Lecture 7: Unsupervised Representation Learning and Generative Models (cont.)

Shujian Yu

Deep Learning 2023

dlvu.github.io

VRIJE UNIVERSITEIT AMSTERDAM

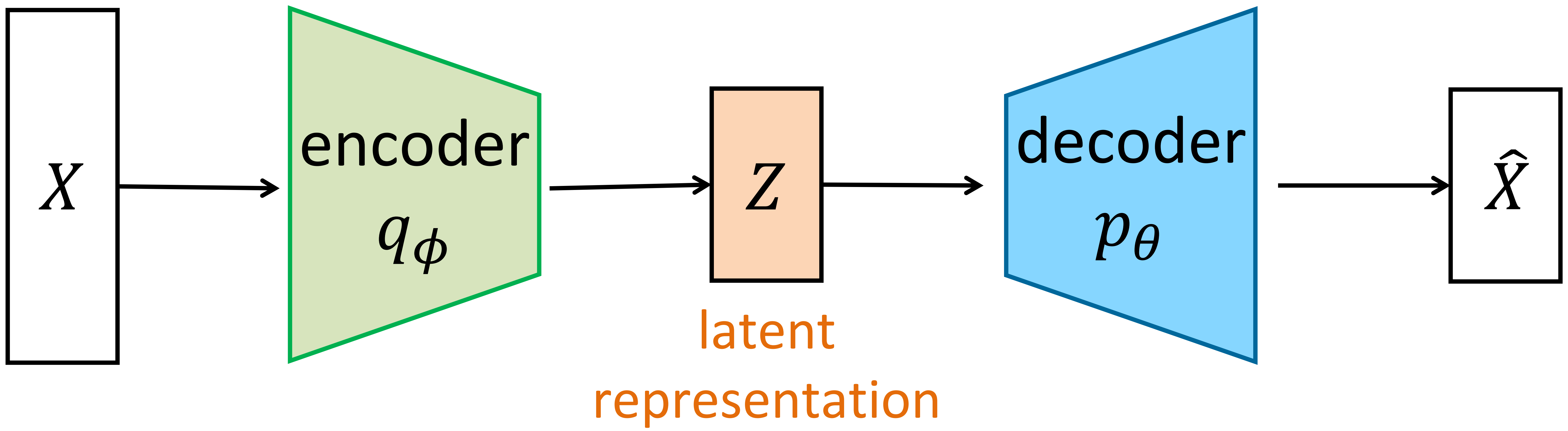
THE PLAN

part 1: VAE implementation

part 2: KL divergence and maximum mean discrepancy

part 3: MMD-VAE and β-VAE

RECAP


X
encoder
q_ϕ
Z
latent representation
decoder
p_θ
$\hat{X}$

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

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

Variational posterior

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

$$= \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z}) + \log \frac{p_\lambda(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \right] d\boldsymbol{z}$$

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

Evidence Lower BOund (ELBO)

$$
\begin{aligned}
\ln p_\theta(\boldsymbol{x}) &= \log \int p_\theta(\boldsymbol{x}|\boldsymbol{z}) p_\lambda(\boldsymbol{z}) d\boldsymbol{z} \\
&= \log \int \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})} p_\theta(\boldsymbol{x}|\boldsymbol{z}) p_\lambda(\boldsymbol{z}) d\boldsymbol{z} \\
&\geq \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z}) p_\lambda(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} \\
&= \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z}) + \log \frac{p_\lambda(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \right] d\boldsymbol{z} \\
&= \underbrace{\mathbb{E}_{\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} [\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]}_{\text{Reconstruction error (RE)}} - \underbrace{D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}); p_\lambda(\boldsymbol{z}) \right)}_{\text{Regularization (KL)}}
\end{aligned}
$$

Reconstruction error (RE)

Regularization (KL)

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

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

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

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

$$= \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z}) + \log \frac{p_\lambda(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \right] d\boldsymbol{z}$$

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

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) \sim \mathcal{N}(f_\theta(\boldsymbol{z}), \sigma \boldsymbol{I})$$

$$\mathbb{E}_{\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} [\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]$$

$$= \mathbb{E}_{\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\log \frac{1}{\sqrt{2\pi}\sigma} - \frac{\|\boldsymbol{x} - f_\theta(\boldsymbol{z})\|_2^2}{2\sigma^2} \right]$$

Reconstruction error (RE) **Regularization (KL)**

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

decoder

encoder

marginal (prior)

= Variational Auto-Encoder

VU

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

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

***evidence lower bound* (ELBO)**

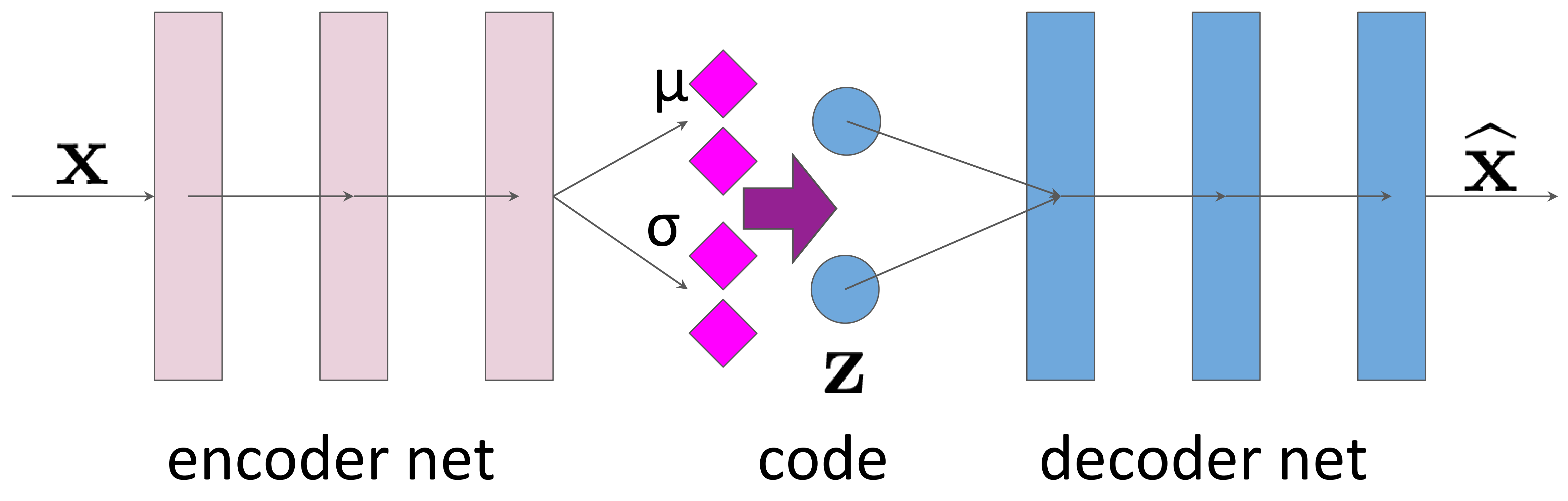
≥ 0

VU

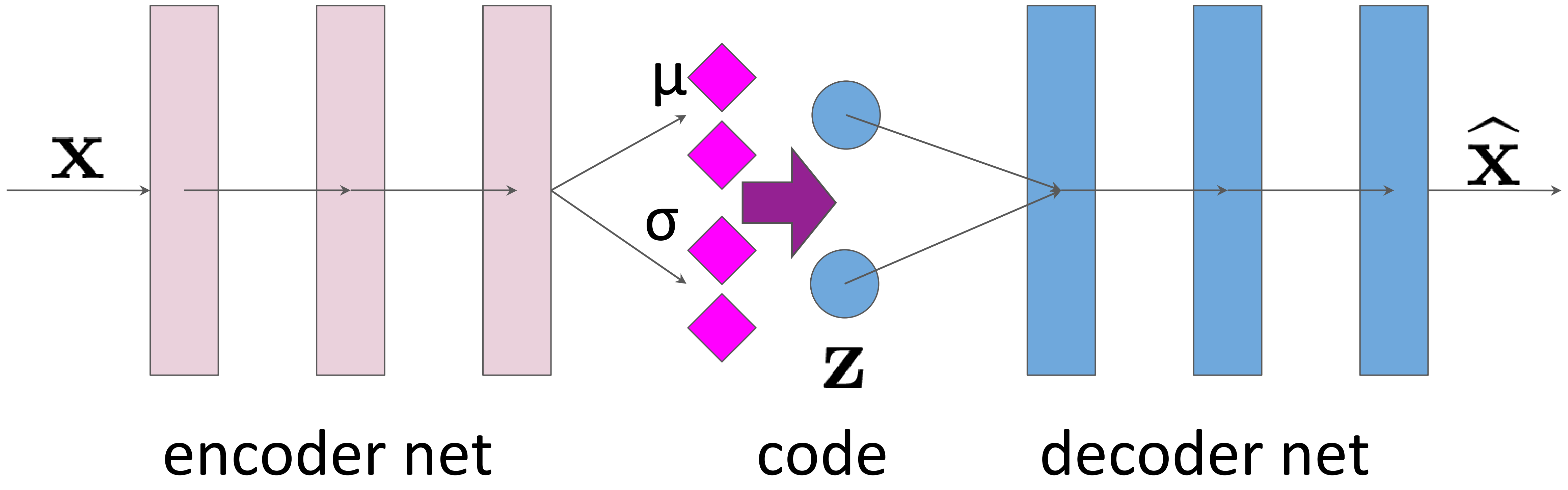
PART ONE: VAE IMPLEMENTATION

VARIATIONAL AUTO-ENCODERS

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



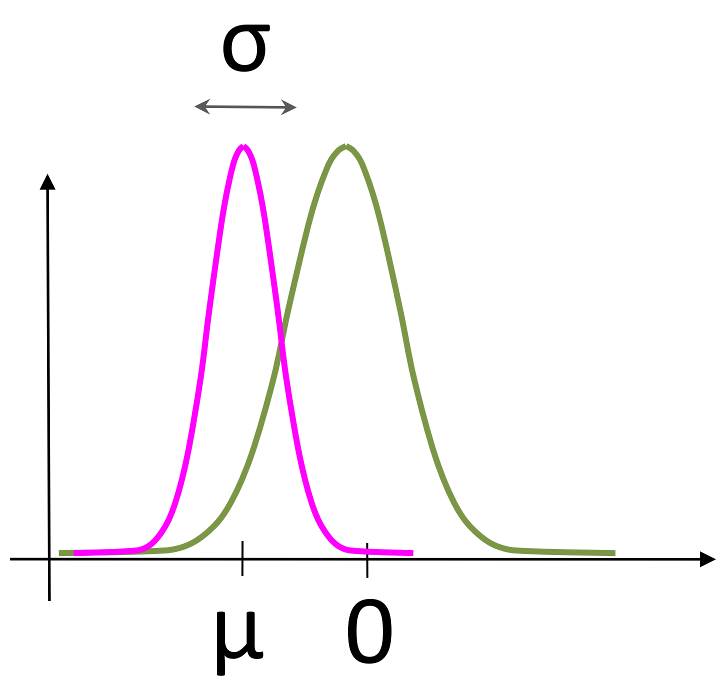
Variational posterior (**encoder**) and the likelihood function (**decoder**) are parameterized by neural networks.



