Lecture 6: Unsupervised Representation Learning and Generative Models

Shujian Yu

Deep Learning 2023

dlvu.github.io

VU VRIJE UNIVERSITEIT AMSTERDAM

THE PLAN

part 1: Why generative modeling and unsupervised learning

part 2: Autoencoders

part 3: Variational autoencoders

PART ONE: WHY GENERATIVE MODELING AND UNSUPERVISED LEARNING

IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

Example from: https://adversarial-ml-tutorial.org/introduction/

IS GENERATIVE MODELING IMPORTANT?

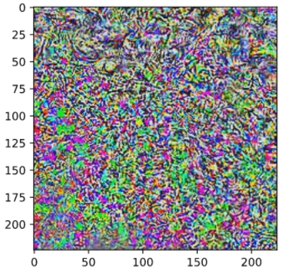
We learn a neural network to classify images:

p(**hog**|x)=0.99

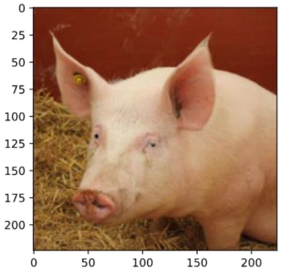
...

We learn a neural network to classify images:

+



=



p(**hog**|x)=0.99

...

noise

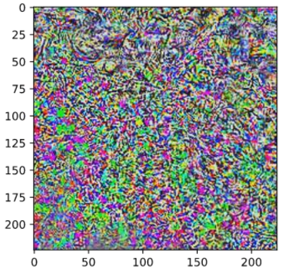
IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

p(**hog**|x)=0.99

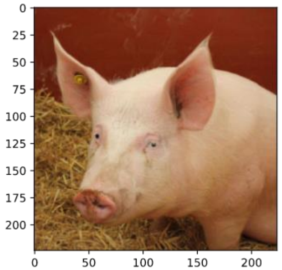
...

+



noise

=



p(**hog**|x)=0.01

…

p(**airliner**|x)=0.97

There is no semantic understanding of images.

IS GENERATIVE MODELING IMPORTANT?

This simple example shows that:

- A discriminative model is (probably) **not enough**.
- We need a notion of **uncertainty**.
- We need to **understand** the reality.

IS GENERATIVE MODELING IMPORTANT?

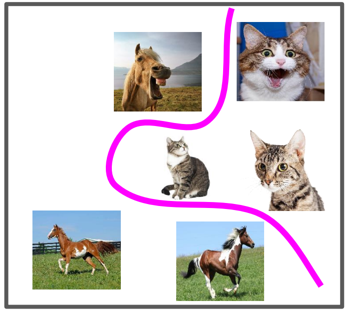
This simple example shows that:

- A discriminative model is (probably) **not enough**.
- We need a notion of **uncertainty**.
- We need to **understand** the reality.

A possible solution is **generative modeling**.

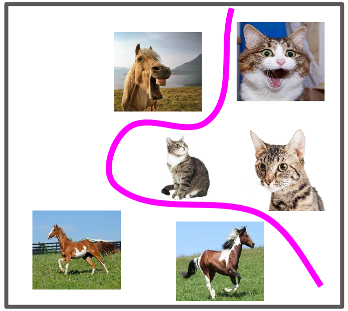
IS GENERATIVE MODELING IMPORTANT?

IS GENERATIVE MODELING IMPORTANT?

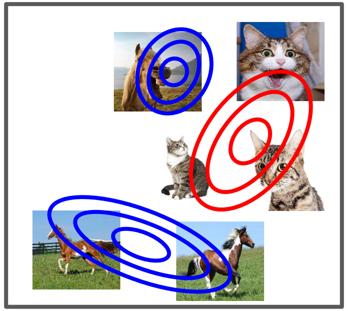


$$p_\theta(y|x)$$

IS GENERATIVE MODELING IMPORTANT?

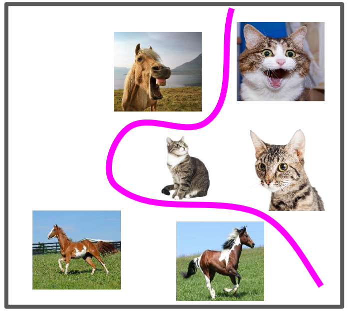


$$p_\theta(y|x)$$

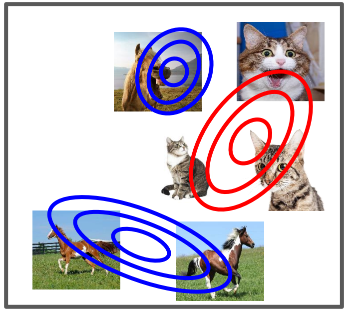


$$p_\theta(x,y) = p_\theta(y|x)\ p_\theta(x)$$

IS GENERATIVE MODELING IMPORTANT?



$$p_\theta(y|x)$$

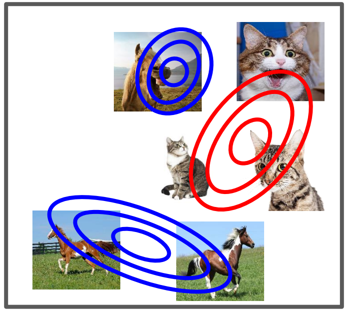


$$p_\theta(x,y) = p_\theta(y|x)\; p_\theta(x)$$

IS GENERATIVE MODELING IMPORTANT?

$$p_\theta(y|x)$$

High probability
of a **horse**.
=
Highly probable decision!



$$p_\theta(x, y) = p_\theta(y|x)\ p_\theta(x)$$

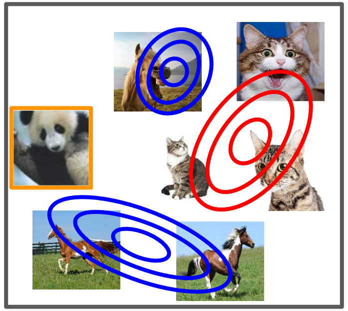
IS GENERATIVE MODELING IMPORTANT?

$$p_\theta(y|x)$$

High probability of a **horse**.

=

Highly probable decision!



$$p_\theta(x, y) = p_\theta(y|x)\ p_\theta(x)$$

High probability of a **horse**.

x

Low probability of the **object**

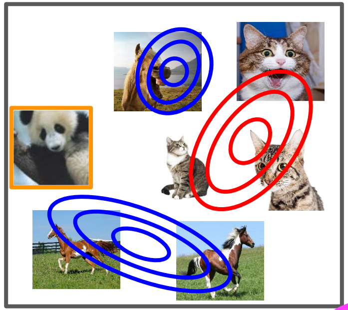
=

Uncertain decision!

IS GENERATIVE MODELING IMPORTANT?

$$p_\theta(y|x)$$

High probability of a **horse**.
=
Highly probable decision!



$$p_\theta(x, y) = p_\theta(y|x)\, p_\theta(x)$$

!

High probability of a **horse**.
x
Low probability of the **object**
=
Uncertain decision!

WHERE DO WE USE DEEP GENERATIVE MODELING?

Generate images

Child, Rewon. "Very deep vaes generalize autoregressive models and can outperform them on images." *arXiv preprint arXiv:2011.10650* (2020). https://github.com/openai/vdvae

WHERE DO WE USE DEEP GENERATIVE MODELING?

Generate audios

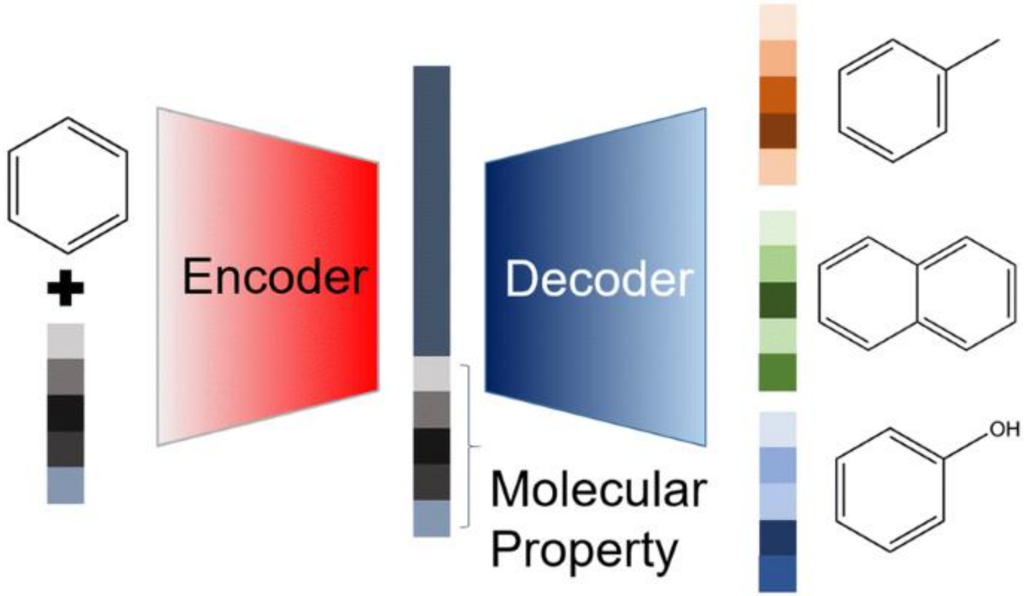
darbouka solo

speech

Caillon, Antoine, and Philippe Esling. "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis." *arXiv preprint arXiv:2111.05011* (2021). https://github.com/acids-ircam/RAVE

WHERE DO WE USE DEEP GENERATIVE MODELING?

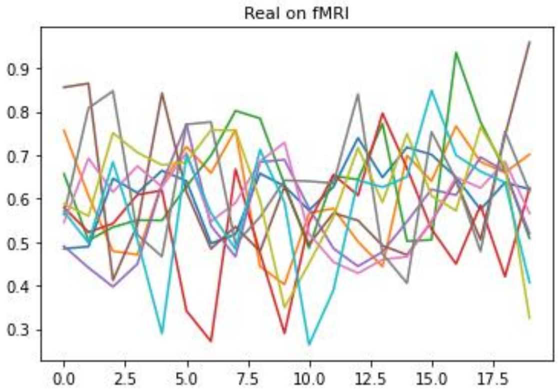
Generate molecules



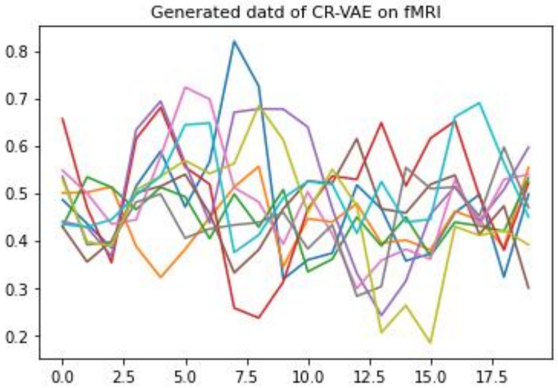
Lim, Jaechang, et al. "Molecular generative model based on conditional variational autoencoder for de novo molecular design." *Journal of cheminformatics* 10.1 (2018): 1-9. https://jcheminf.biomedcentral.com/articles/10.1186/s13321-018-0286-7

WHERE DO WE USE DEEP GENERATIVE MODELING?

Generate medical data (e.g., fMRI signals)



real fMRI signal

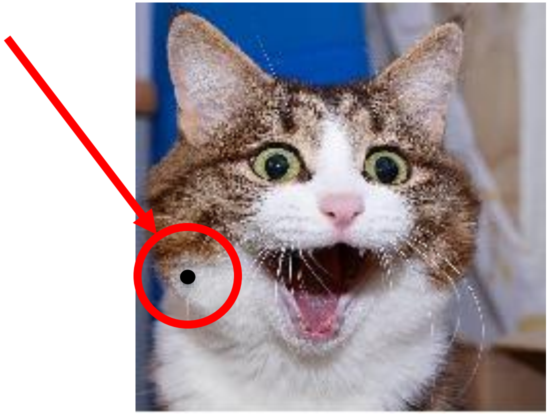


generated fMRI signal

Li, Hongming, Shujian Yu, and Jose Principe. "Causal recurrent variational autoencoder for medical time series generation." *arXiv preprint arXiv:2301.06574* (2023). https://github.com/hongmingli1995/CR-VAE

Modeling in high-dimensional spaces is difficult.

Modeling in high-dimensional spaces is difficult.



Modeling in high-dimensional spaces is difficult.

Modeling **all dependencies** among pixels:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c=1}^{C} \psi_c(\mathbf{x}_c)$$

Modeling in high-dimensional spaces is difficult.

Modeling **all dependencies** among pixels:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c=1}^{C} \psi_c(\mathbf{x}_c)$$

problematic

Modeling in high-dimensional spaces is difficult.

Modeling **all dependencies** among pixels:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c=1}^{C} \psi_c(\mathbf{x}_c)$$

problematic

A possible **solution**: **Latent Variable Models** *or* **Latent Representation Learning**!

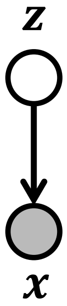
LATENT VARIATIONAL MODELS: DEFINITION

A latent variable model defines a probability distribution:

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

Containing two sets of variables:

- Observed variables x that represent the high-dimensional observation.
- Latent variable z that are not in the observation space, but that are *hidden* and associated with x via $p(z|x)$ and can encode the structure of the data.

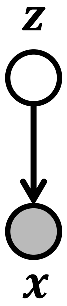


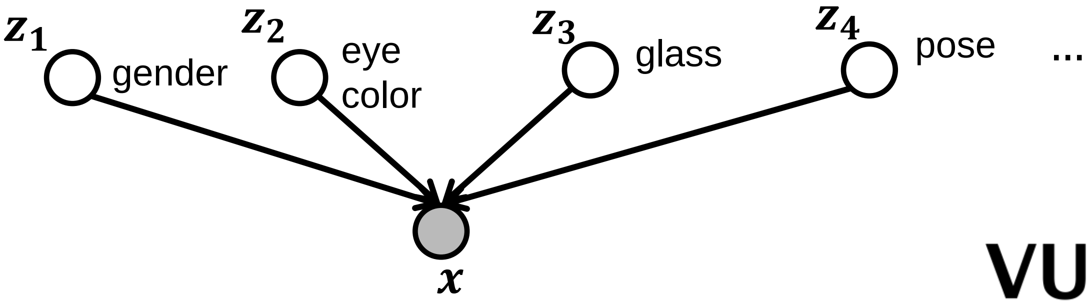
A latent variable model defines a probability distribution:

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

Containing two sets of variables:

- Observed variables x that represent the high-dimensional observation.
- Latent variable z that are not in the observation space, but that are *hidden* and associated with x via $p(z|x)$ and can encode the structure of the data.





