabilistic Encoder
Deterministic Mapping between X and Z



- Decoder block is similar to VAE but only has the components predicting mean!

$$Q(z|x_i) = f(x)\delta(x - x_i)$$

+ deterministic mapping



Autoencoder Loss

$$E_{Q(Z|x,\phi)}[P(x,z)] + H_{Q(Z|x,\phi)}(Q(Z|x,\phi))$$

$$\simeq 1/N \sum_{n=1}^{N} \{\log p(z_n|\theta) + \log p(x_n|z_n,\theta) - \log Q(z_n|x_n,\phi)\}$$

= ?

+ No probabilistic relation between x and z for Auto Encoder, so this term would not be considered to compute the loss.

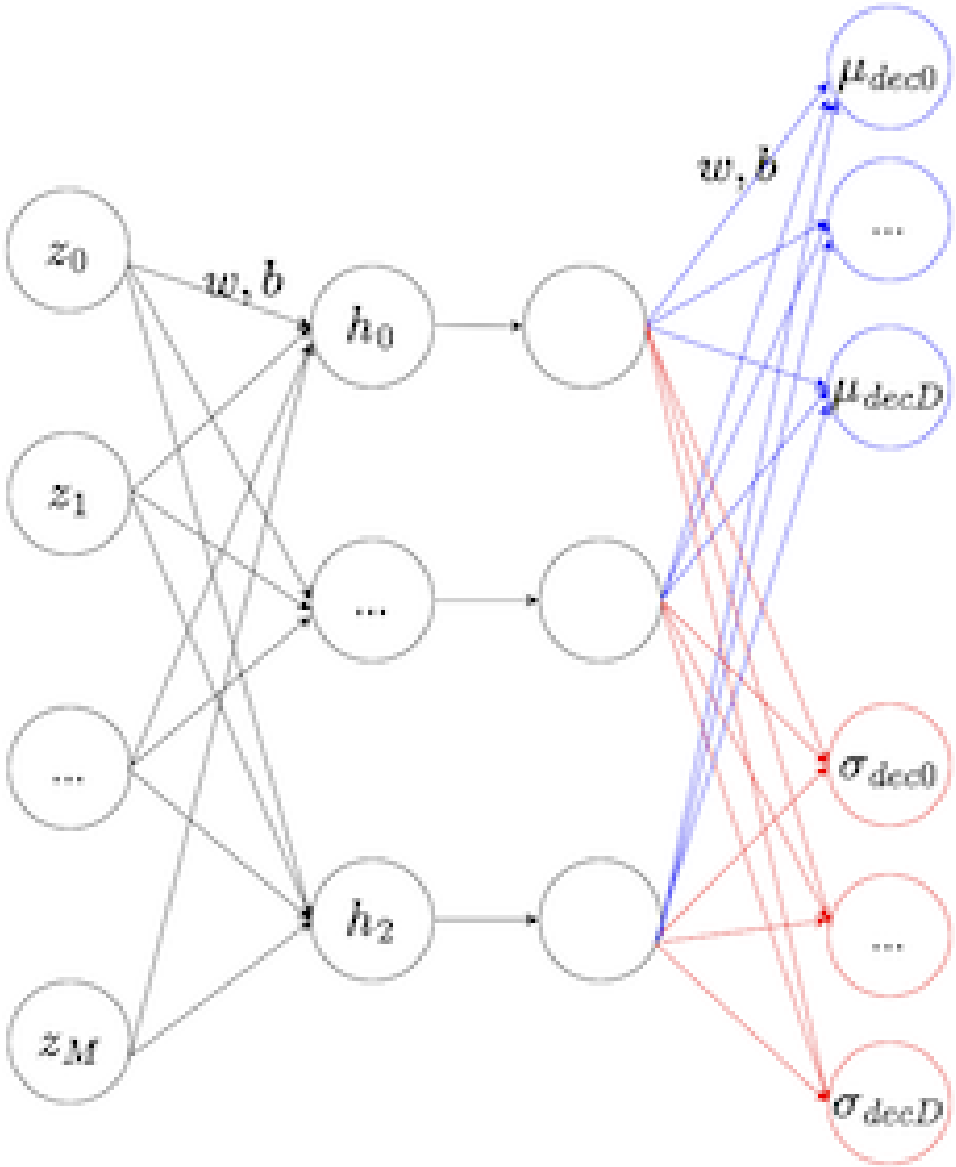
VAE and Autoencoder both learns low dimensional manifolds of data.
A key difference between the two systems is that
VAE provides a probabilistic hidden space (continuous),
so we can use VAE decoder as a generative model.