Reparameterization trick:

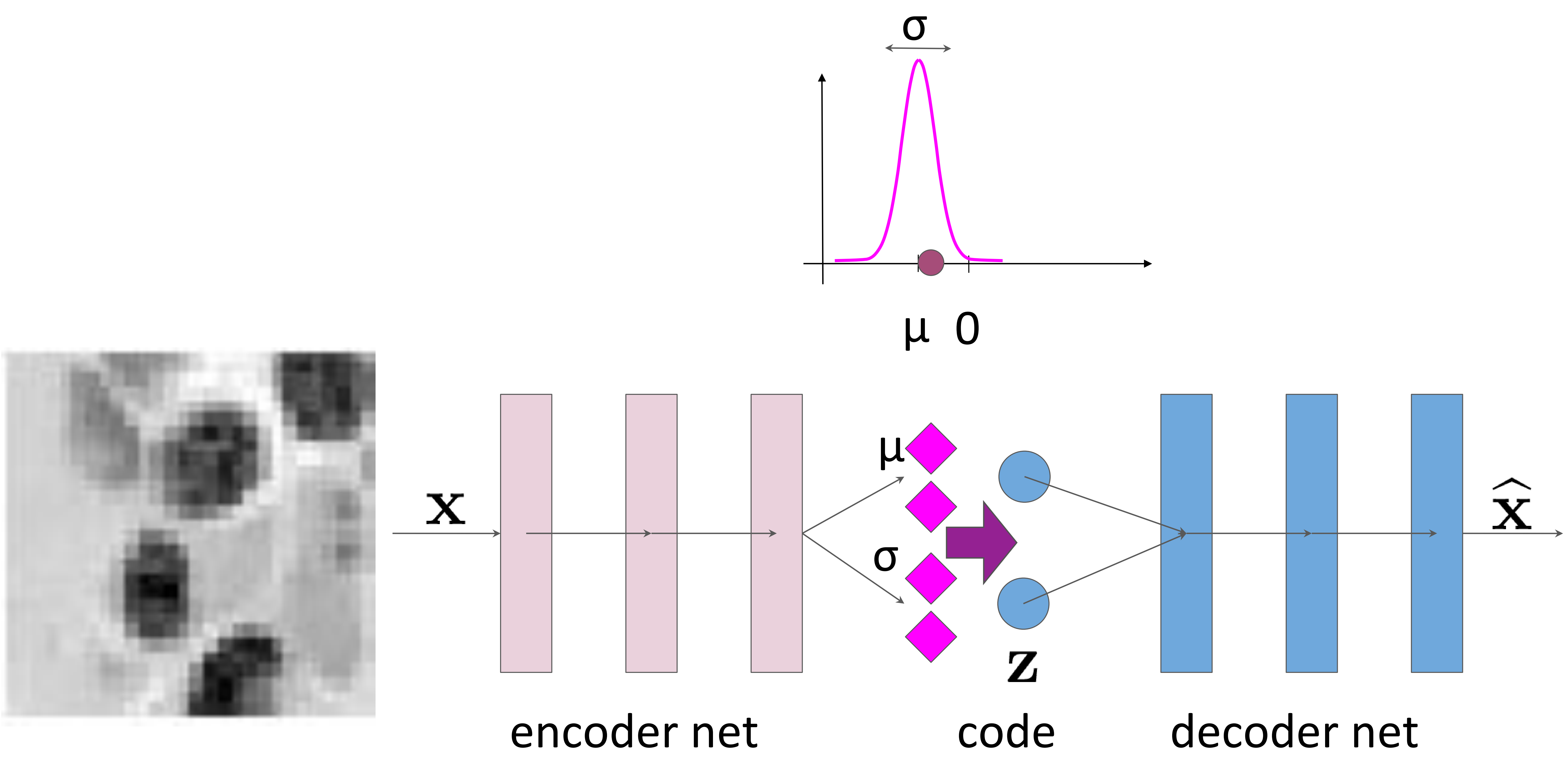
$$z \sim \mathcal{N}(\mu, \sigma)$$

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



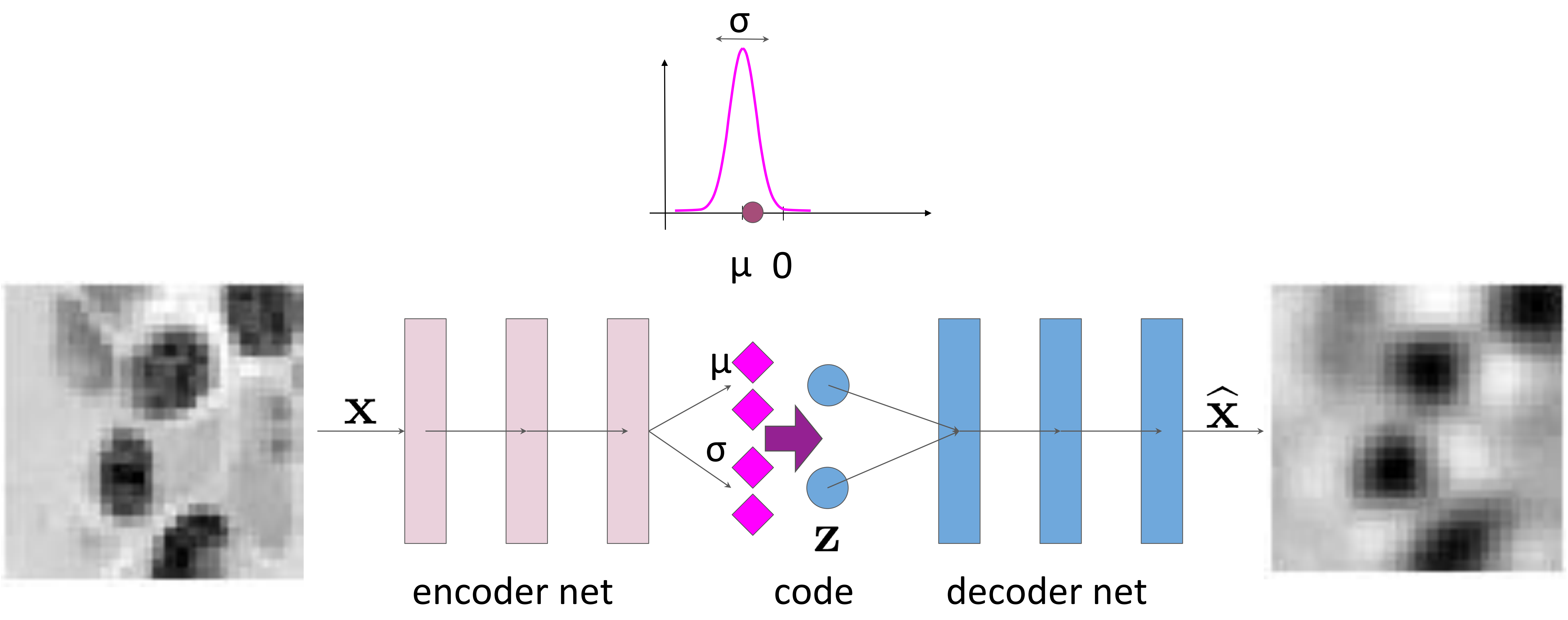
VAE copies input to output through a **bottleneck**.

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



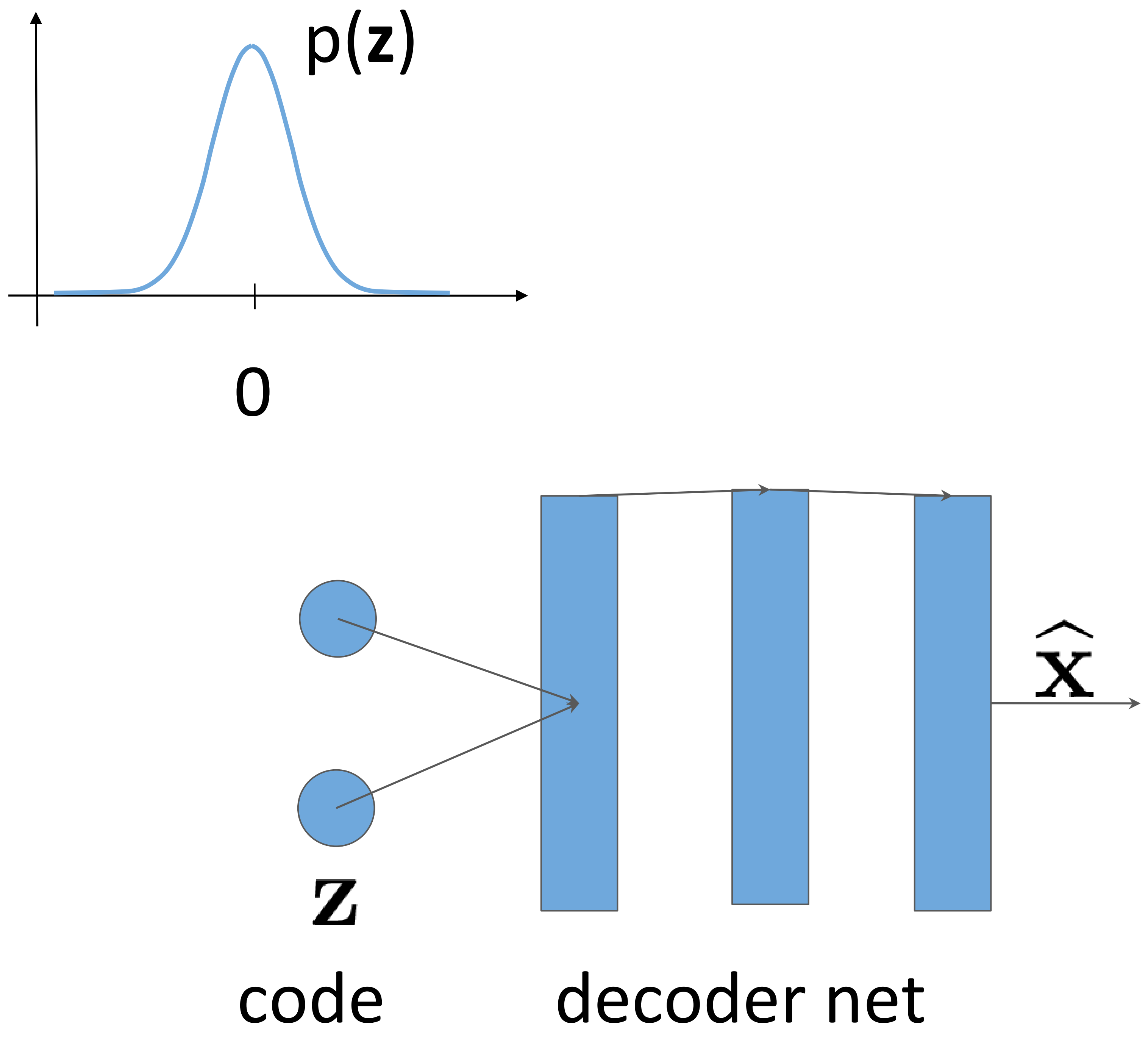
VAE copies input to output through a **bottleneck**.