PART TWO: AUTOENCODERS

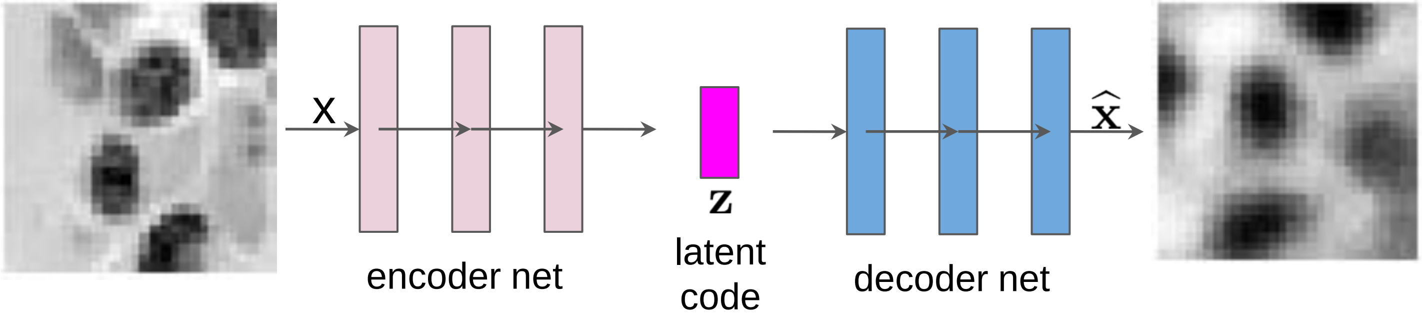
Learn a compressed representation of the input data x.

We have two functions (usually neural networks)

encoder: $\mathbf{z} = g_\phi(\mathbf{x})$ decoder: $\hat{\mathbf{x}} = f_\theta(\mathbf{z})$

Train using a reconstruction loss

$$L(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 = \left\|\mathbf{x} - f_\theta\left(g_\phi(\mathbf{x})\right)\right\|^2$$

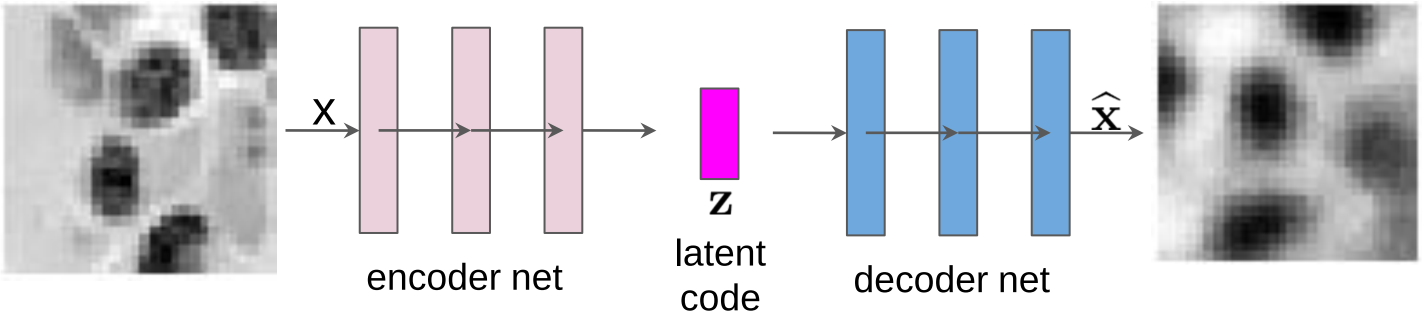


Learn a compressed representation of the input data x.

In case of linear encoder and decoder

encoder: $\mathbf{z} = W^T\mathbf{x}$ decoder: $\hat{\mathbf{x}} = W\mathbf{z}$

The minimum error solution W yields the same subspace as PCA



AUTOENCODER AND COMPRESSION

Examine the latent space of autoencoder.

- Plot the latent space and examine the separation
- First two PCA components of latent space

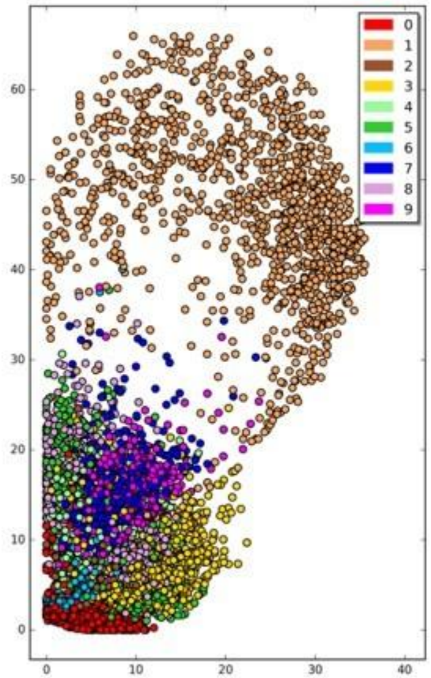


Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

Examine the latent space of autoencoder.

- We start at the start of the arrows in latent space and then move to end of the arrow in 7 steps
- For each value of $\mathbf{z}$, we use the already trained decoder to produce an image.

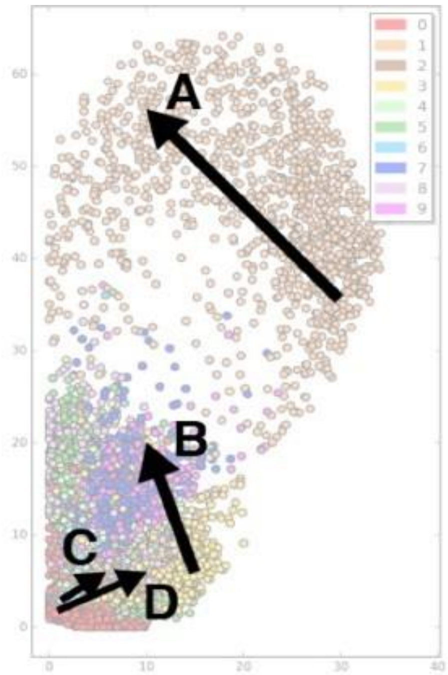
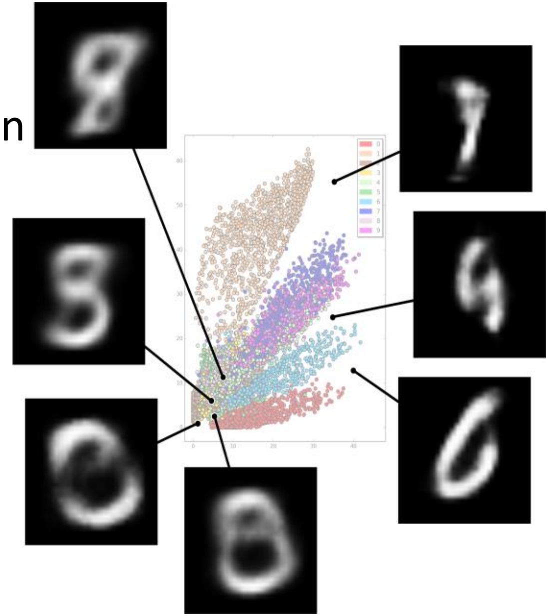


Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

Problems with naïve autoencoder.

- Not a generative model (only learn to reconstruct)
- Only “interesting” when $\mathbf{z}$ has much smaller dimension than $\mathbf{x}$
- Gaps in the latent space
- Separability in the latent space
- Difficulty in interpreting latent space

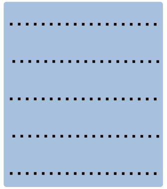


GENERATIVE MODELING WITH LATENT VARIABLES

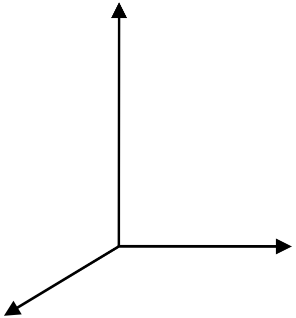
Generative process:

1. $\mathbf{z} \sim p_{\lambda}(\mathbf{z})$
2. $\mathbf{x} \sim p_{\theta}(\mathbf{x}|\mathbf{z})$

Z: 2D

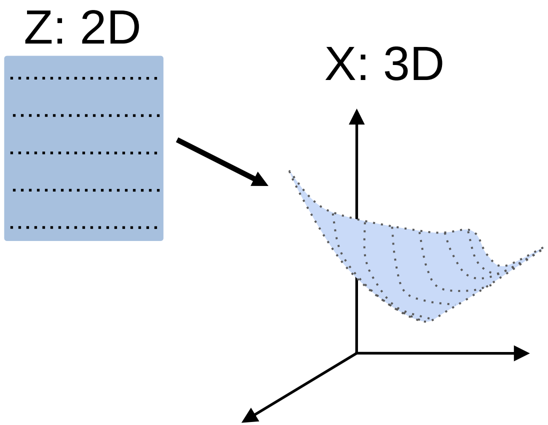


X: 3D



Generative process:

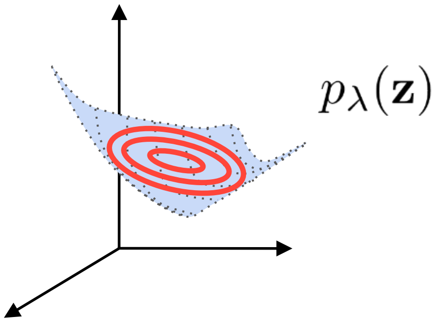
1. $\mathbf{z} \sim p_{\lambda}(\mathbf{z})$
2. $\mathbf{x} \sim p_{\theta}(\mathbf{x}|\mathbf{z})$



GENERATIVE MODELING WITH LATENT VARIABLES

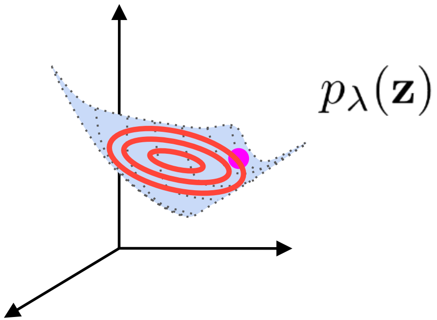
Generative process:

1. $\mathbf{z} \sim p_{\lambda}(\mathbf{z})$
2. $\mathbf{x} \sim p_{\theta}(\mathbf{x}|\mathbf{z})$



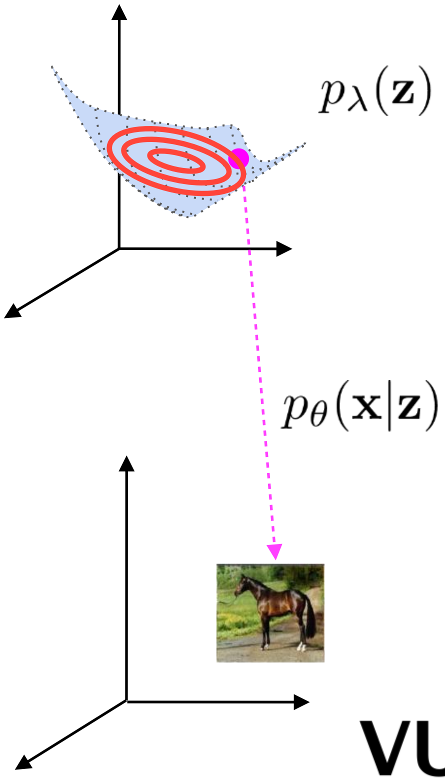
Generative process:

$$1.\ \mathbf{z} \sim p_{\lambda}(\mathbf{z})$$
$$2.\ \mathbf{x} \sim p_{\theta}(\mathbf{x}|\mathbf{z})$$



Generative process:

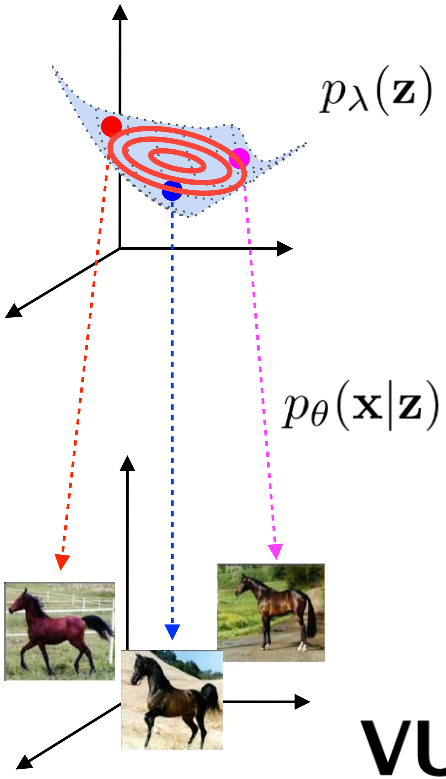
$$1.\ \mathbf{z} \sim p_{\lambda}(\mathbf{z})$$
$$2.\ \mathbf{x} \sim p_{\theta}(\mathbf{x}|\mathbf{z})$$



Generative process:

$$1.\ \mathbf{z} \sim p_\lambda(\mathbf{z})$$

$$2.\ \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$$



GENERATIVE MODELING WITH LATENT VARIABLES

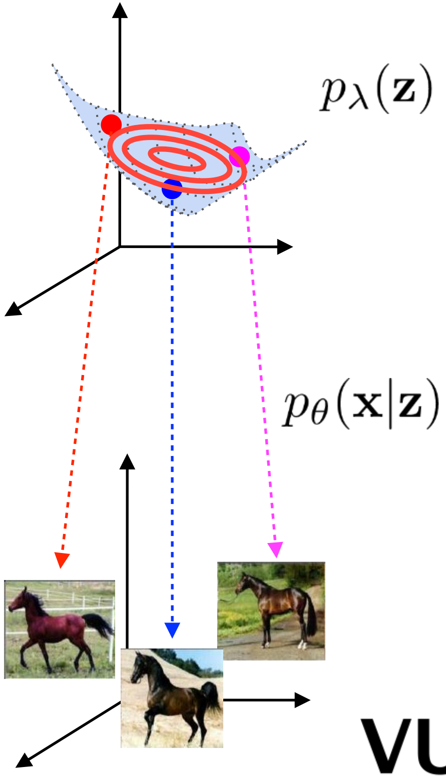
Generative process:

$$1.\ \mathbf{z} \sim p_{\lambda}(\mathbf{z})$$

$$2.\ \mathbf{x} \sim p_{\theta}(\mathbf{x}|\mathbf{z})$$

The log-likelihood function:

$$\log p_{\vartheta}(\mathbf{x}) = \log \int p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})\mathrm{d}\mathbf{z}$$



GENERATIVE MODELING WITH LATENT VARIABLES

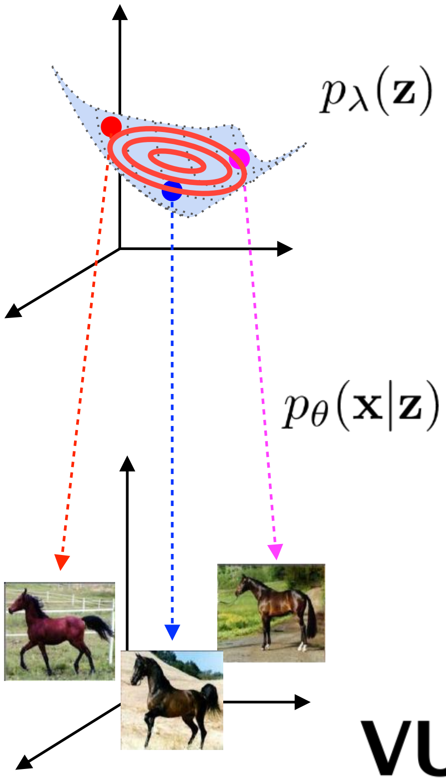
Generative process:

$$1.\ \mathbf{z} \sim p_\lambda(\mathbf{z})$$
$$2.\ \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$$

The log-likelihood function:

$$\log p_\vartheta(\mathbf{x}) = \log \underbrace{\int}_{} p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\mathrm{d}\mathbf{z}$$

How to train such model efficiently?