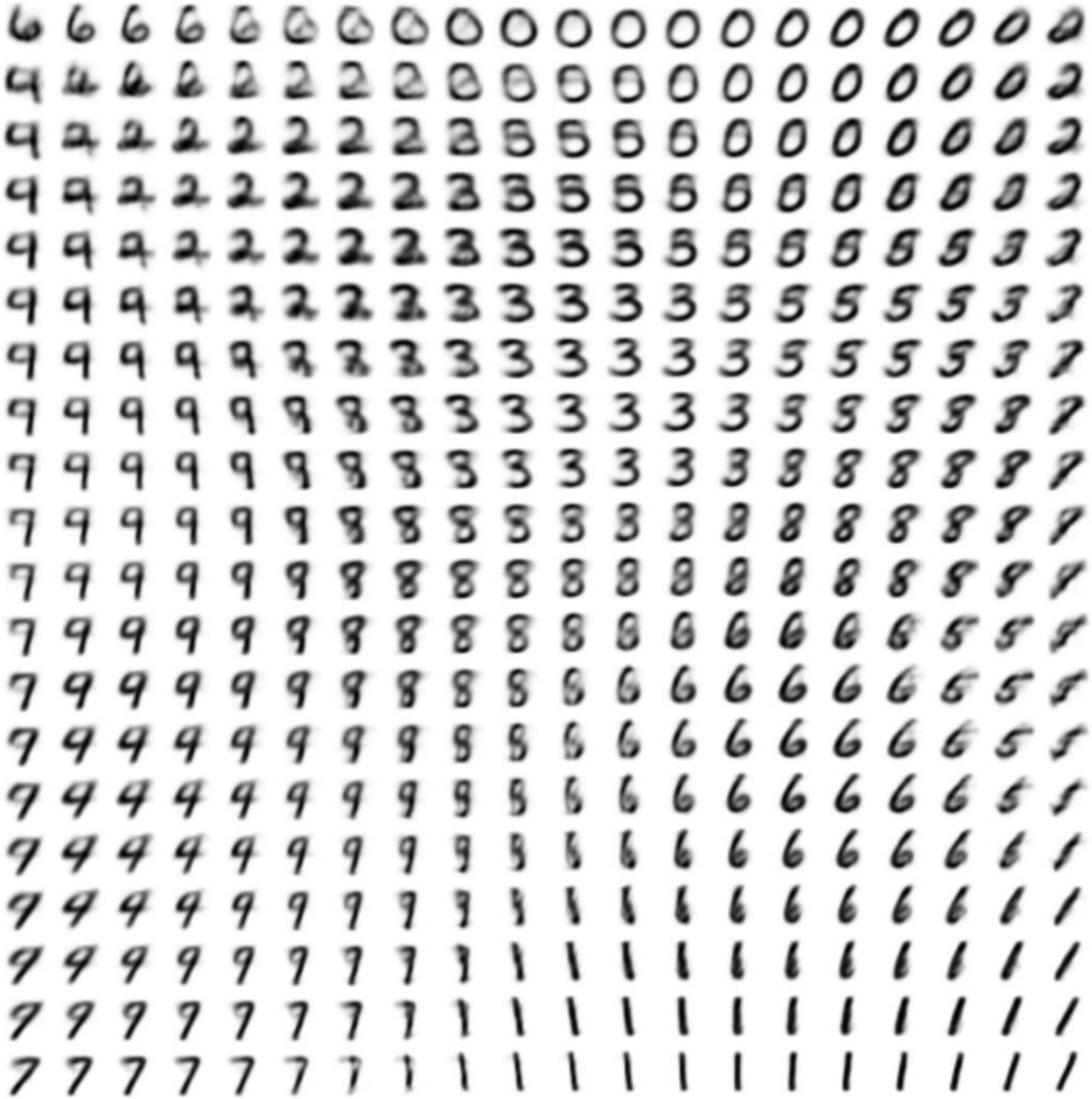
Decoder as a Generative Model.

+ take the value of z

+ compute μ_{dec} and Σ_{dec}

+ sample $X \sim N(\mu_{dec}, \Sigma_{dec})$





All visual variations is smoothly (continuously) encoded into 2D space.

Learned 2D Manifold MNIST

Fig 4 https://arxiv.org/abs/1312.6114

Inverse CDF Technique

Suppose $F\ (x) =\ P[X \leq x]$ (CDF)

If U is a uniform R.V on [0,1] then X = $F^{-1}\ (x)$ follows the distribution $\frac{dF}{dx}$

Intensity of Facial Expression

Degree of Smiling

2D disentanglement of manifold representations

Fig4: https://arxiv.org/abs/1312.6114