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



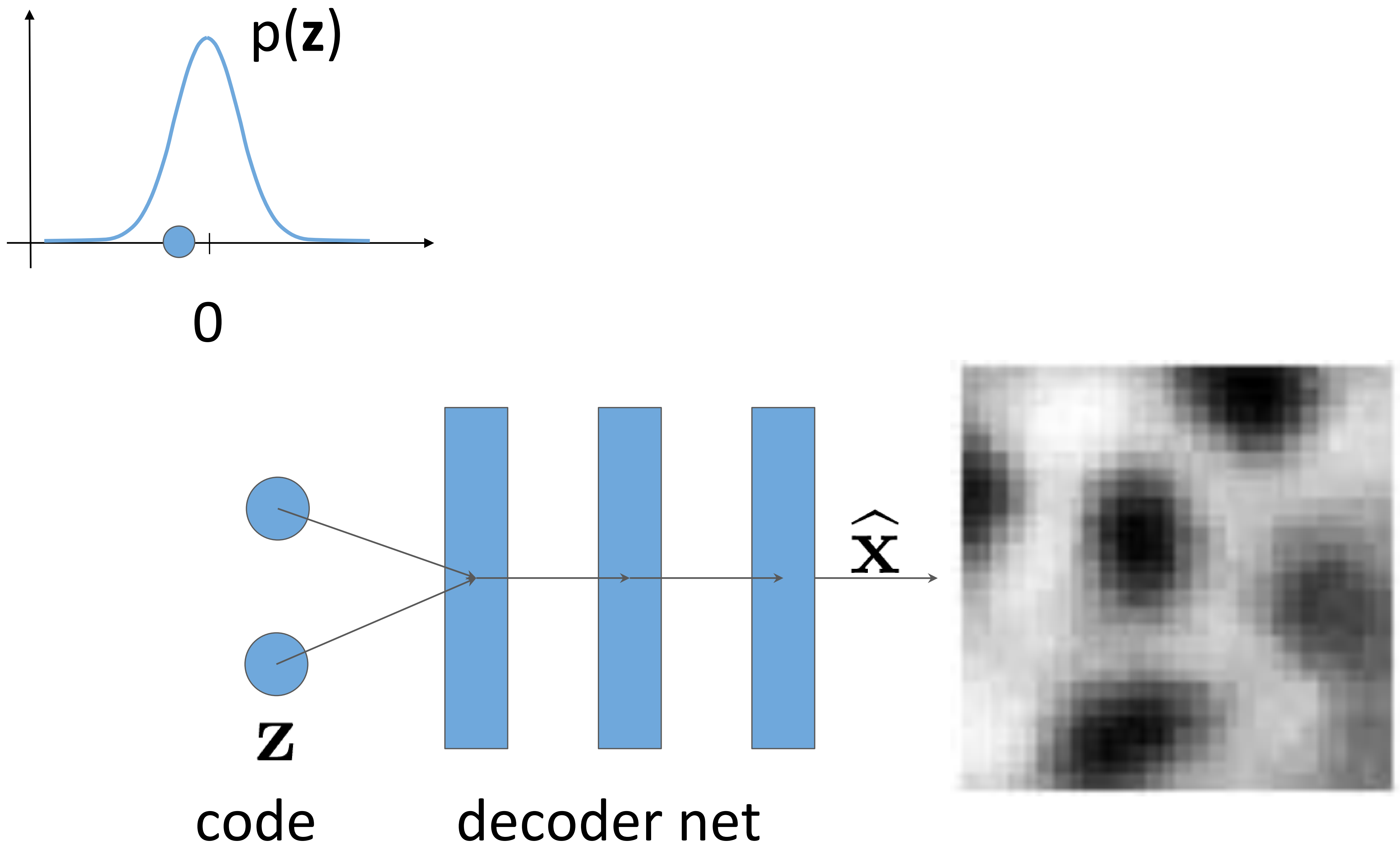
VAE has a **marginal** on the latent code.

VAE can **generate** new data.



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

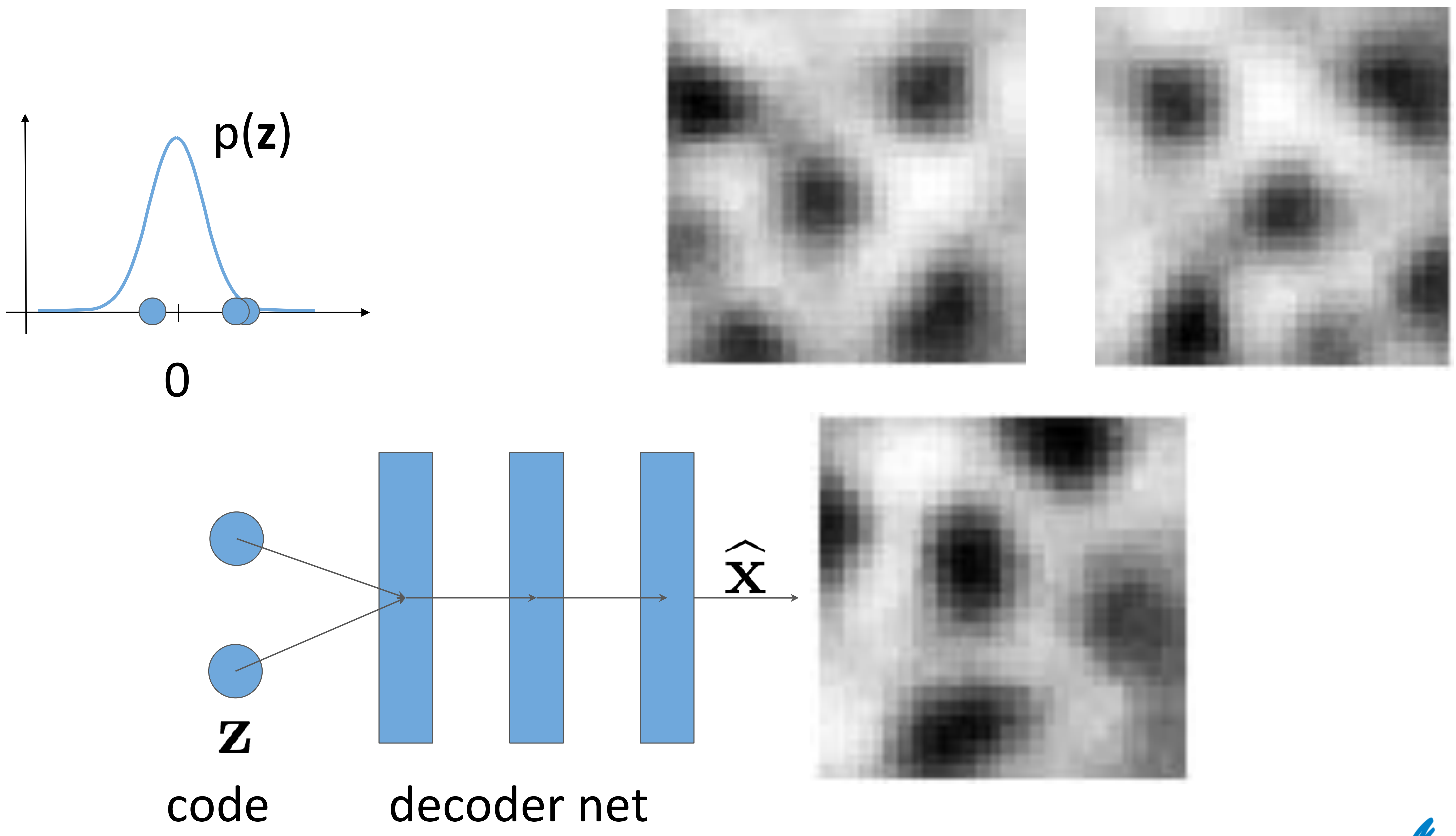
VAE can **generate** new data.



VARIATIONAL AUTO-ENCODERS

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

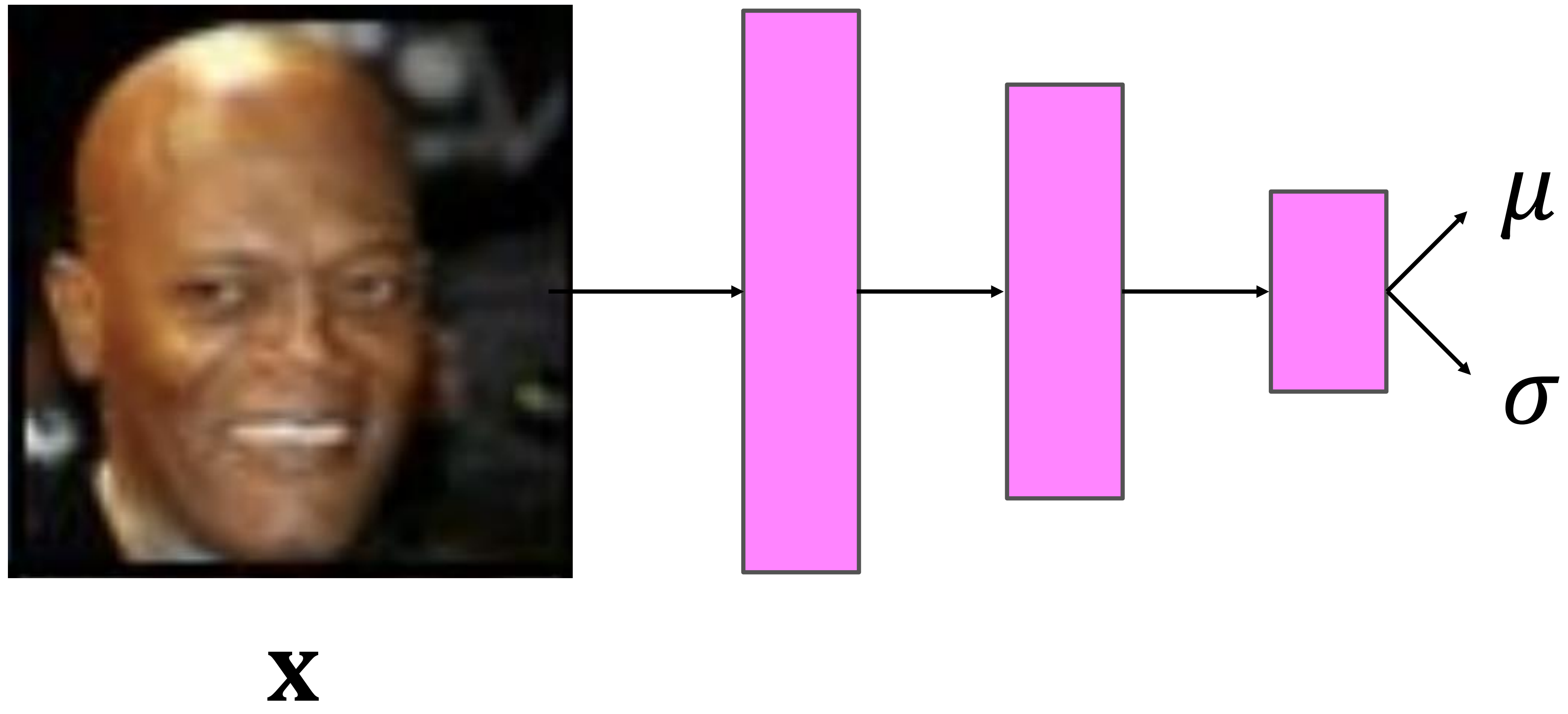
VAE can **generate** new data.