Let us assume: $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$.

Let us assume: $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$.

And a linear transformation ($\mathbf{W} \in \mathbb{R}^{D \times M}$):

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \mu + \varepsilon, \text{ where } \varepsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$$

Let us assume: $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$.

And a linear transformation ($\mathbf{W} \in \mathbb{R}^{D \times M}$):

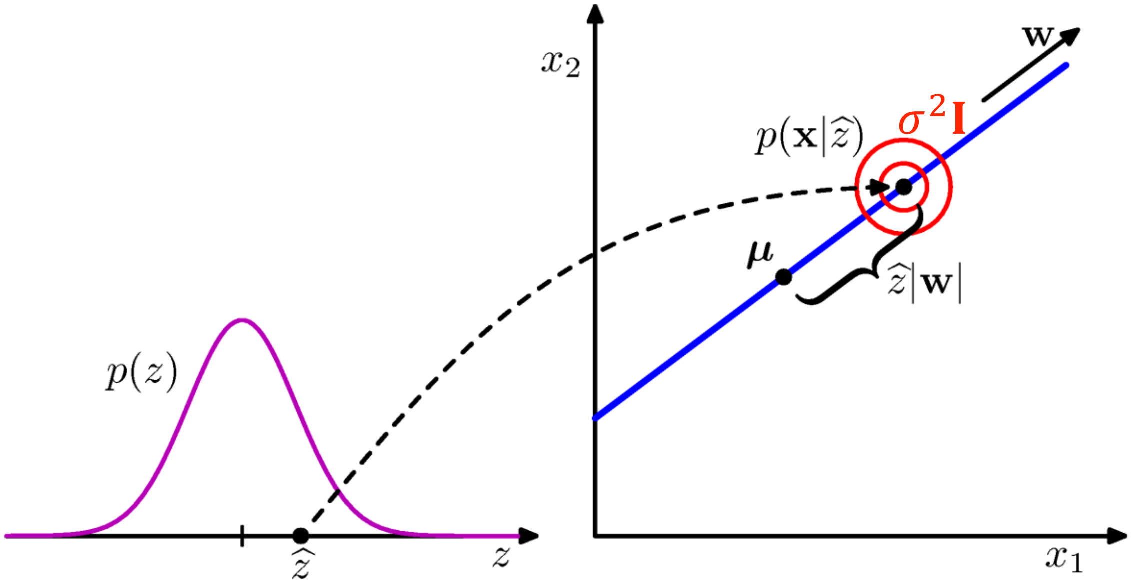
$$\mathbf{x} = \mathbf{W}\mathbf{z} + \mu + \varepsilon, \text{ where } \varepsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$$

that results in the following conditional distribution:

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{W}\mathbf{z} + \mu, \sigma^2 \mathbf{I})$$

Let us assume: $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$.

And the following conditional distribution: $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{W}\mathbf{z} + \mu, \sigma^2\mathbf{I})$



Generative process

Now, the question is how to calculate the log-likelihood:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\mathrm{d}\mathbf{z}$$

LINEAR LATENT VARIABLE MODELS

Now, the question is how to calculate the log-likelihood:

$$p(\mathbf{x}) = \int \underbrace{p(\mathbf{x}|\mathbf{z})}_{\text{Gaussian}} \underbrace{p(\mathbf{z})}_{\text{Gaussian}} \mathrm{d}\mathbf{z}$$

Now, the question is how to calculate the log-likelihood:

$$p(\mathbf{x}) = \int \underbrace{p(\mathbf{x}|\mathbf{z})}_{\text{Gaussian}}\underbrace{p(\mathbf{z})}_{\text{Gaussian}}\mathrm{d}\mathbf{z}$$

$$= \mathcal{N}(\mu, \mathbf{W}\mathbf{W}^{\top} + \sigma^2\mathbf{I})$$

Marginal and Conditional Gaussians

Given a marginal Gaussian distribution for $\mathbf{x}$ and a conditional Gaussian distribution for $\mathbf{y}$ given $\mathbf{x}$ in the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \tag{2.113}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}) \tag{2.114}$$

the marginal distribution of $\mathbf{y}$ and the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ are given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \tag{2.115}$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \tag{2.116}$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}. \tag{2.117}$$

Bishop, “Pattern Recognition and Machine Learning”

Now, the question is how to calculate the log-likelihood:

$$p(\mathbf{x}) = \int \underbrace{p(\mathbf{x}|\mathbf{z})}_{\text{Gaussian}}\underbrace{p(\mathbf{z})}_{\text{Gaussian}}\mathrm{d}\mathbf{z}$$

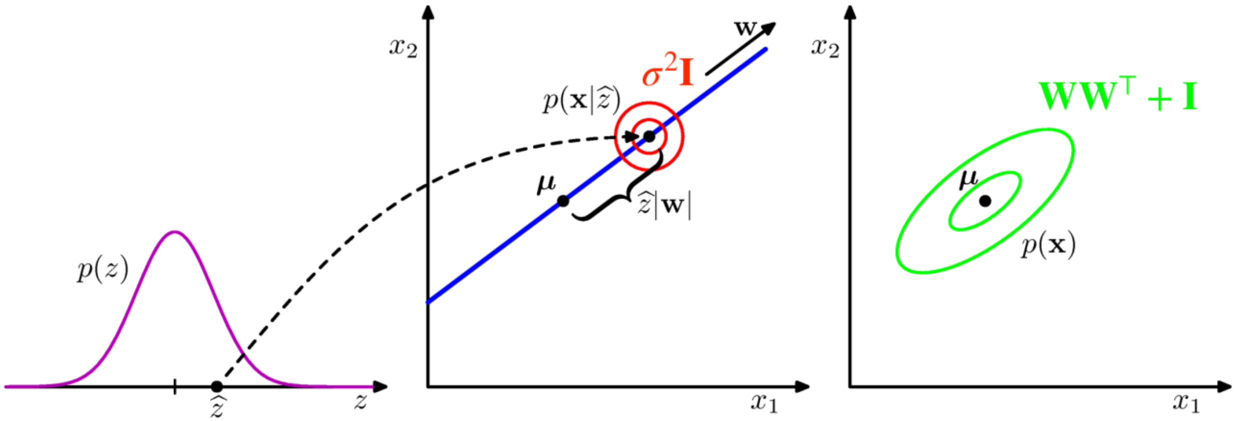
$$= \mathcal{N}(\mu, \mathbf{W}\mathbf{W}^{\top} + \sigma^{2}\mathbf{I})$$

The integral is tractable, and it is again Gaussian!

Now, the question is how to calculate the log-likelihood:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\mathrm{d}\mathbf{z}$$

$$= \mathcal{N}(\mu, \mathbf{W}\mathbf{W}^{\top} + \sigma^2\mathbf{I})$$



Since the model is linear, and all distributions are Gaussians, we can also calculate the posterior over $\mathbf{z}$:

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^{\top}(\mathbf{x} - \mu), \sigma^{-2}\mathbf{M})$$

where:

$$\mathbf{M} = \mathbf{W}^{\top}\mathbf{W} + \sigma^{2}\mathbf{I}$$

Since the model is linear, and all distributions are Gaussians, we can also calculate the posterior over $\mathbf{z}$:

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^{\top}(\mathbf{x} - \mu), \sigma^{-2}\mathbf{M})$$

where:

$$\mathbf{M} = \mathbf{W}^{\top}\mathbf{W} + \sigma^2\mathbf{I}$$

Marginal and Conditional Gaussians

Given a marginal Gaussian distribution for $\mathbf{x}$ and a conditional Gaussian distribution for $\mathbf{y}$ given $\mathbf{x}$ in the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \tag{2.113}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}) \tag{2.114}$$

the marginal distribution of $\mathbf{y}$ and the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ are given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \tag{2.115}$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \tag{2.116}$$

where

$$\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}. \tag{2.117}$$

The final model is the following ($\mathbf{W} \in \mathbb{R}^{D\times M}$):

$$p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$$
$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{Wz} + \mu, \sigma^2\mathbf{I})$$
$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^\top(\mathbf{x} - \mu), \sigma^{-2}\mathbf{M})$$

where $\mathbf{M} = \mathbf{W}^\top\mathbf{W} + \sigma^2\mathbf{I}$.

and the marginal distribution:

$$p(\mathbf{x}) = \mathcal{N}(\mu, \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I})$$

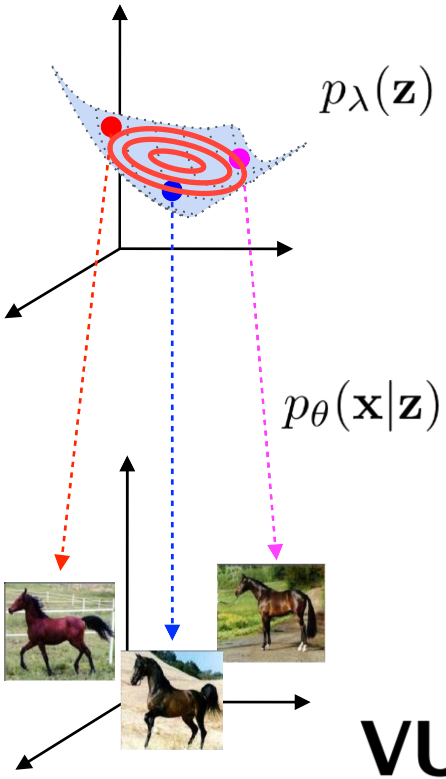
PART THREE: VARIATIONAL AUTOENCODERS

Generative process:

$$1.\ \mathbf{z} \sim p_{\lambda}(\mathbf{z})$$
$$2.\ \mathbf{x} \sim p_{\theta}(\mathbf{x}|\mathbf{z})$$

The log-likelihood function:

$$\log p_{\vartheta}(\mathbf{x}) = \log \underbrace{\int}\, p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})\mathrm{d}\mathbf{z}$$



How to train such model efficiently?

Now we consider non-linear transformations.

Let us assume: $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$.

Linear model: $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{Wz} + \mu, \sigma^2 \mathbf{I})$

Let us assume: $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$.

Linear model: $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{Wz} + \mu, \sigma^2\mathbf{I})$

Now, we consider: $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(f(\mathbf{z}; \mathbf{W}), \sigma^2\mathbf{I})$.

Let us assume: $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$.

Linear model: $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{Wz} + \mu, \sigma^2\mathbf{I})$

Now, we consider: $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(f(\mathbf{z}; \mathbf{W}), \sigma^2\mathbf{I})$.

Since f could be any non-linear transformation, Prof. Bishop cannot provide us any tricks to solve the integral:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\mathrm{d}\mathbf{z}$$

This is an infinite mixture of Gaussians.

Let us assume: $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$.

Linear model: $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{W}\mathbf{z} + \mu, \sigma^2\mathbf{I})$

Now, we consider: $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(f(\mathbf{z}; \mathbf{W}), \sigma^2\mathbf{I})$.

Since f could be any non-linear transformation, Prof. Bishop cannot provide us any tricks to solve the integral:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\mathrm{d}\mathbf{z}$$

This is an infinite mixture of Gaussians.

BUT we can use variational inference!
(Chapter 10 in Bishop's book 😉)

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

$$
\begin{aligned}
\log p_\vartheta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z})\Big] - \mathrm{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x}) || p_\lambda(\mathbf{z})\Big)
\end{aligned}
$$

$$
\begin{aligned}
\log p_{\vartheta}(\mathbf{x}) &= \log \int p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})\mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{q_{\phi}(\mathbf{z}|\mathbf{x})} p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})\mathrm{d}\mathbf{z} \\
&\geq \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}\Big[\log p_{\theta}(\mathbf{x}|\mathbf{z})\Big] - \mathrm{KL}\Big(q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\lambda}(\mathbf{z})\Big)
\end{aligned}
$$

Variational posterior

$$\log p_\vartheta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\mathrm{d}\mathbf{z}$$

$$= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\mathrm{d}\mathbf{z}$$

$$\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{d}\mathbf{z}$$

$$= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z})\Big] - \mathrm{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})\Big)$$

Variational posterior

We can learn a separate *q* for each **x**, but it would be too complicated. Therefore, we use **amortization**.

$$\begin{aligned}
\log p_\vartheta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\mathrm{d}\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z})\Big] - \mathrm{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})\Big)
\end{aligned}$$

Jensen's inequality

$\log \mathbb{E}_q[\ldots] \geq \mathbb{E}_q[\log \ldots]$

$$
\begin{aligned}
\log p_{\vartheta}(\mathbf{x}) &= \log \int p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{q_{\phi}(\mathbf{z}|\mathbf{x})} p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&\geq \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \mathrm{d}\mathbf{z} \\
&= \underbrace{\mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} \Big[\log p_{\theta}(\mathbf{x}|\mathbf{z}) \Big] - \mathrm{KL}\Big(q_{\phi}(\mathbf{z}|\mathbf{x}) || p_{\lambda}(\mathbf{z}) \Big)}_{\text{Evidence Lower BOund (ELBO)}}
\end{aligned}
$$

$$\begin{aligned}
\log p_{\vartheta}(\mathbf{x}) &= \log \int p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})\mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{q_{\phi}(\mathbf{z}|\mathbf{x})} p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})\mathrm{d}\mathbf{z} \\
&\geq \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \mathrm{d}\mathbf{z} \\
&= \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}\Big[\log p_{\theta}(\mathbf{x}|\mathbf{z})\Big]}_{\text{Reconstruction error (RE)}} - \underbrace{\mathrm{KL}\Big(q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\lambda}(\mathbf{z})\Big)}_{\text{Regularization (KL)}}
\end{aligned}$$

$$
\begin{aligned}
\log p_\vartheta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) || p_\lambda(\mathbf{z})\right)
\end{aligned}
$$

decoder

encoder

marginal
(prior)

$$\log p_\vartheta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z}) d\mathbf{z}$$

$$= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z}) d\mathbf{z}$$

$$\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

$$= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) || p_\lambda(\mathbf{z})\right)$$

decoder

encoder

marginal (prior)