VANILLA VAE

$\mathbf{x}$

VANILLA VAE

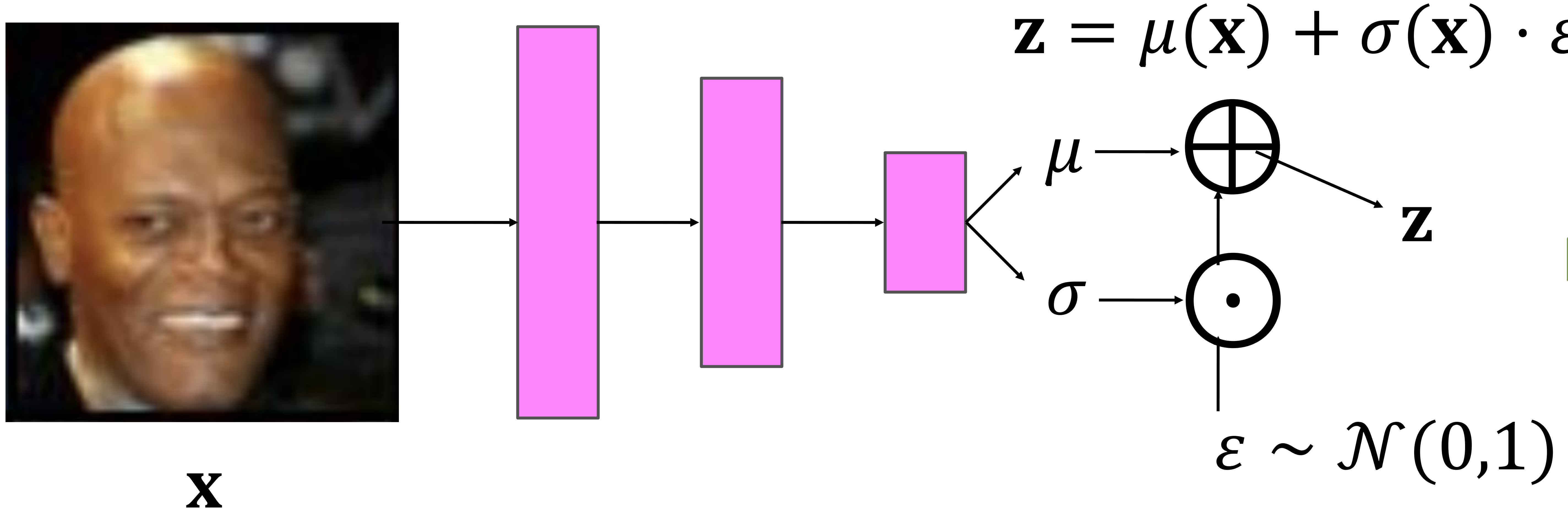


Example architecture for the encoder:

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

VANILLA VAE

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



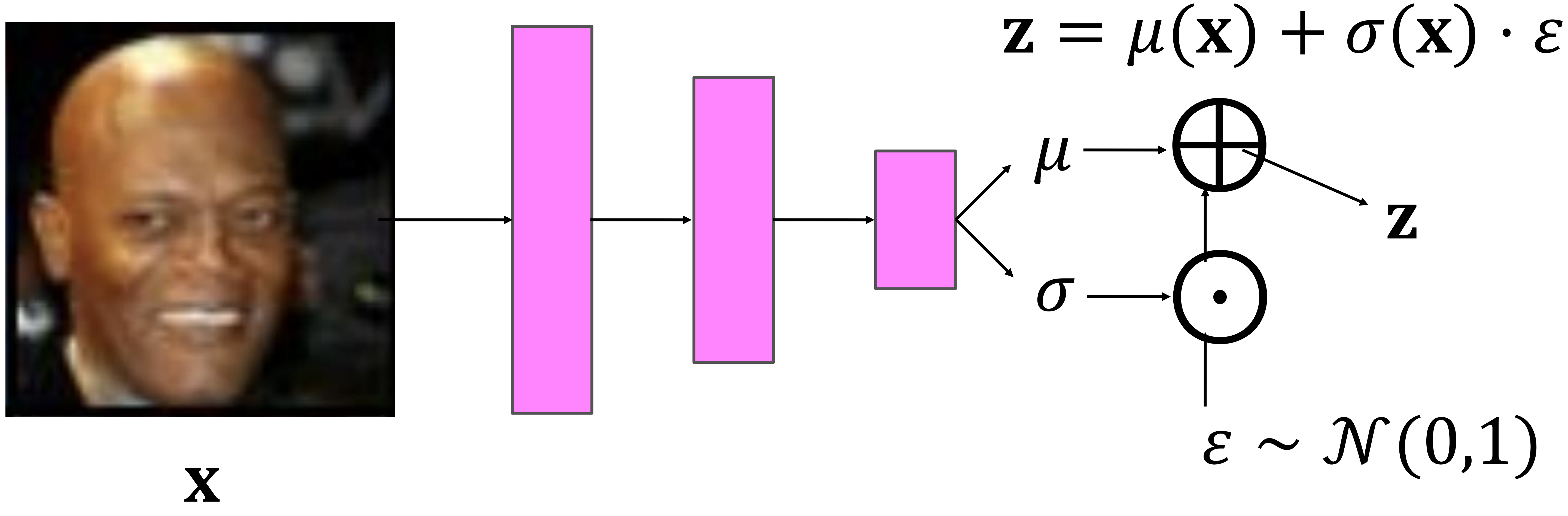
Reparameterization trick!

Example architecture for the encoder:

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

VANILLA VAE

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

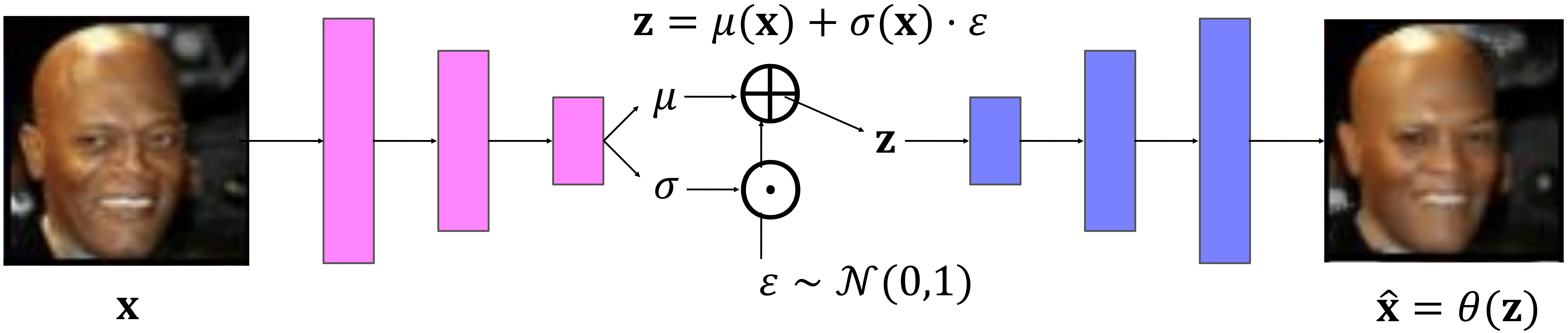


Example architecture for the encoder:

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



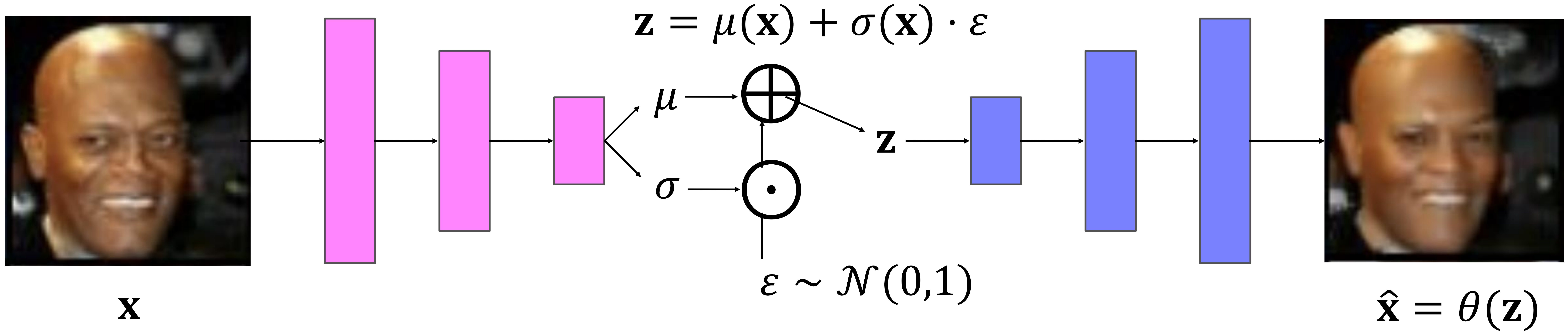
Example architecture for the encoder:

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

Example architecture for the decoder:

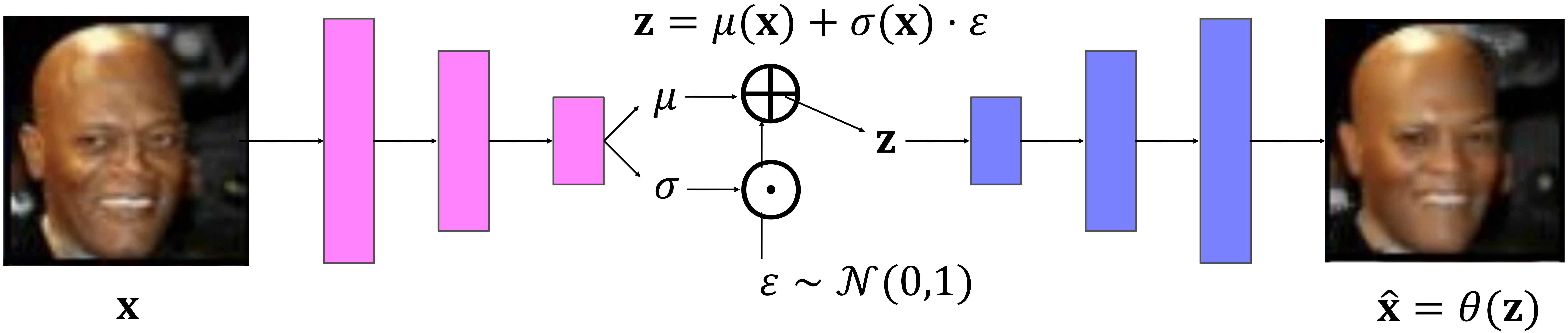
z → Linear(*M*, 300) → ReLU → Linear(300, *D*) → means

No non-linearity here!
We model means only.