= Variational Auto-Encoder

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS (A DIFFERENT DERIVATION)

$$
\begin{aligned}
\ln p_\theta(\boldsymbol{x}) &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x})] && \text{(}p_\theta(\boldsymbol{x})\text{ does not depend on }\boldsymbol{z}\text{)} \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{z}|\boldsymbol{x})p_\theta(\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{(Multiply by } \tfrac{p_\theta(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\text{)} \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{(Bayes' rule)} \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{(Multiply numerator and denominator by } q_\phi(\boldsymbol{z}|\boldsymbol{x})\text{)}
\end{aligned}
$$

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS (A DIFFERENT DERIVATION)

$$
\begin{aligned}
\ln p_\theta(\boldsymbol{x}) &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x})] && \text{($p_\theta(\boldsymbol{x})$ does not depend on $\boldsymbol{z}$)} \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{z}|\boldsymbol{x}) p_\theta(\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{(Multiply by $\frac{p_\theta(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}$)} \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{(Bayes' rule)} \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x},\boldsymbol{z}) q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x}) q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] && \text{(Multiply numerator and denominator by $q_\phi(\boldsymbol{z}|\boldsymbol{x})$)} \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z}) p_\lambda(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathbb{E}_{q_\phi(\boldsymbol{Z}|\boldsymbol{X})}\left[\ln \frac{q_\phi(\boldsymbol{Z}|\boldsymbol{X})}{p_\lambda(\boldsymbol{z})}\right] + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}); p_\lambda(\boldsymbol{z})\right) + D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}); p_\theta(\boldsymbol{z}|\boldsymbol{x})\right)
\end{aligned}
$$

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS (A DIFFERENT DERIVATION)

$$\begin{aligned}
\ln p_\theta(\boldsymbol{x}) &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x})] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{z}|\boldsymbol{x})p_\theta(\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p_\lambda(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\lambda(\boldsymbol{z})}\right] + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}); p_\lambda(\boldsymbol{z})\right)}_{\textit{evidence lower bound}\ \textbf{(ELBO)}} + D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}); p_\theta(\boldsymbol{z}|\boldsymbol{x})\right)
\end{aligned}$$

≥ 0

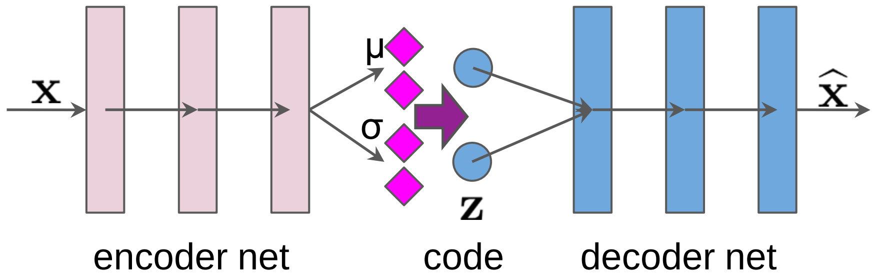
VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS (A DIFFERENT DERIVATION)

$$
\begin{aligned}
\ln p_\theta(\boldsymbol{x}) &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x})] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{z}|\boldsymbol{x})p_\theta(\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p_\lambda(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathbb{E}_{q_\phi(\boldsymbol{Z}|\boldsymbol{X})}\left[\ln \frac{q_\phi(\boldsymbol{Z}|\boldsymbol{X})}{p_\lambda(\boldsymbol{z})}\right] + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\ln \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x}); p_\lambda(\boldsymbol{z})\big)}_{\textit{evidence lower bound}\ \textbf{(ELBO)}} + \underbrace{D_{KL}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x}); p_\theta(\boldsymbol{z}|\boldsymbol{x})\big)}_{\geq 0}
\end{aligned}
$$

If variational posterior is poorly chosen, then the lower bound is very loose.

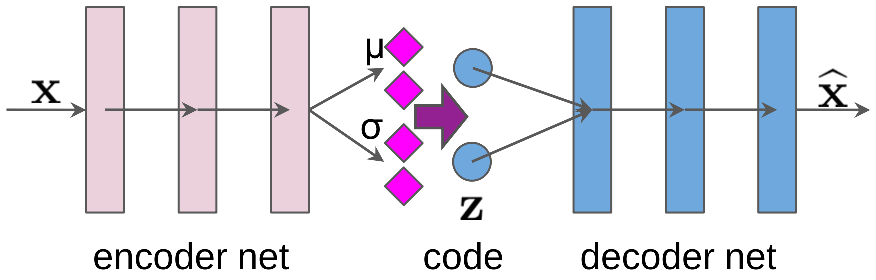
VU

Variational posterior (**encoder**) and the likelihood function (**decoder**) are parameterized by neural networks.



VARIATIONAL AUTO-ENCODERS

Variational posterior (**encoder**) and the likelihood function (**decoder**) are parameterized by neural networks.



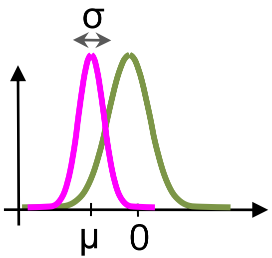
Reparameterization trick:

move the stochasticity to independent random variables

$$z = f(\theta, \varepsilon), \varepsilon \sim p(\varepsilon)$$

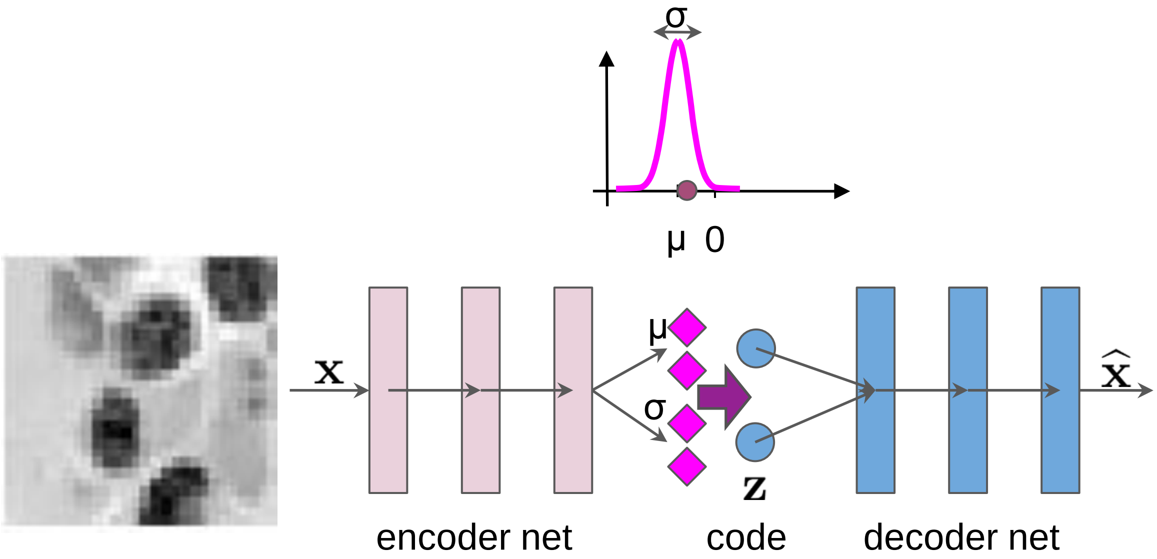
e.g.

$$z = \mu + \sigma \cdot \varepsilon, \varepsilon \sim \mathcal{N}(0,1)$$



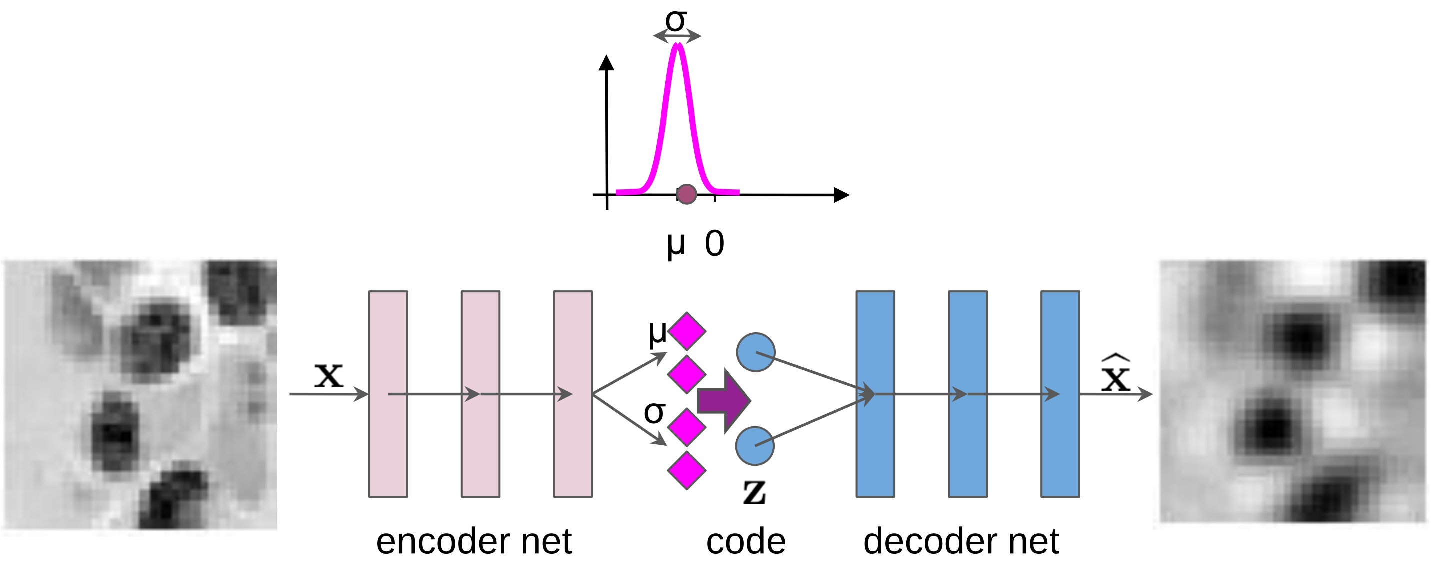
VAE copies input to output through a **bottleneck**.

VAE learns a **code** of the data.



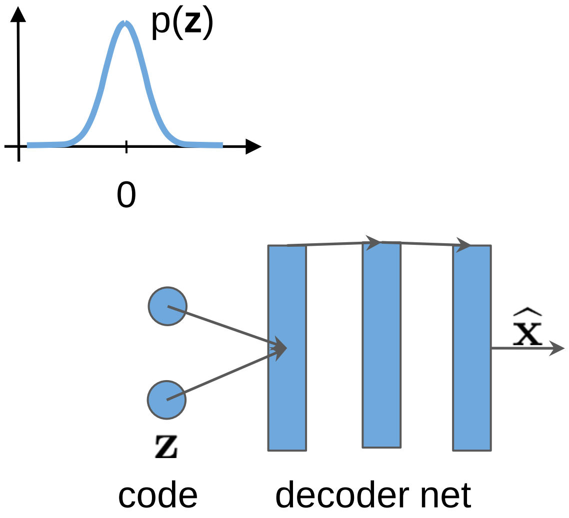
VAE copies input to output through a **bottleneck**.

VAE learns a **code** of the data.



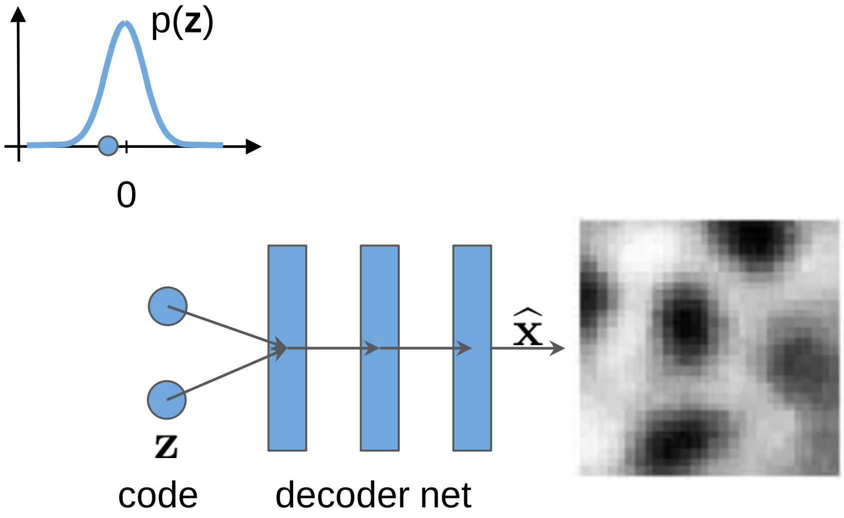
VAE has a **marginal** on the latent code.

VAE can **generate** new data.



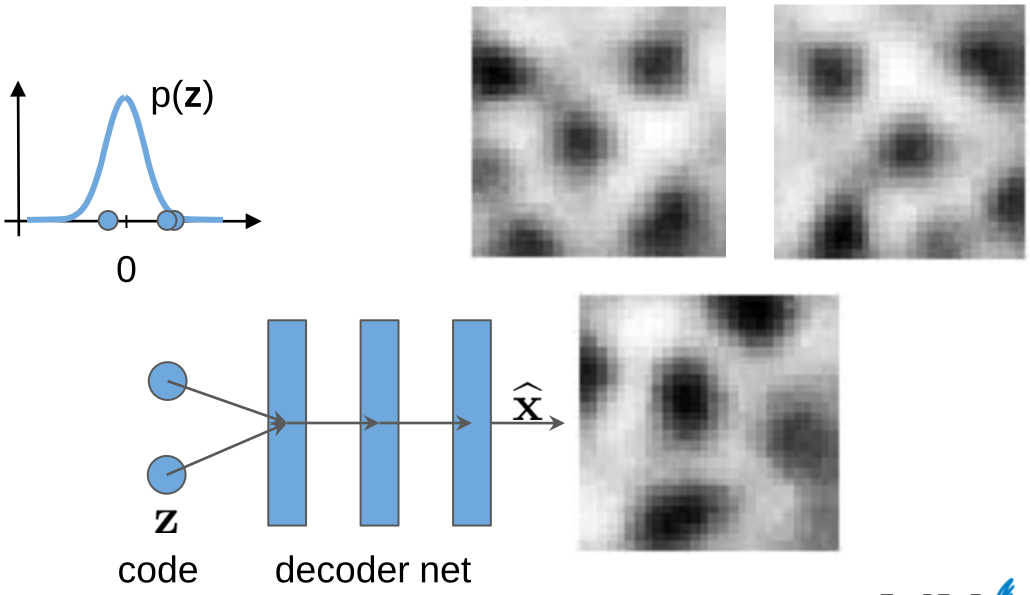
VAE has a **marginal** on the latent code.

VAE can **generate** new data.



VAE has a **marginal** on the latent code.

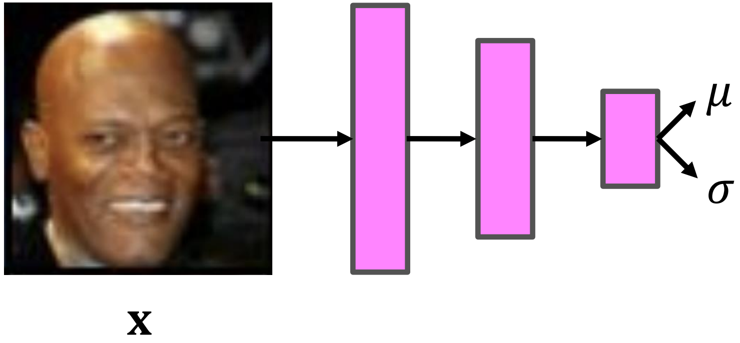
VAE can **generate** new data.