We approximate expected values using a single sample:

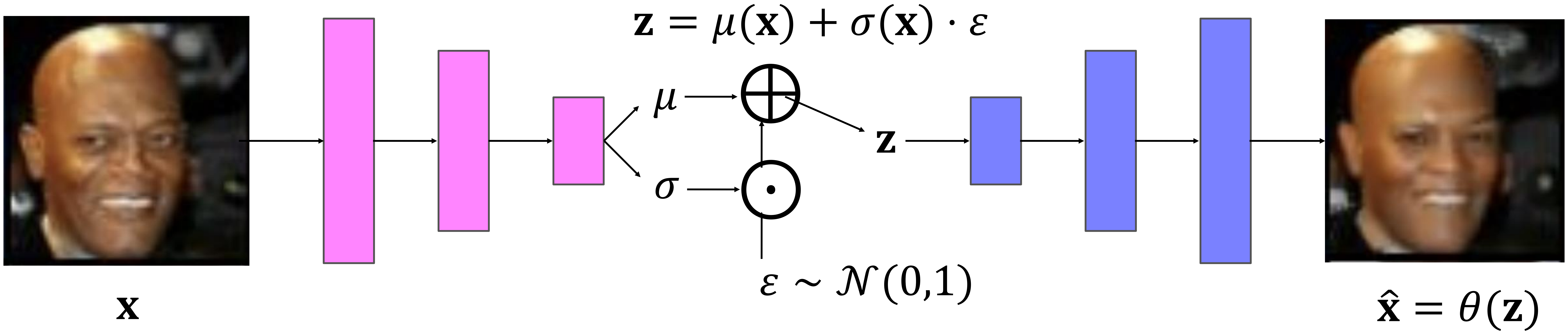
$$ELBO = \underbrace{\ln\mathcal{N}(\mathbf{x}|\theta(\mathbf{z}), 1)}_{p_\theta(\mathbf{x}|\mathbf{z})} - [\underbrace{\ln\mathcal{N}(\mathbf{z}|\mu(\mathbf{x}), \sigma^2(\mathbf{x}))}_{q_\phi(\mathbf{z}|\mathbf{x})} - \underbrace{\ln\mathcal{N}(\mathbf{z}|0,1)}_{p_\lambda(\mathbf{z})}]$$



We approximate expected values using a single sample:

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

VANILLA VAE



We approximate expected values using a single sample:

We assume a Gaussian variational posterior.

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

RE KL

We assume a standard Gaussian prior.

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_std)
        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
```

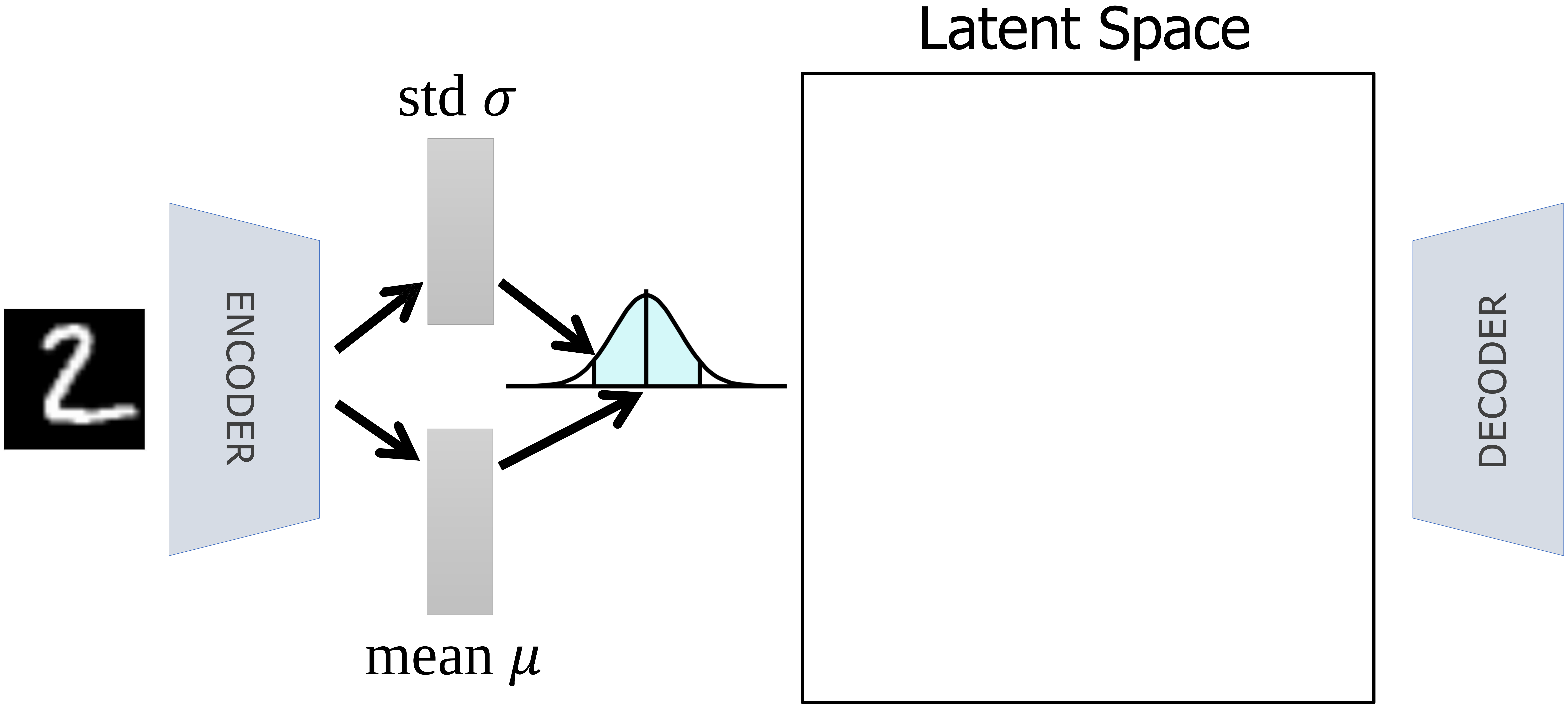
VANILLA VAE