VANILLA VAE

$\mathbf{x}$

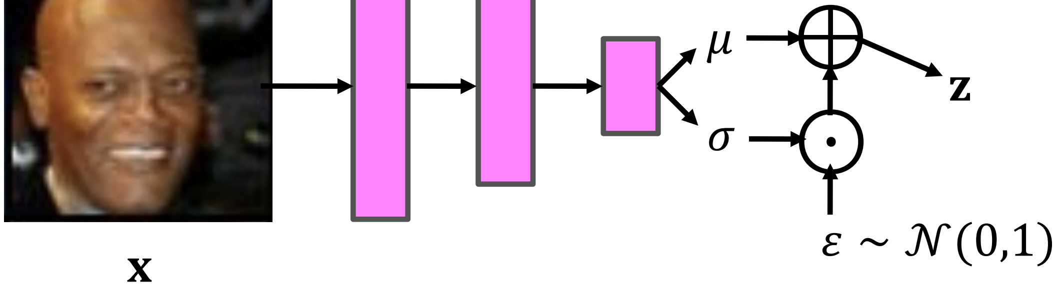
VANILLA VAE



Example architecture for the encoder:

$\mathbf{x}$ → Linear(D, 300) → ReLU → Linear(300, 2M) → split to 2 vectors

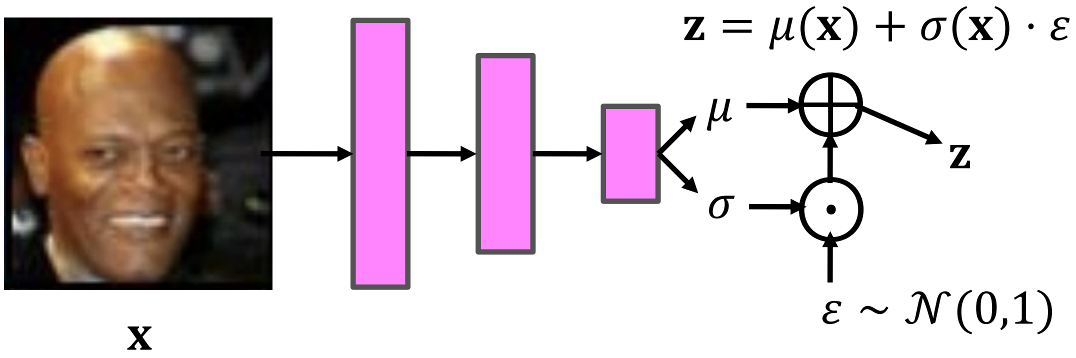
$$\mathbf{z} = \mu(\mathbf{x}) + \sigma(\mathbf{x}) \cdot \varepsilon$$



Reparameterization trick!

Example architecture for the encoder:

x → Linear(D, 300) → ReLU → Linear(300, 2*M*) → split to 2 vectors

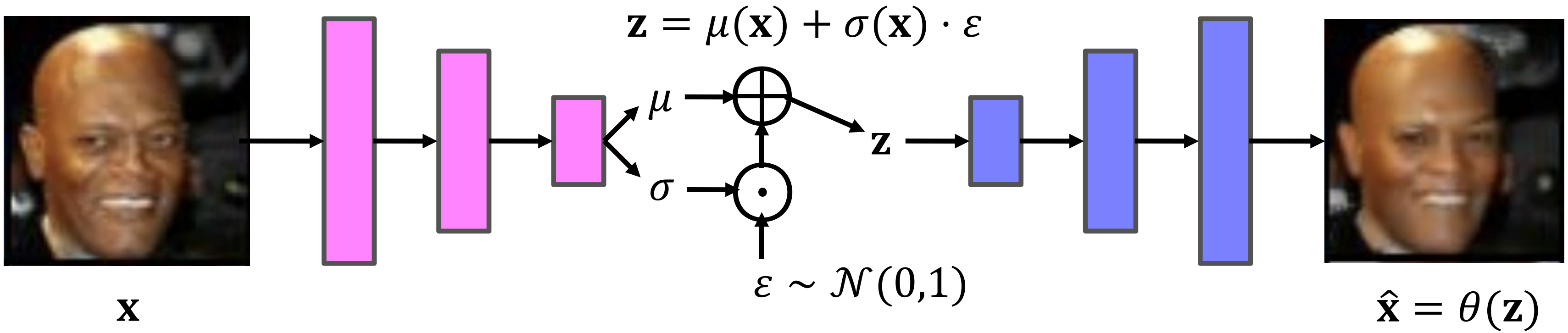


Example architecture for the encoder:

x → Linear(D, 300) → ReLU → Linear(300, 2*M*) → split to 2 vectors

No non-linearity here!
We model means and log-std
for Gaussian.

VANILLA VAE



Example architecture for the encoder:

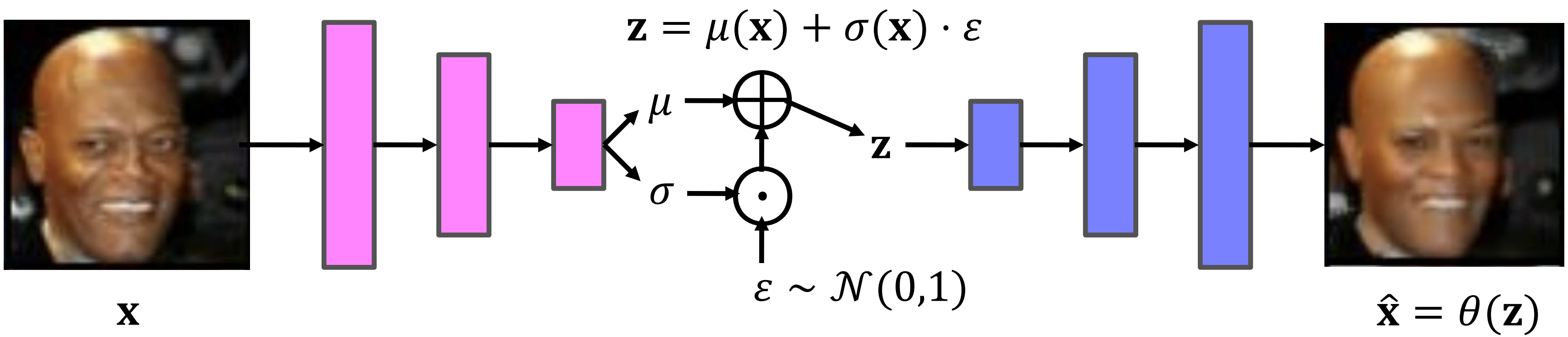
x → Linear(D, 300) → ReLU → Linear(300, 2*M*) → split to 2 vectors

Example architecture for the decoder:

z → Linear(*M*, 300) → ReLU → Linear(300, *D*) → means

No non-linearity here!
We model means only.

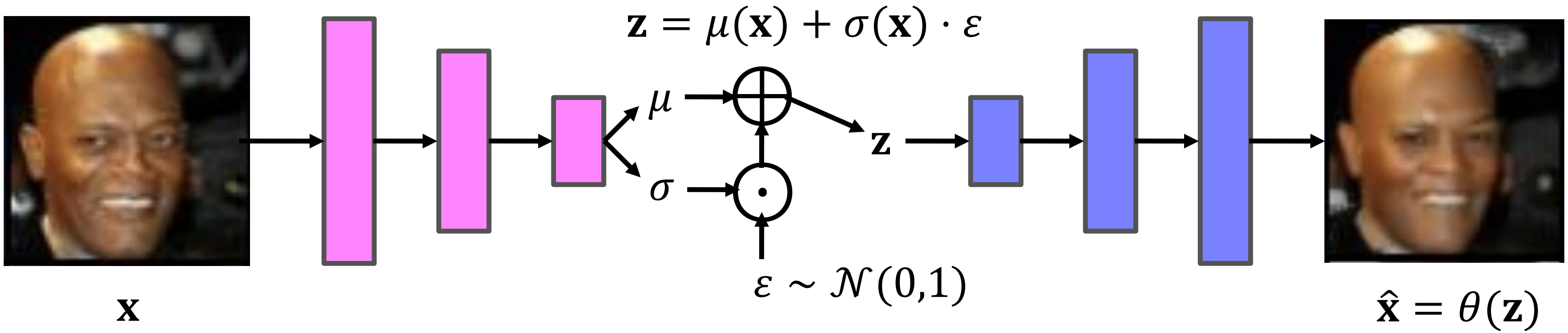
VANILLA VAE



We approximate expected values using a single sample:

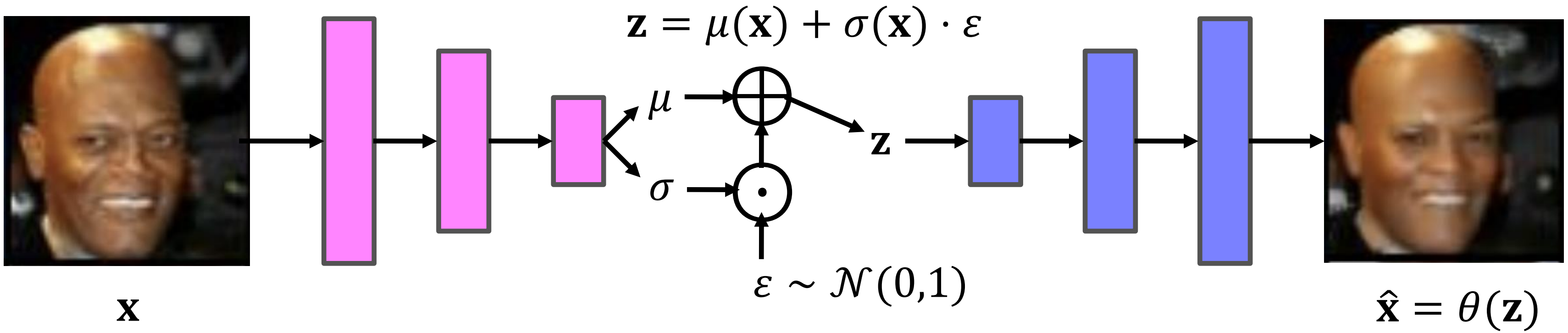
$$ELBO = \underbrace{\ln \mathcal{N}(\mathbf{x}|\theta(\mathbf{z}), 1)}_{p_\theta(\mathbf{x}|\mathbf{z})} - [\underbrace{\ln \mathcal{N}(\mathbf{z}|\mu(\mathbf{x}), \sigma^2(\mathbf{x}))}_{q_\phi(\mathbf{z}|\mathbf{x})} - \underbrace{\ln \mathcal{N}(\mathbf{z}|0,1)}_{p_\lambda(\mathbf{z})}]$$

VANILLA VAE



We approximate expected values using a single sample:

$$ELBO = \underbrace{\ln \mathcal{N}(\mathbf{x}|\theta(\mathbf{z}), 1)}_{\text{RE}} - \underbrace{[\ln \mathcal{N}(\mathbf{z}|\mu(\mathbf{x}), \sigma^2(\mathbf{x})) - \ln \mathcal{N}(\mathbf{z}|0,1)]}_{\text{KL}}$$

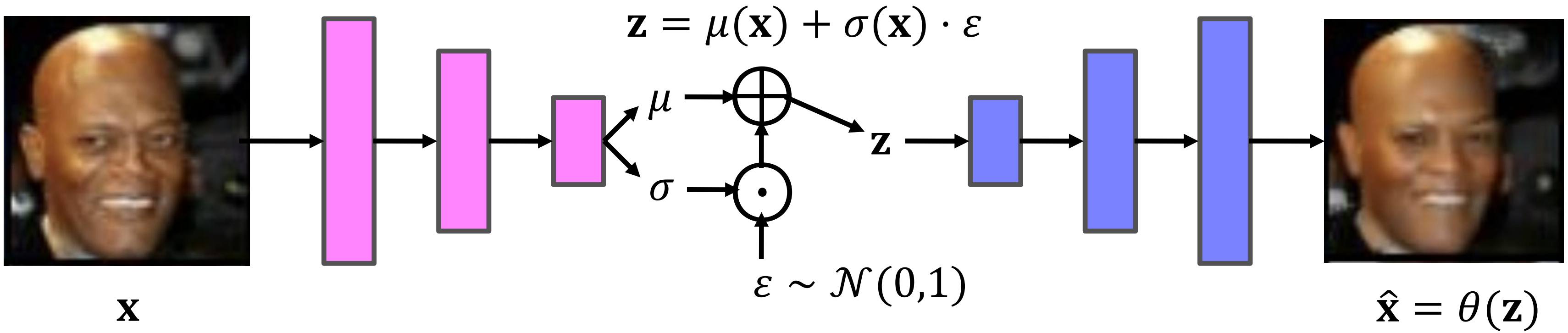


We approximate expected values using a single sample:

We assume a Gaussian variational posterior.

$$ELBO = \underbrace{\ln \mathcal{N}(\mathbf{x}|\theta(\mathbf{z}), 1)}_{\text{RE}} - \underbrace{[\ln \mathcal{N}(\mathbf{z}|\mu(\mathbf{x}), \sigma^2(\mathbf{x})) - \ln \mathcal{N}(\mathbf{z}|0,1)]}_{\text{KL}}$$

We assume a standard Gaussian prior.