```
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)
```

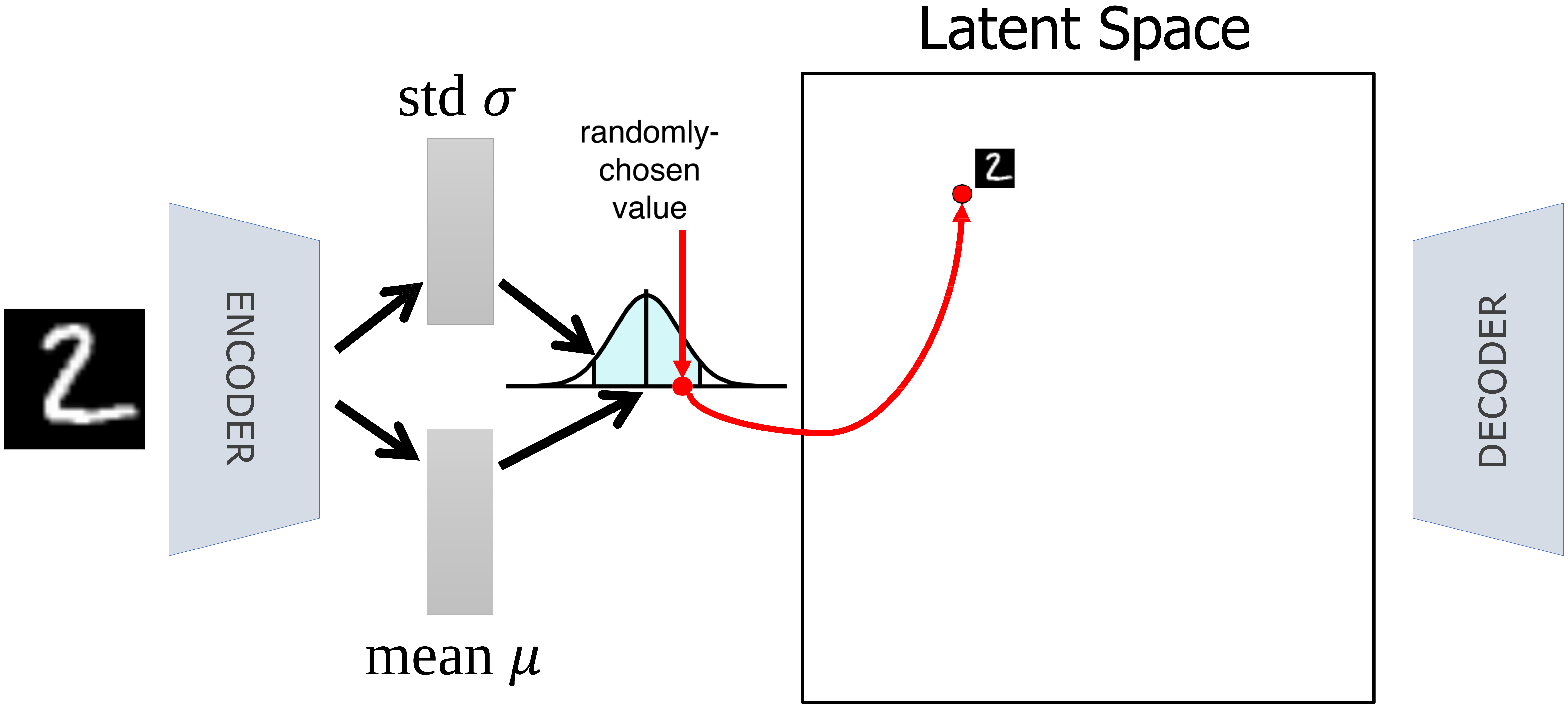
LATENT SPACE OF VAE



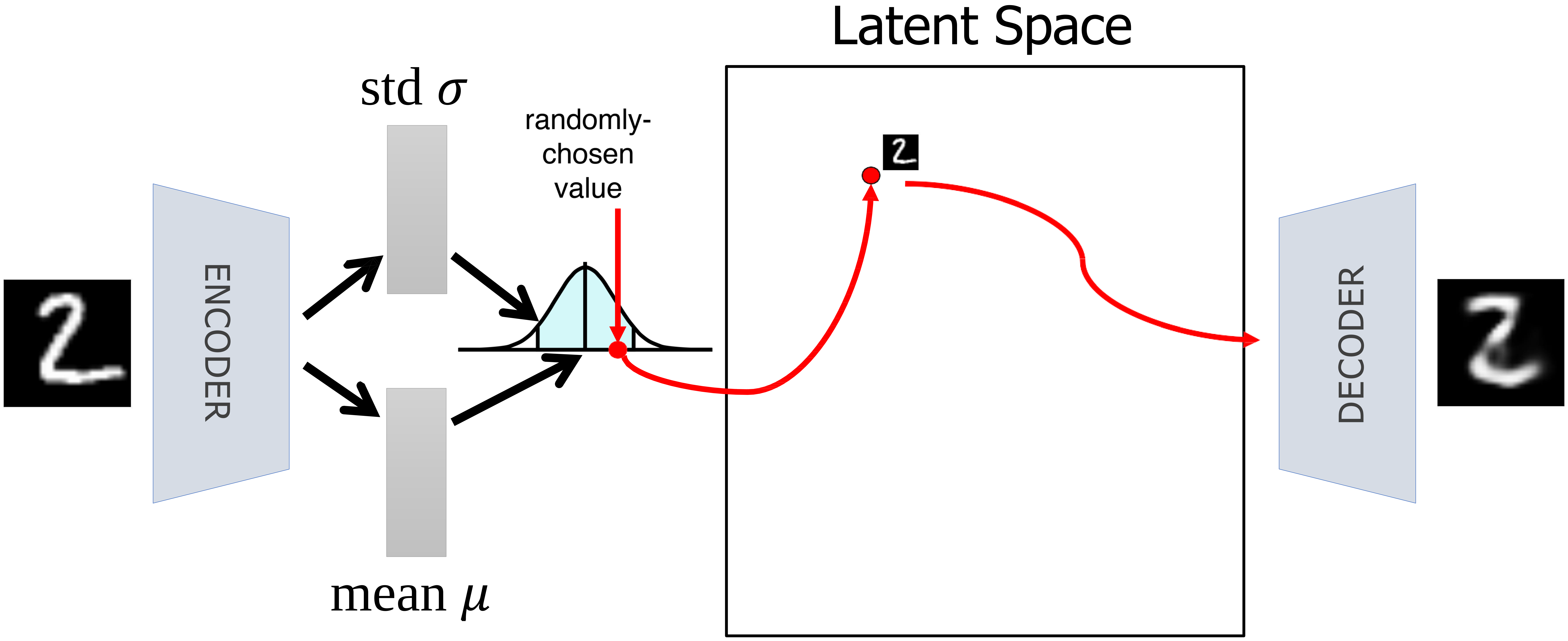
Encode the first sample (a "2") and find μ_1, σ_1

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

LATENT SPACE OF VAE

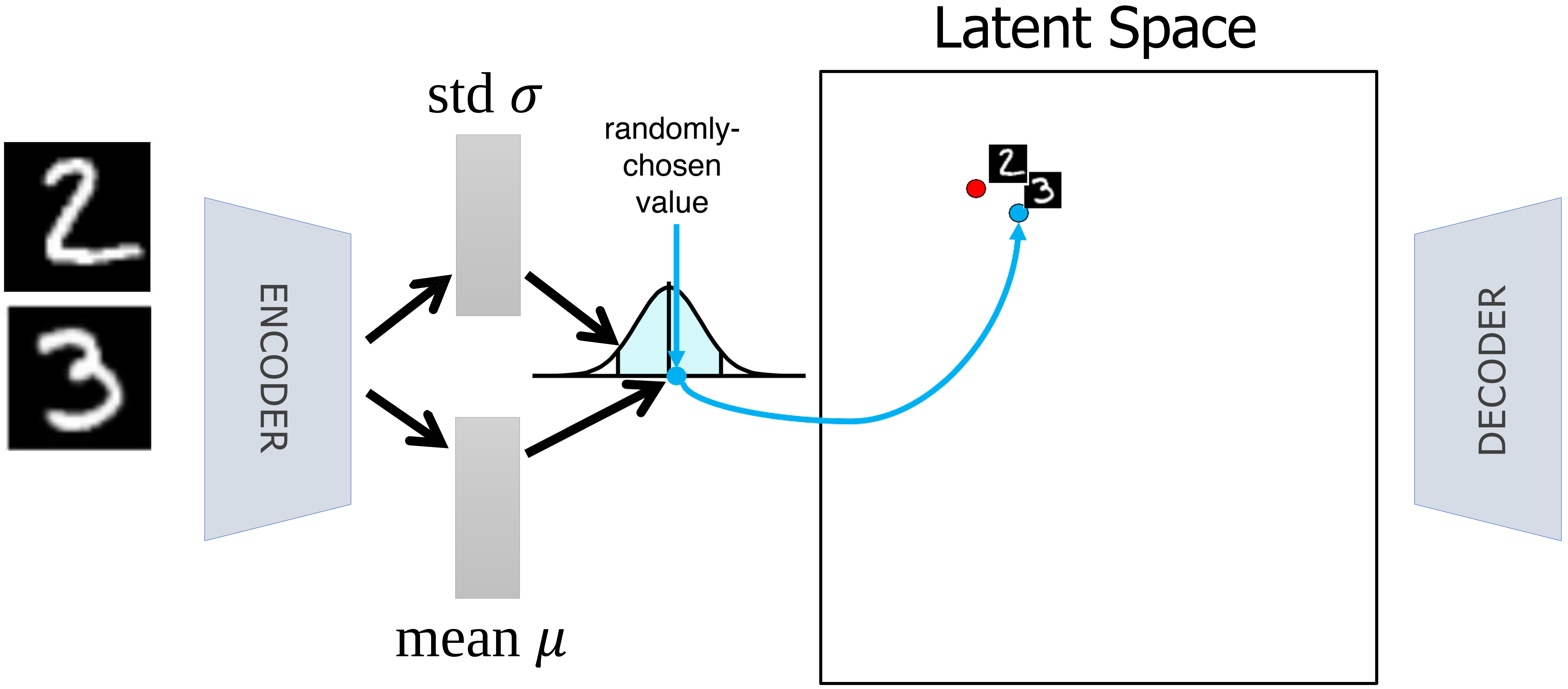


Sample $\boldsymbol{z}_1 \sim N(\mu_1, \sigma_1)$

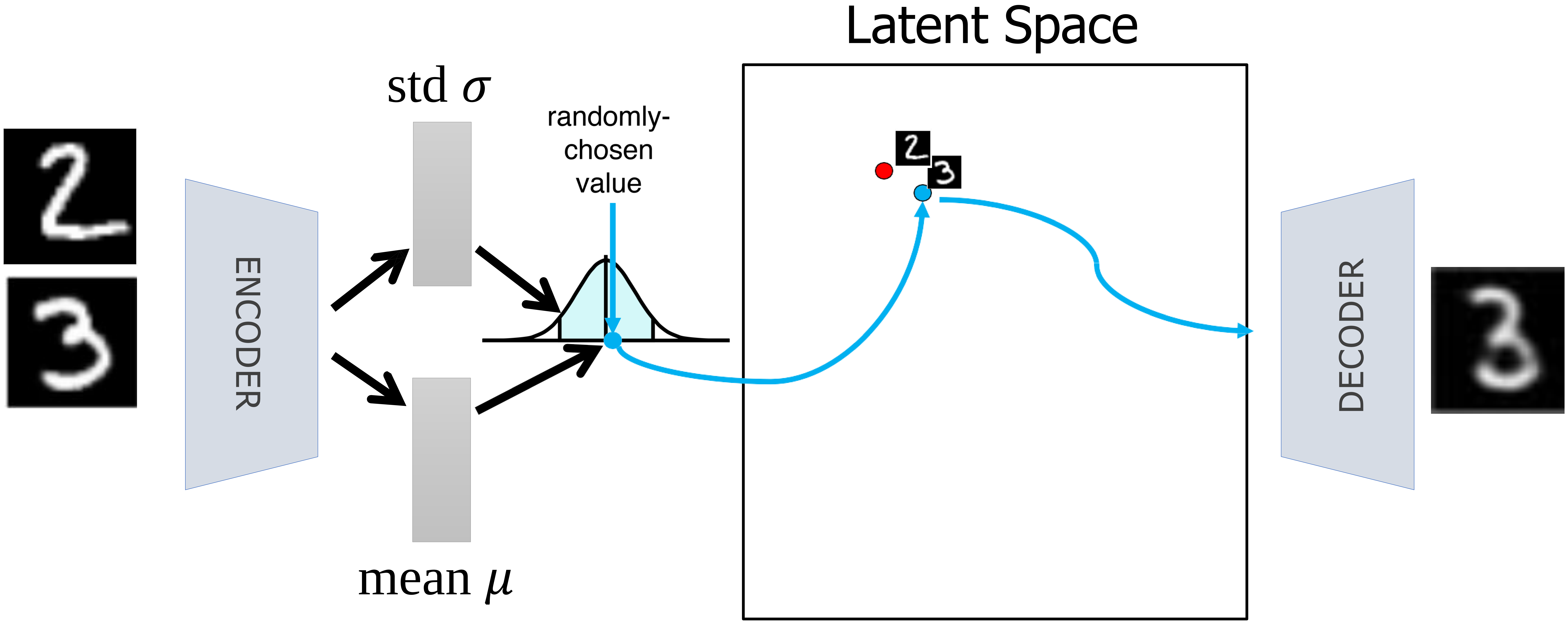


Denote to $\hat{\boldsymbol{x}}_1$