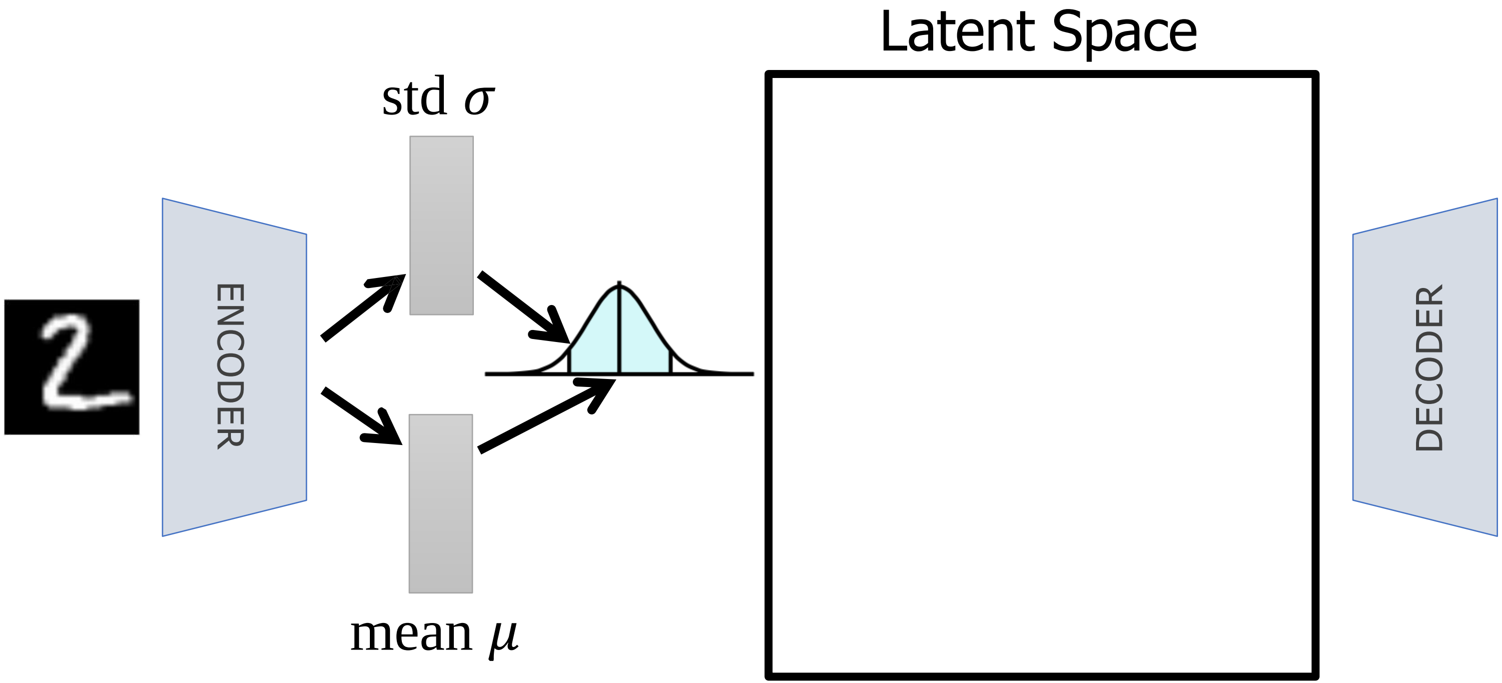


We approximate expected values using a single sample:

$$ELBO = \underbrace{\ln\mathcal{N}(\mathbf{x}|\theta(\mathbf{z}),1)}_{\text{RE}} - \underbrace{[\ln\mathcal{N}(\mathbf{z}|\mu(\mathbf{x}),\sigma^2(\mathbf{x})) - \ln\mathcal{N}(\mathbf{z}|0,1)]}_{\text{KL}}$$

REMEMBER! We cannot pick an arbitrary distribution. We must choose a distribution that is appropriate for our data.
Real-valued -> e.g., Gaussian
Binary -> Bernoulli

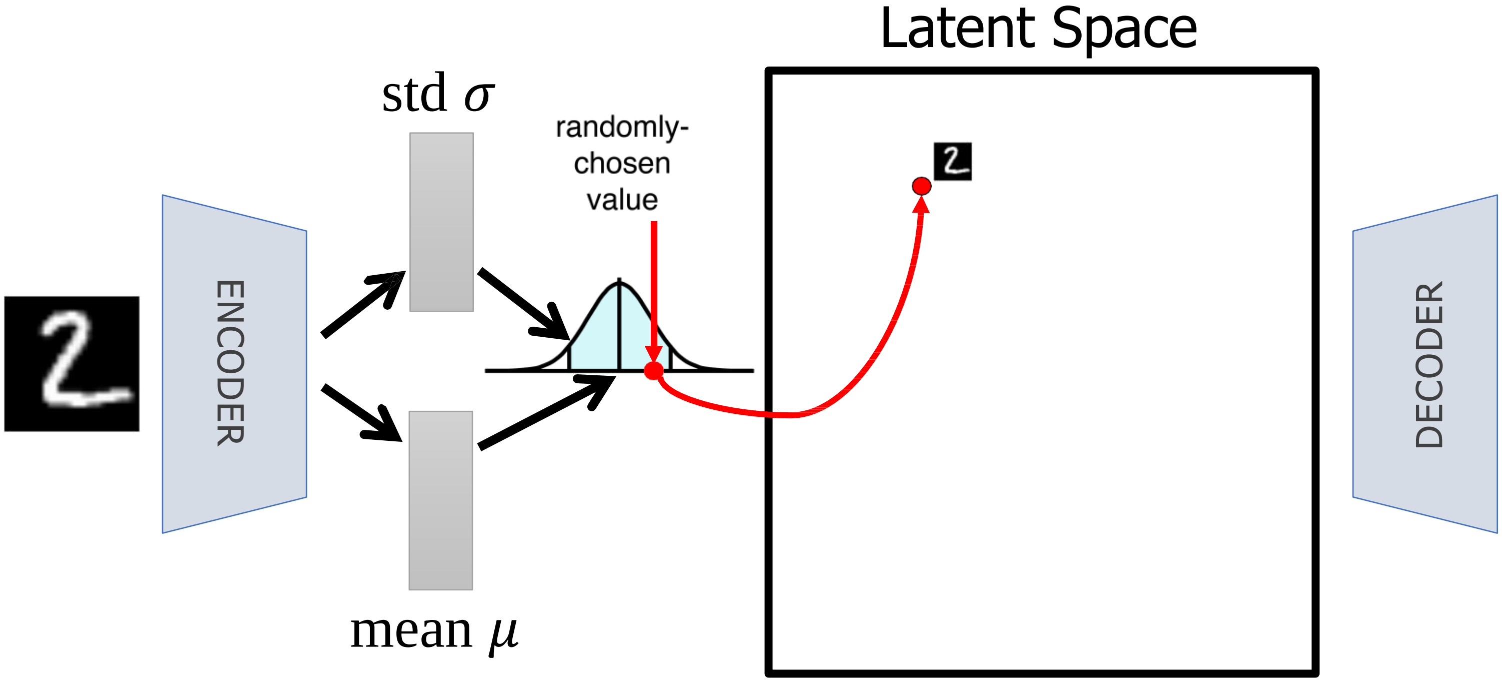
LATENT SPACE OF VAE



Encode the first sample (a “2”) and find μ_1, σ_1

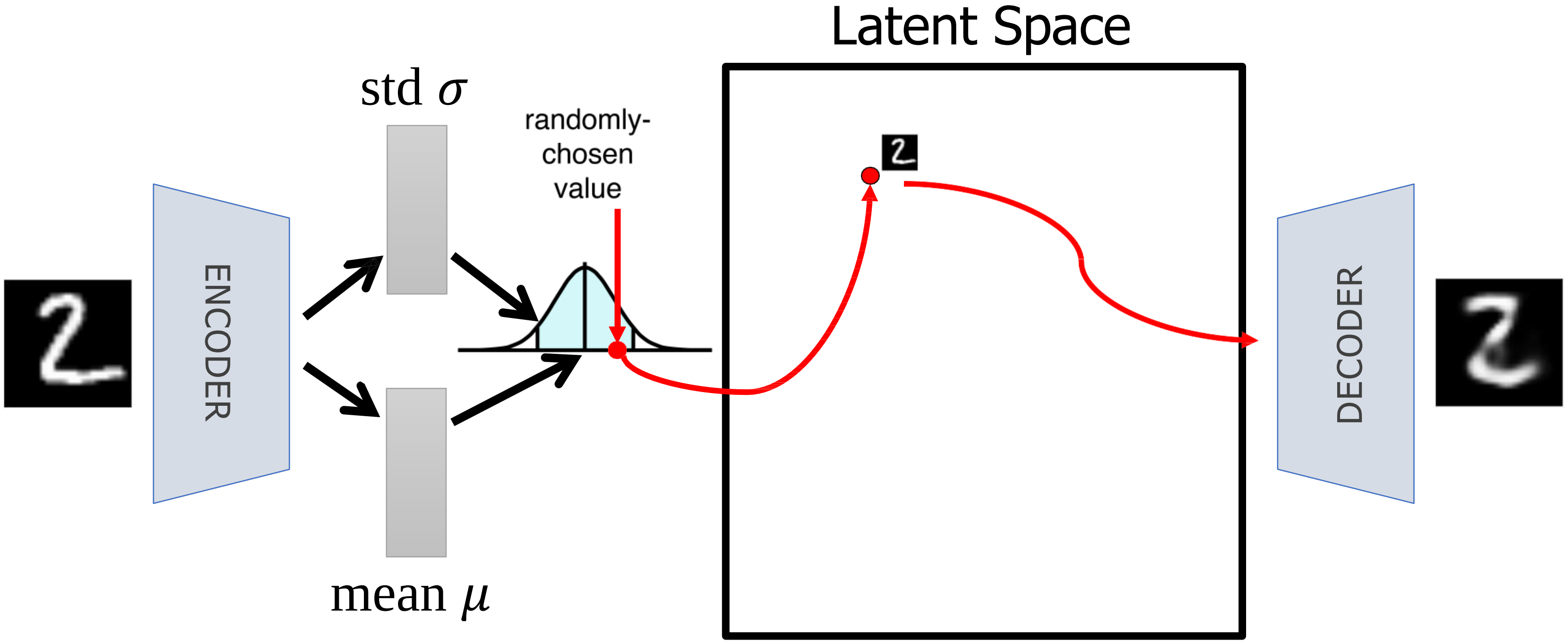
Example from: https://web.cs.hacettepe.edu.tr/~erkut/cmp784.f21/lectures.html

LATENT SPACE OF VAE



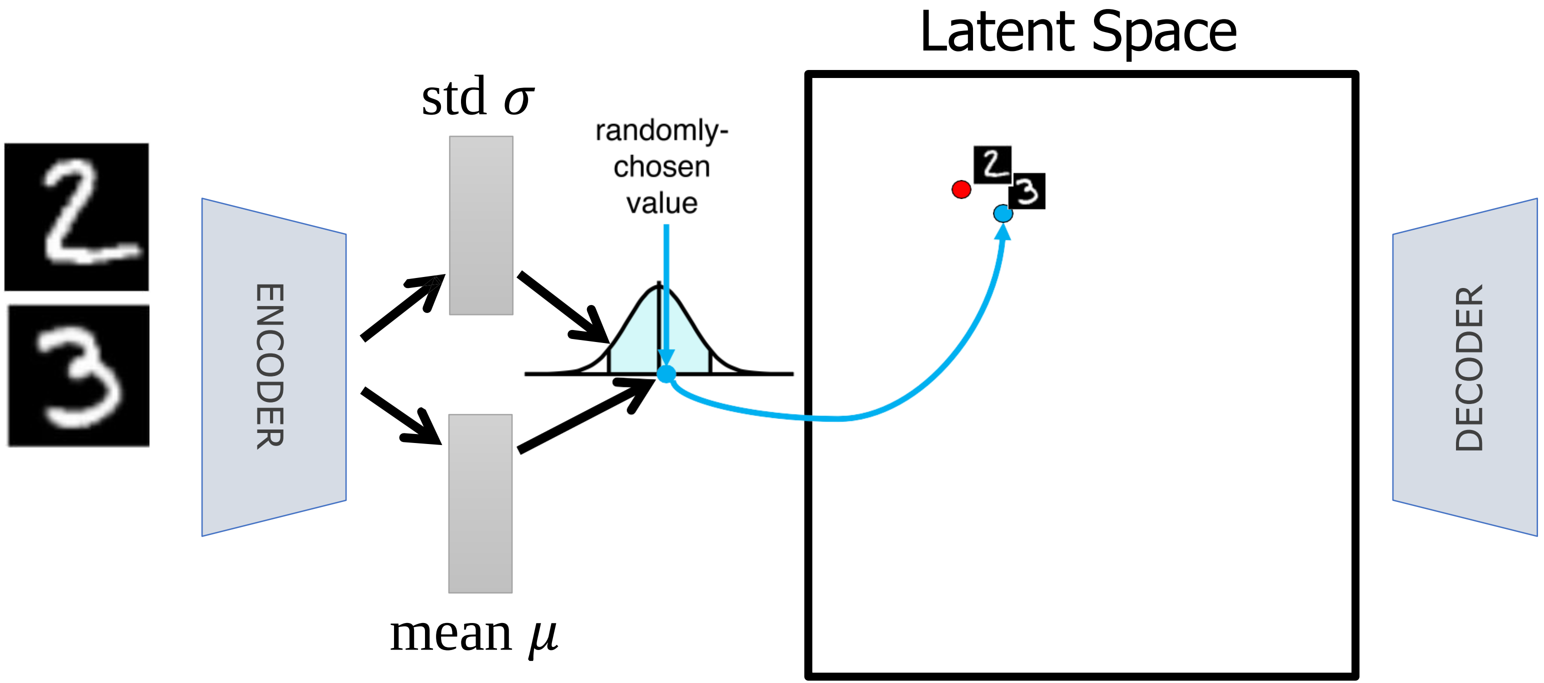
Sample $\boldsymbol{z}_1 \sim N(\mu_1, \sigma_1)$

LATENT SPACE OF VAE



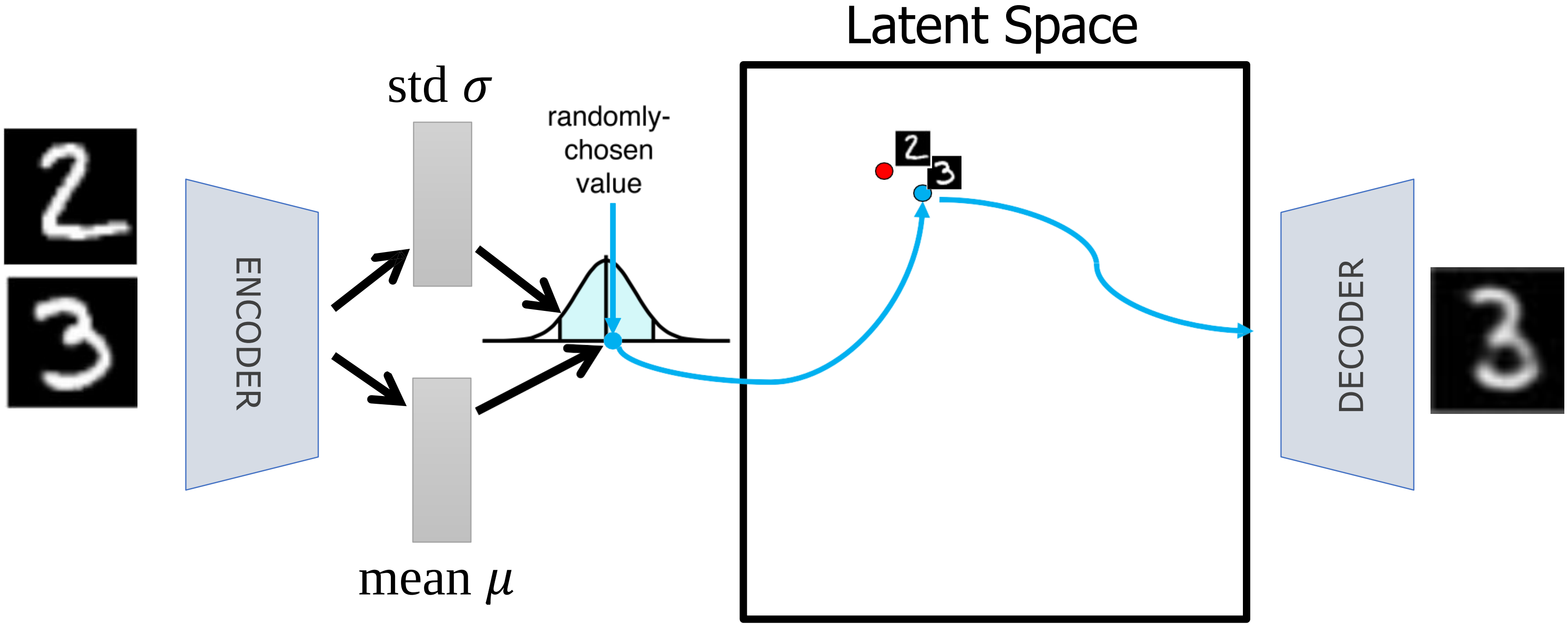
Denote to $\hat{\boldsymbol{x}}_1$

LATENT SPACE OF VAE



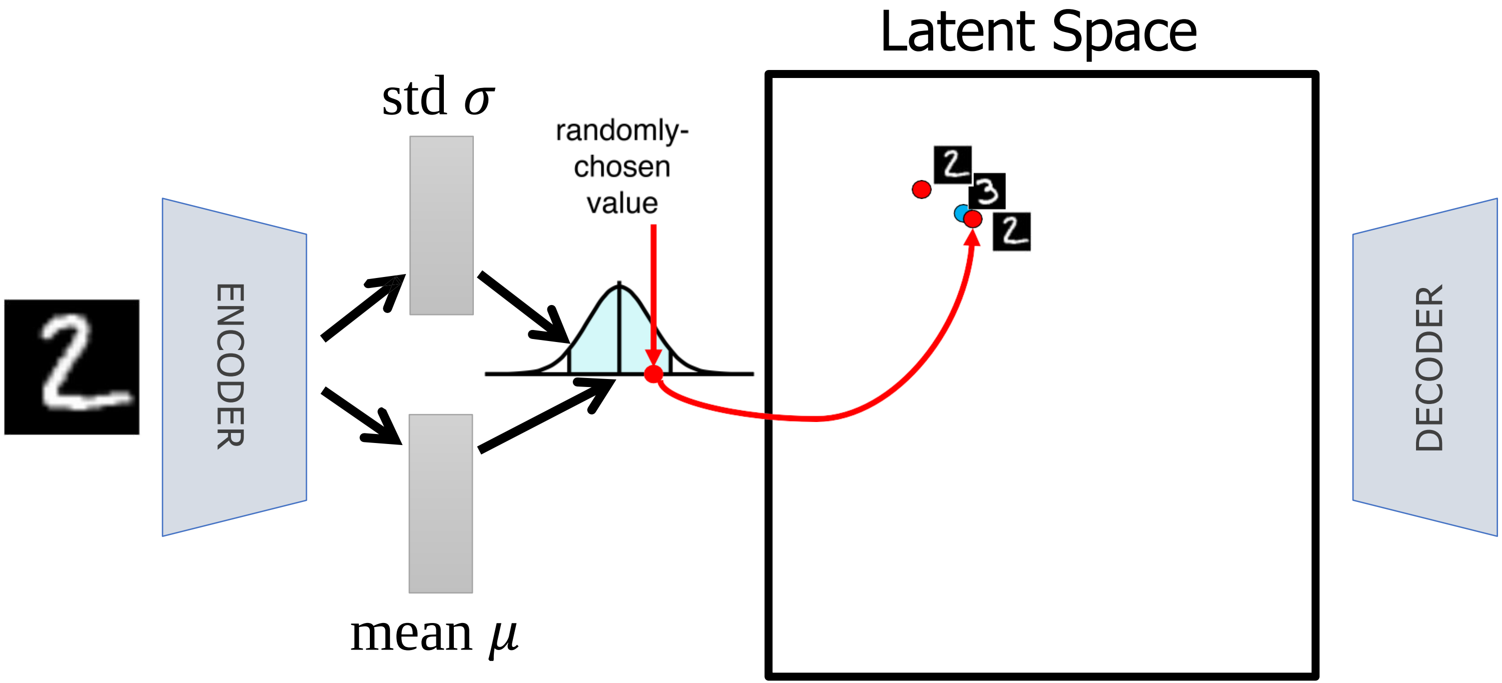
Encode the first sample (a "3") and find μ_2, σ_2, and sample $\boldsymbol{z}_2 \sim N(\mu_2, \sigma_2)$

LATENT SPACE OF VAE



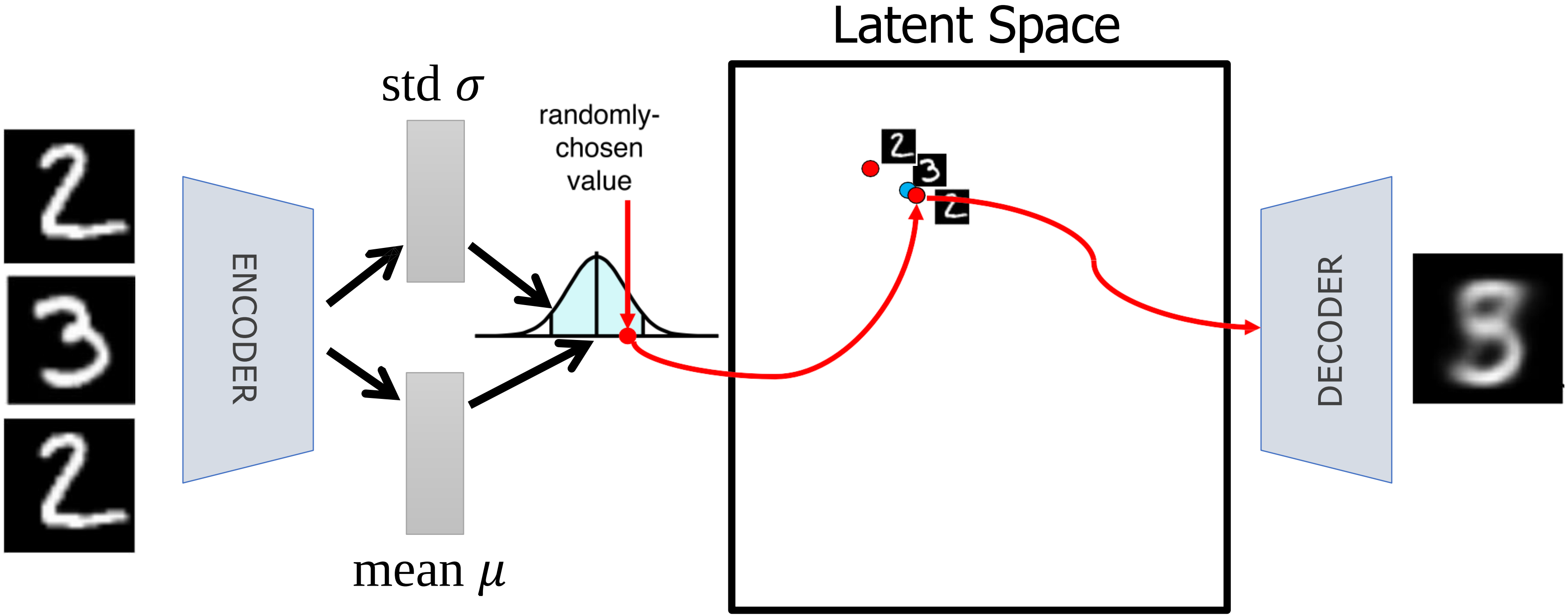
Decode to $\hat{\boldsymbol{x}}_2$

LATENT SPACE OF VAE



Train with the first sample (a "2") again and find μ_1, σ_1. However, $\boldsymbol{z}_1 \sim N(\mu_1, \sigma_1)$ will not be the same. It can happen to be close to the "3" in latent space.

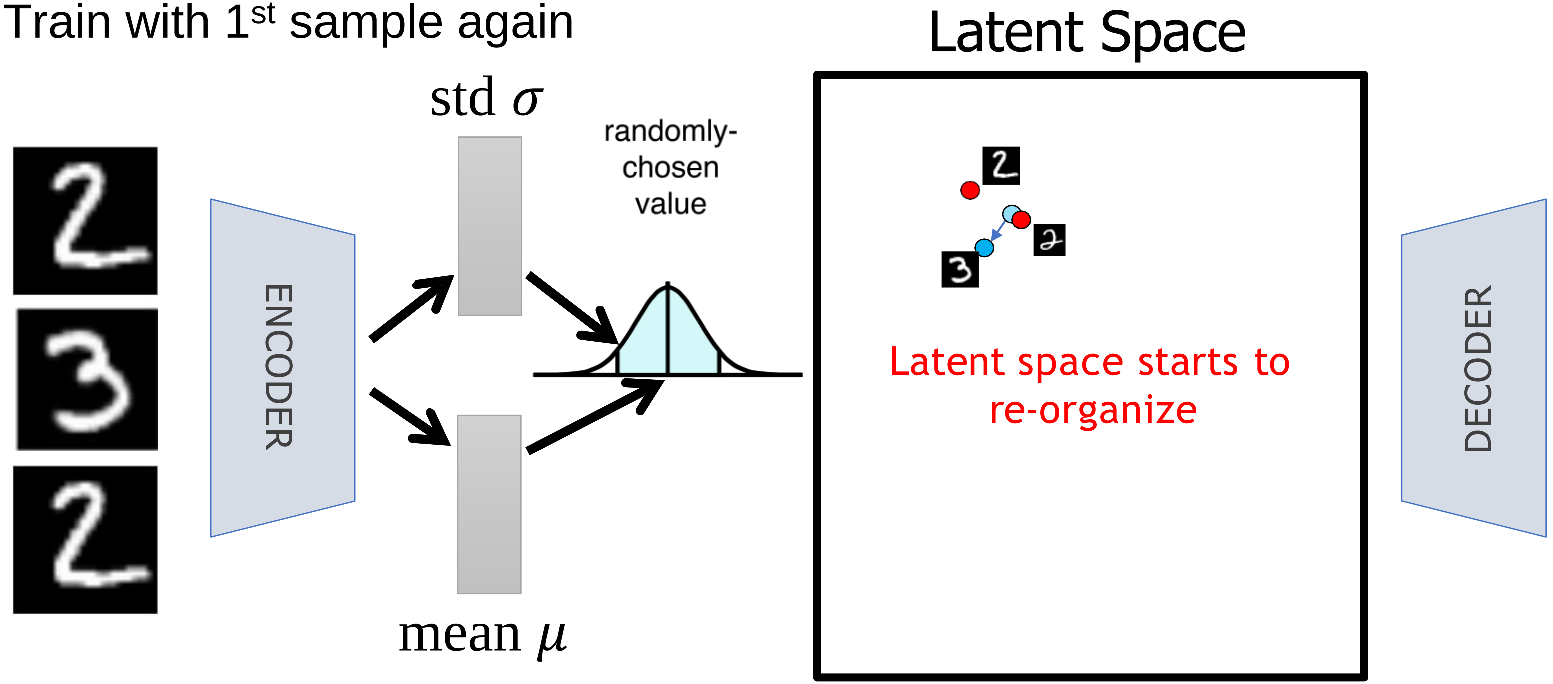
LATENT SPACE OF VAE



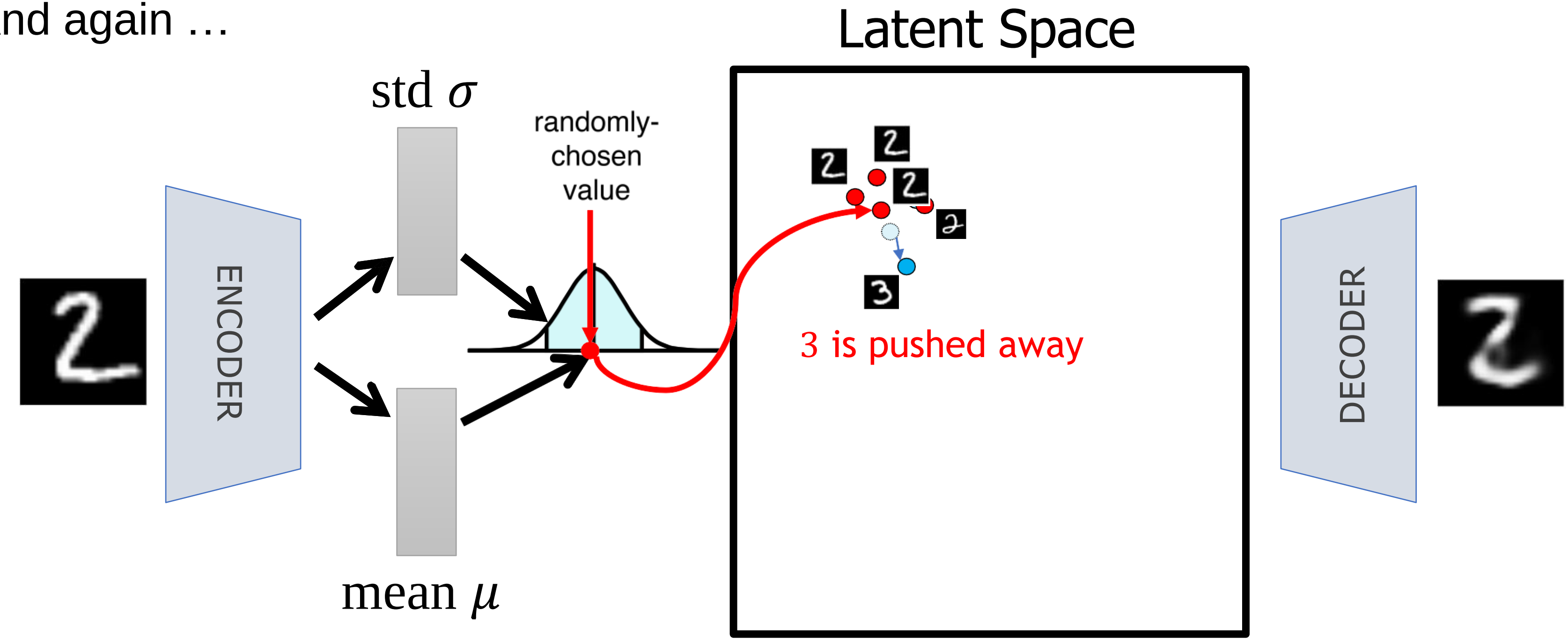
Decode to $\hat{\boldsymbol{x}}_1$. Since the decoder only knows how to map from latent space to $\hat{\boldsymbol{x}}$ space, it will return a “3”.