LATENT SPACE OF VAE

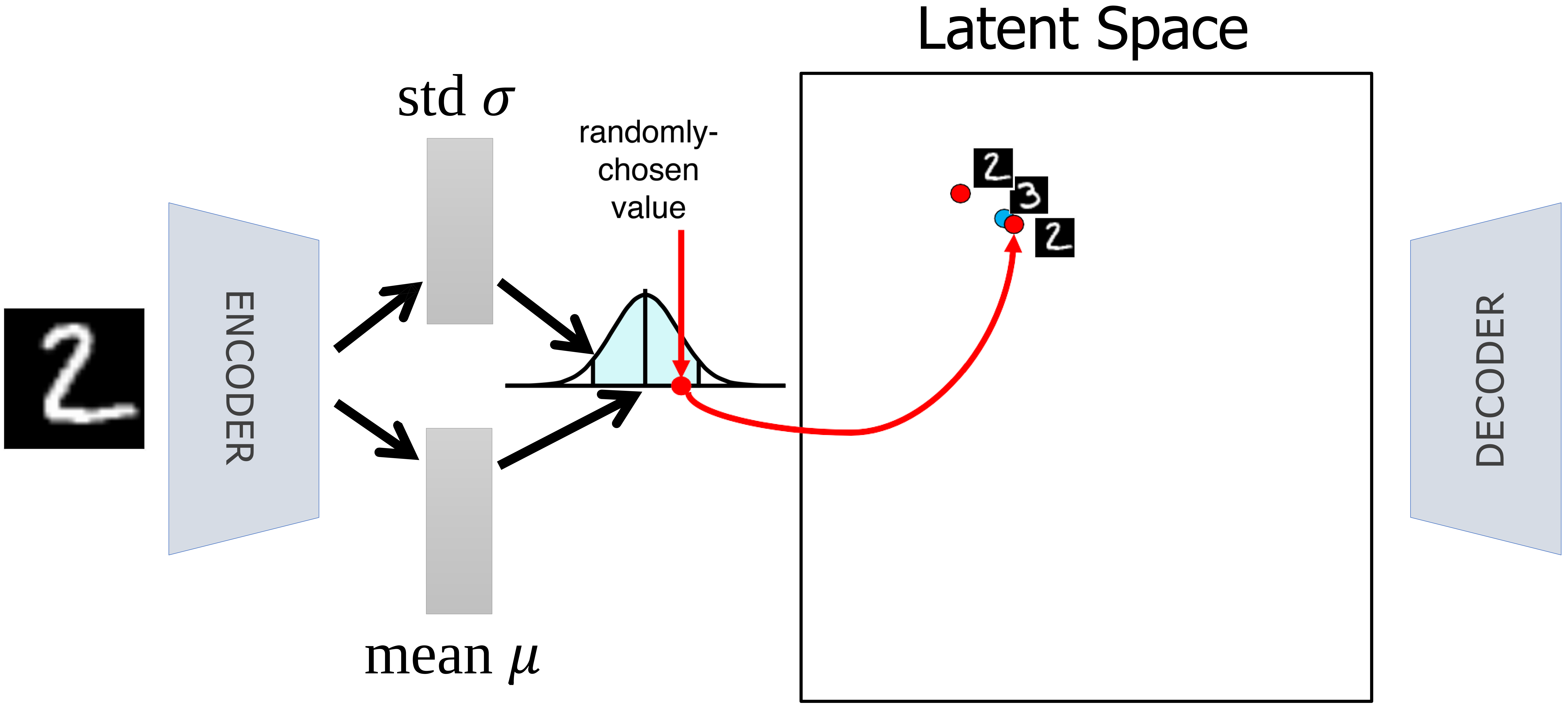


Encode the first sample (a "3") and find μ_2, σ_2, and sample $\boldsymbol{z}_2 \sim N(\mu_2, \sigma_2)$



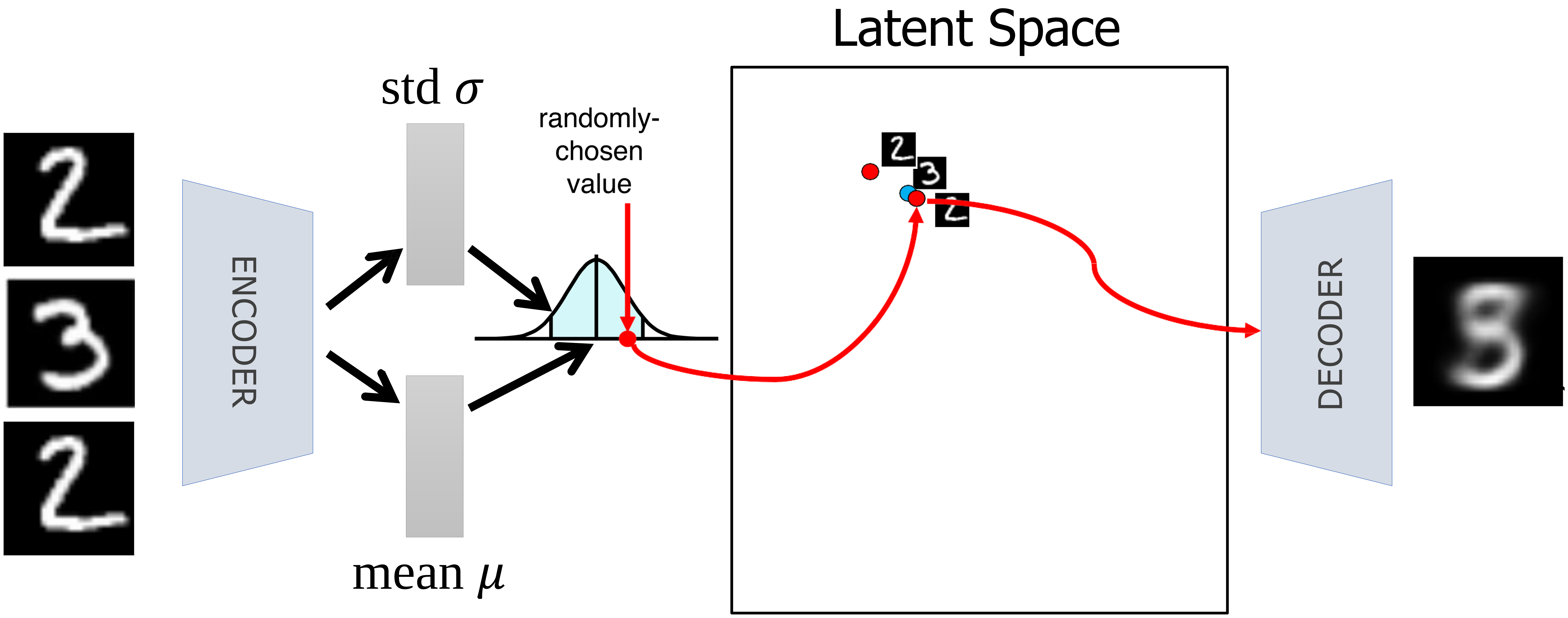
Decode to $\hat{\boldsymbol{x}}_2$

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.

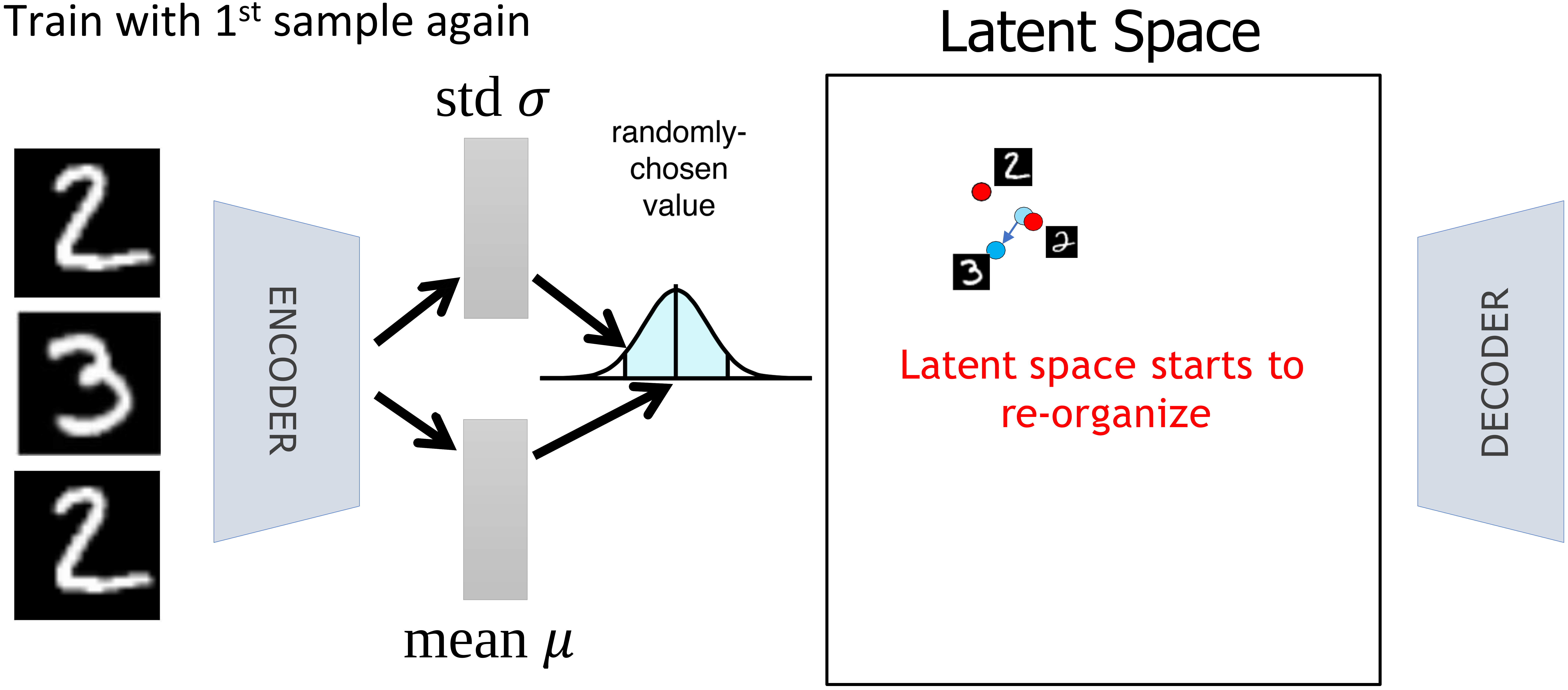
LATENT SPACE OF VAE



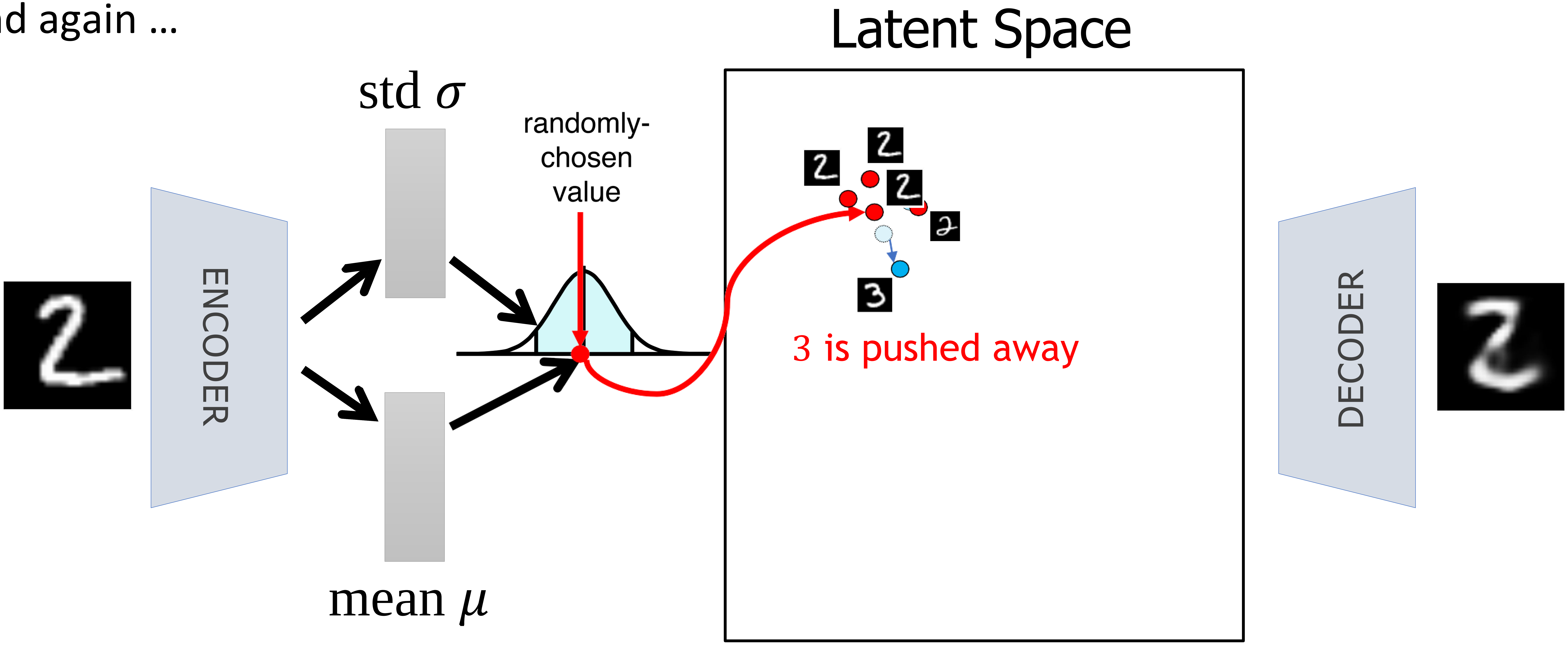