LATENT SPACE OF VAE

Train with 1st sample again

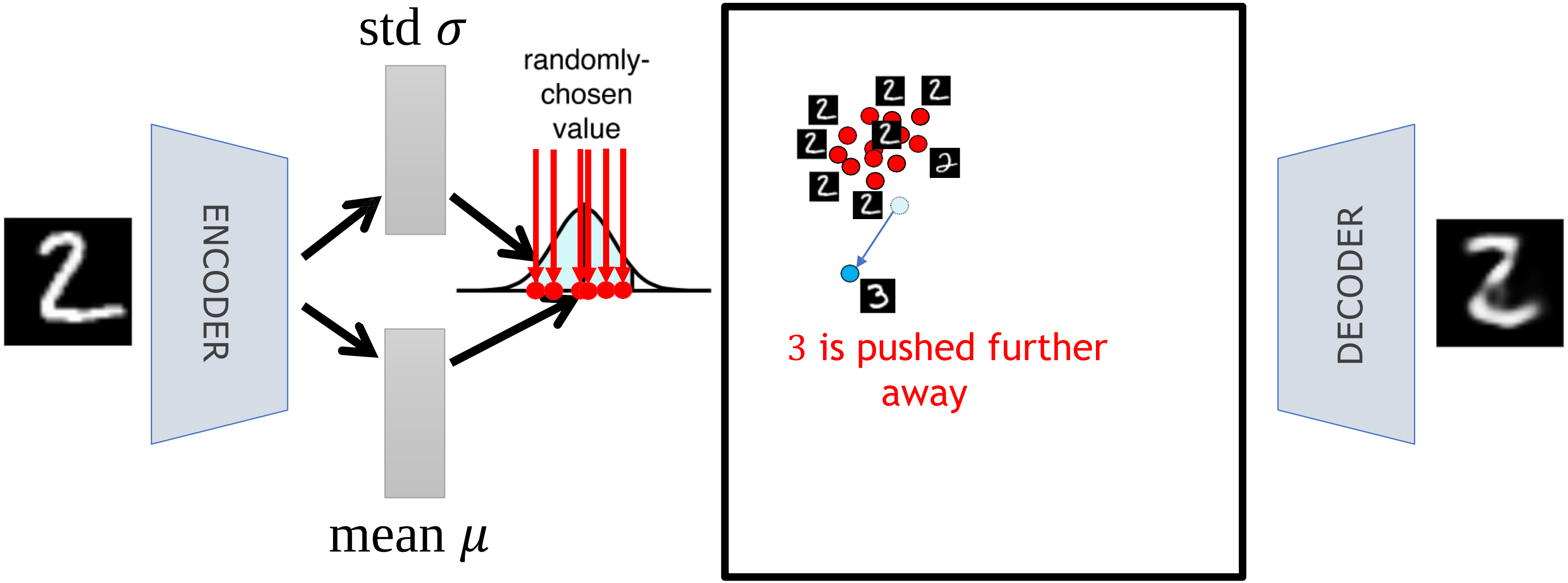


And again …


Latent Space
std σ
randomly-chosen value
ENCODER
DECODER
mean μ
3 is pushed away

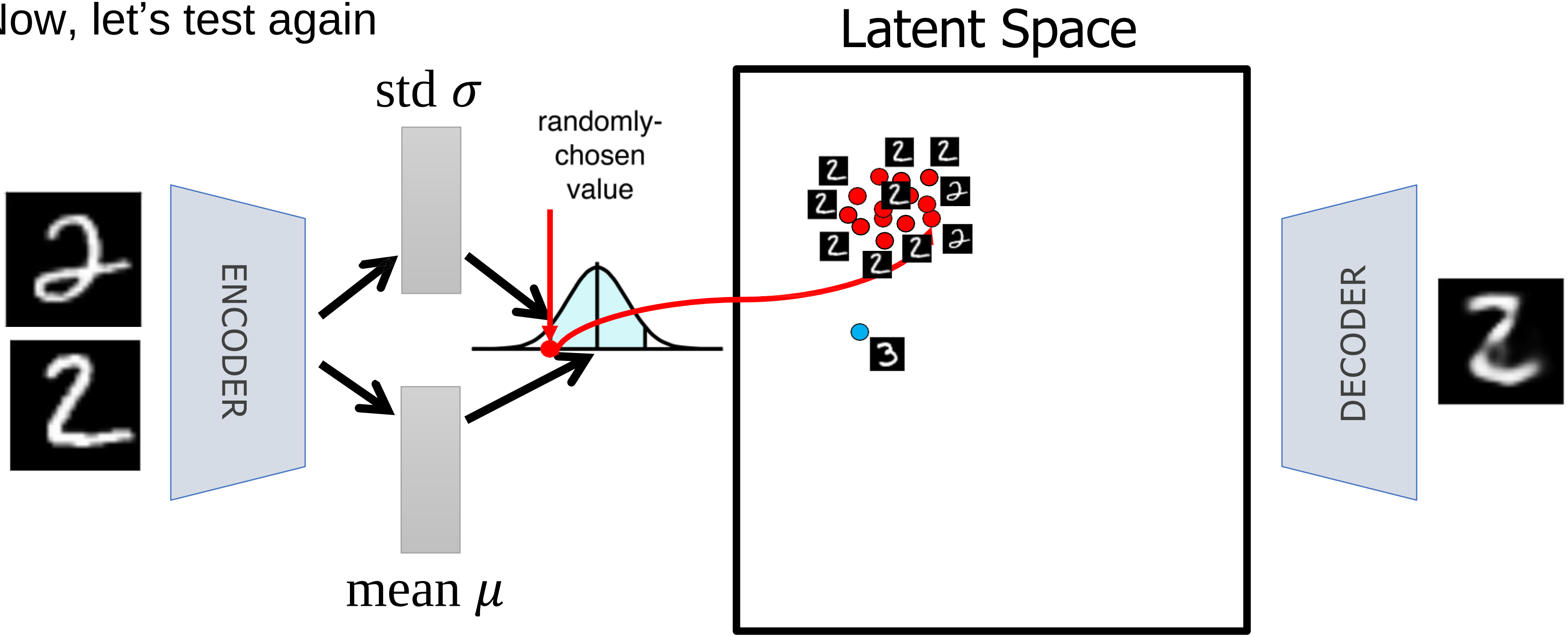
LATENT SPACE OF VAE

Many times

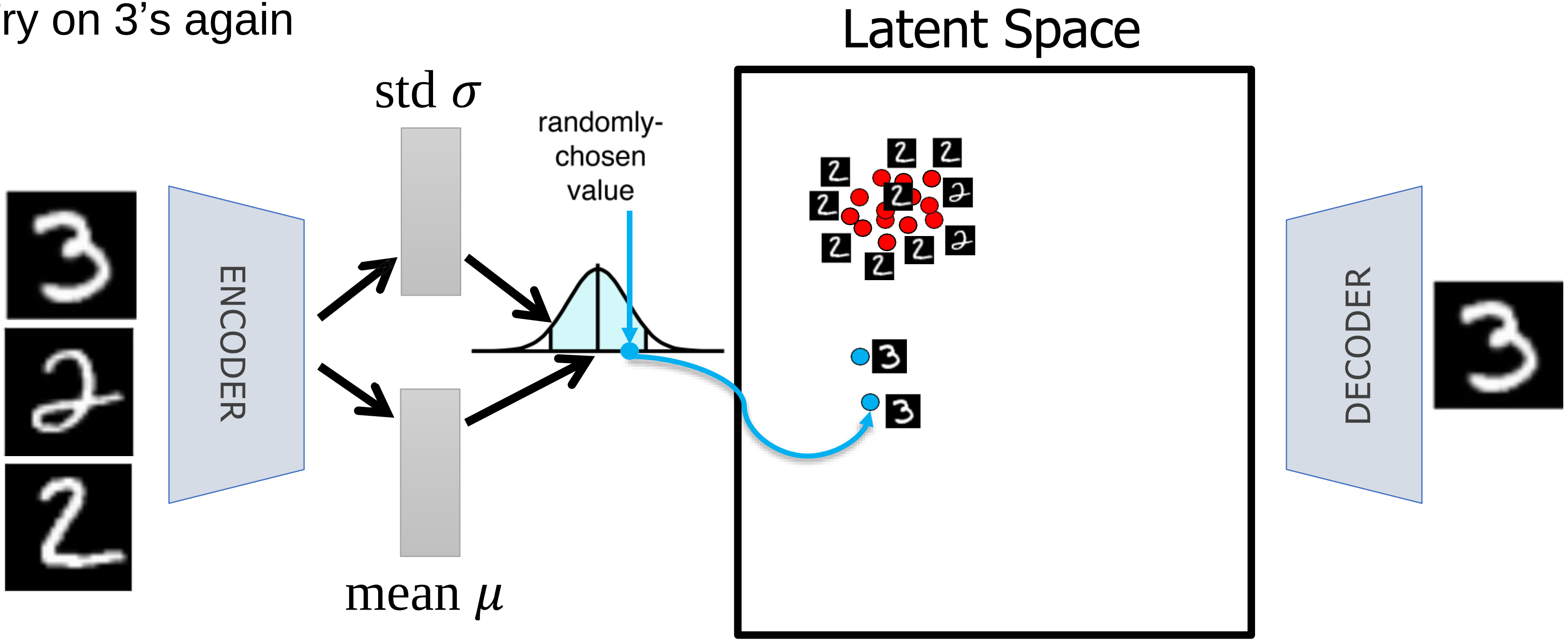


LATENT SPACE OF VAE

Now, let's test again

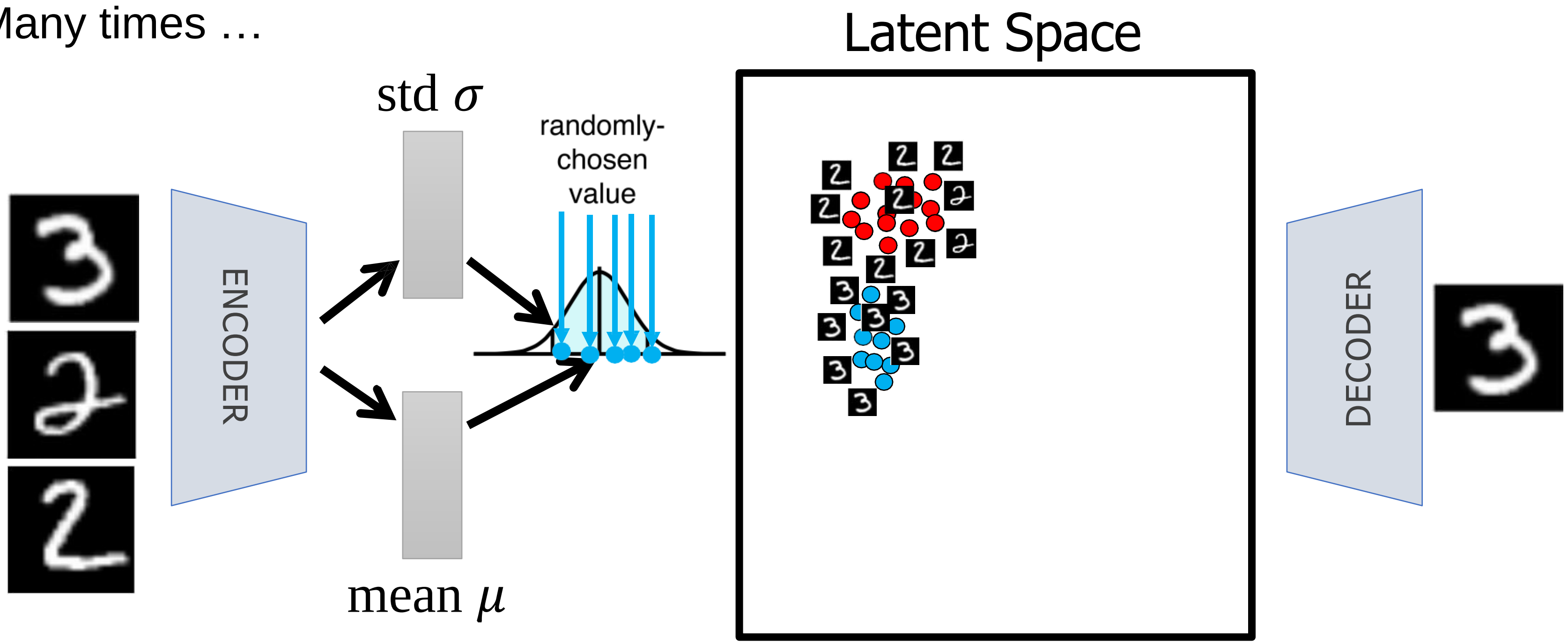


Try on 3's again



LATENT SPACE OF VAE

Many times …



VANILLA VAE

```
import torch.nn as nn

class VAE(nn.Module):
    def __init__(self, D, M):
        super(LinearVAE, self).__init__()
        self.D = D
        self.M = M

        self.enc1 = nn.Linear(in_features=self.D, out_features=300)
        self.enc2 = nn.Linear(in_features=300, out_features=self.M*2)

        self.dec1 = nn.Linear(in_features=self.M, out_features=300)
        self.dec2 = nn.Linear(in_features=300, out_features=self.D)

    def reparameterize(self, mu, log_std):
        std = torch.exp(log_var)
        eps = torch.randn_like(std)
        Z = mu + (eps * std)
        return z
```

VANILLA VAE

```
def forward(self, x):
    # encoder
    x = nn.functional.relu(self.enc1(x))
    x = self.enc2(x).view(-1, 2, self.M)

    # get mean and log-std
    mu = x[:, 0, :]
    log_var = x[:, 1, :]

    # reparameterization
    z = self.reparameterize(mu, log_std)

    # decoder
    x_hat = nn.functional.relu(self.dec1(z))
    x_hat = self.dec2(x)
    return x_hat, mu, log_std
```

VANILLA VAE

```python
def elbo(self, x, x_hat, z, mu, log_std):
    # reconstruction error
    RE = nn.loss.mse(x, x_hat)

    # kl-regularization
    # We assume here that log_normal is implemented
    KL = log_normal(z, mu, log_std) - log_normal(z, 0, 1)

    # REMEMBER! We maximize ELBO, but optimizers minimize.
    # Therefore, we need to take the negative sign!
    return -(RE - KL)
```

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})$$

Weak **decoders** $\rightarrow$ bad generations/reconstructions

Weak **encoders** $\rightarrow$ bad latent representation, *posterior collapse* (variational posterior = prior).

Weak **marginals** $\rightarrow$ bad generations

Variational **posteriors** $\rightarrow$ what family of distributions?

VARIATIONAL AUTO-ENCODERS

Advantages

✓ Non-linear transformations.

✓ Stable training.

✓ Allows compression.

✓ Allows to generation.

✓ The likelihood could be approximated.

Disadvantages

- No analytical solutions.
- No exact likelihood.
- Potential mismatch between true posterior and variational posterior
- Blurry images

VARIATIONAL AUTO-ENCODERS

Advantages

✓ Non-linear transformations.

✓ Stable training.

✓ Allows compression.

✓ Allows to generation.

✓ The likelihood could be approximated.

Disadvantages

- No analytical solutions.
- No exact likelihood.
- Potential mismatch between true posterior and variational posterior
- ~~Blurry images~~

Thank you!

VRIJE UNIVERSITEIT AMSTERDAM