Decode to $\hat{\boldsymbol{x}}_1$. Since the decoder only knows how to map from latent space to $\hat{\boldsymbol{x}}$ space, it will return a “3”.

LATENT SPACE OF VAE

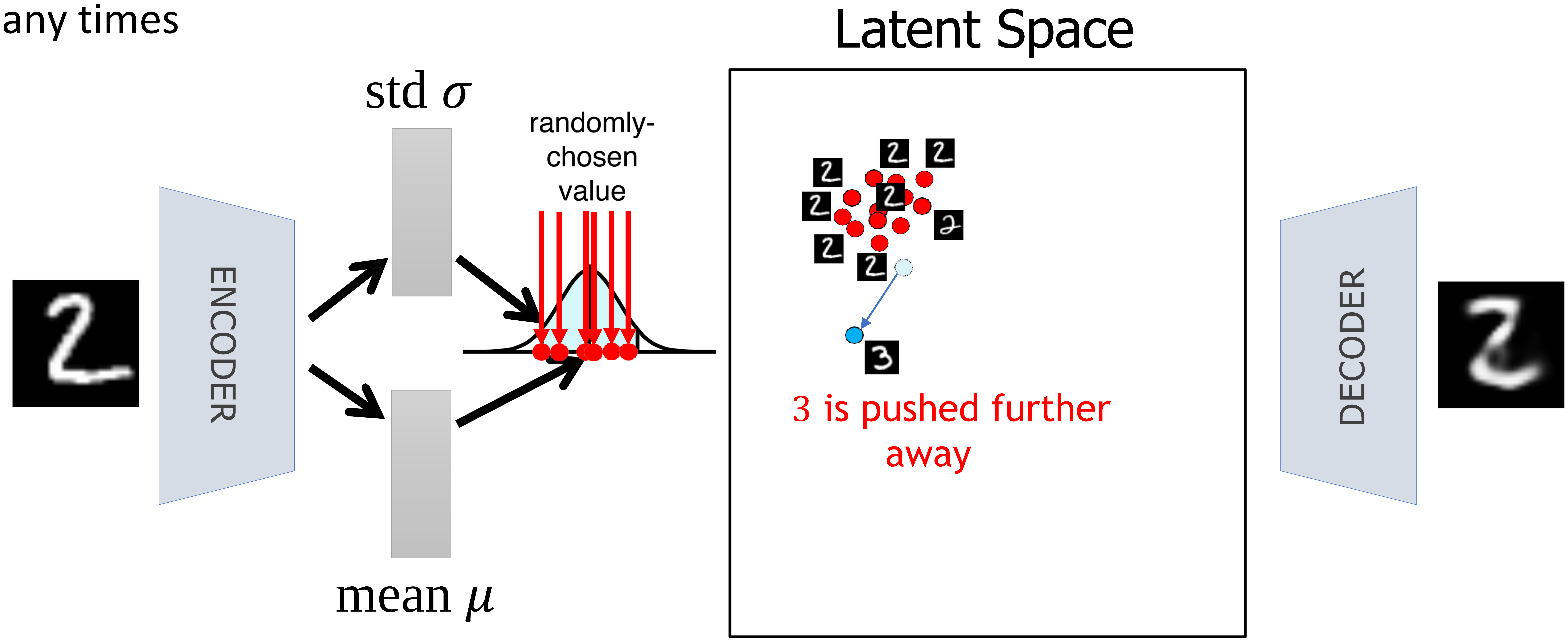
Train with 1st sample again



And again …

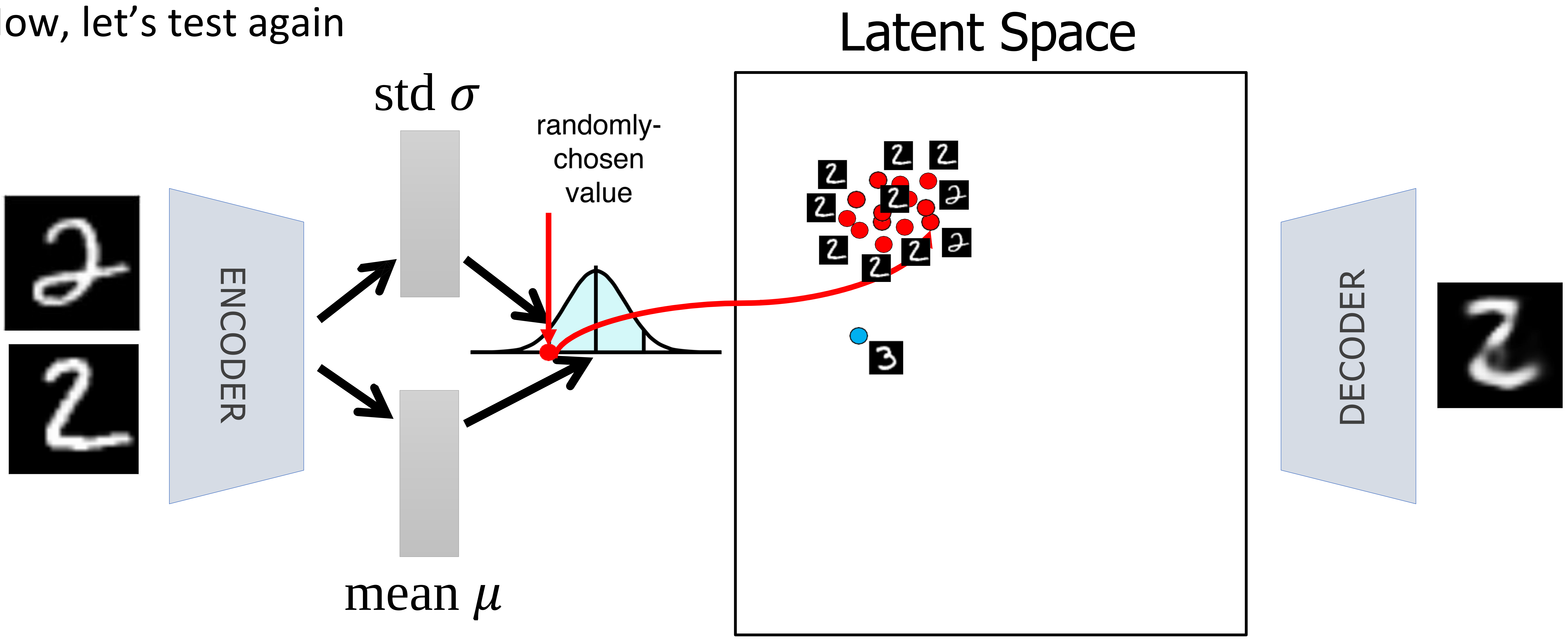

Latent Space
std σ
randomly-chosen value
ENCODER
mean μ
3 is pushed away
DECODER

Many times

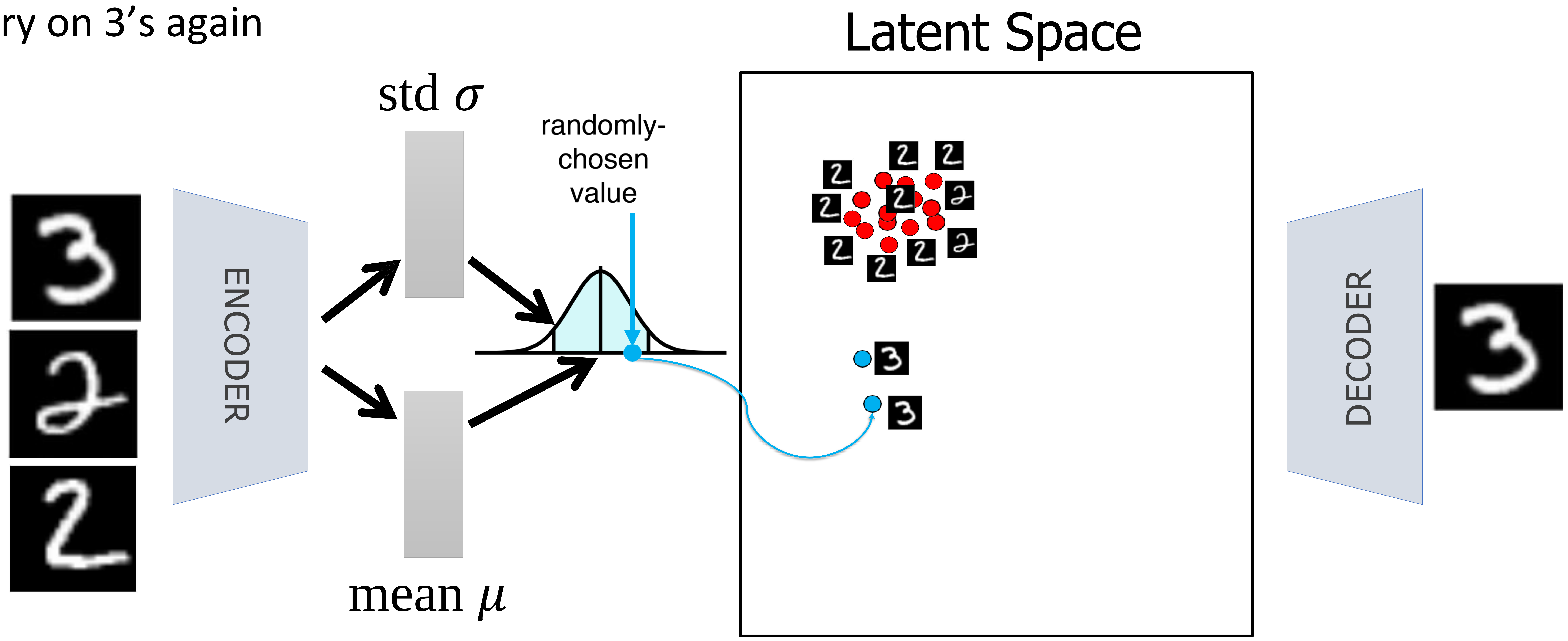


std σ
randomly-chosen value
ENCODER
mean μ
Latent Space
3 is pushed further away
DECODER

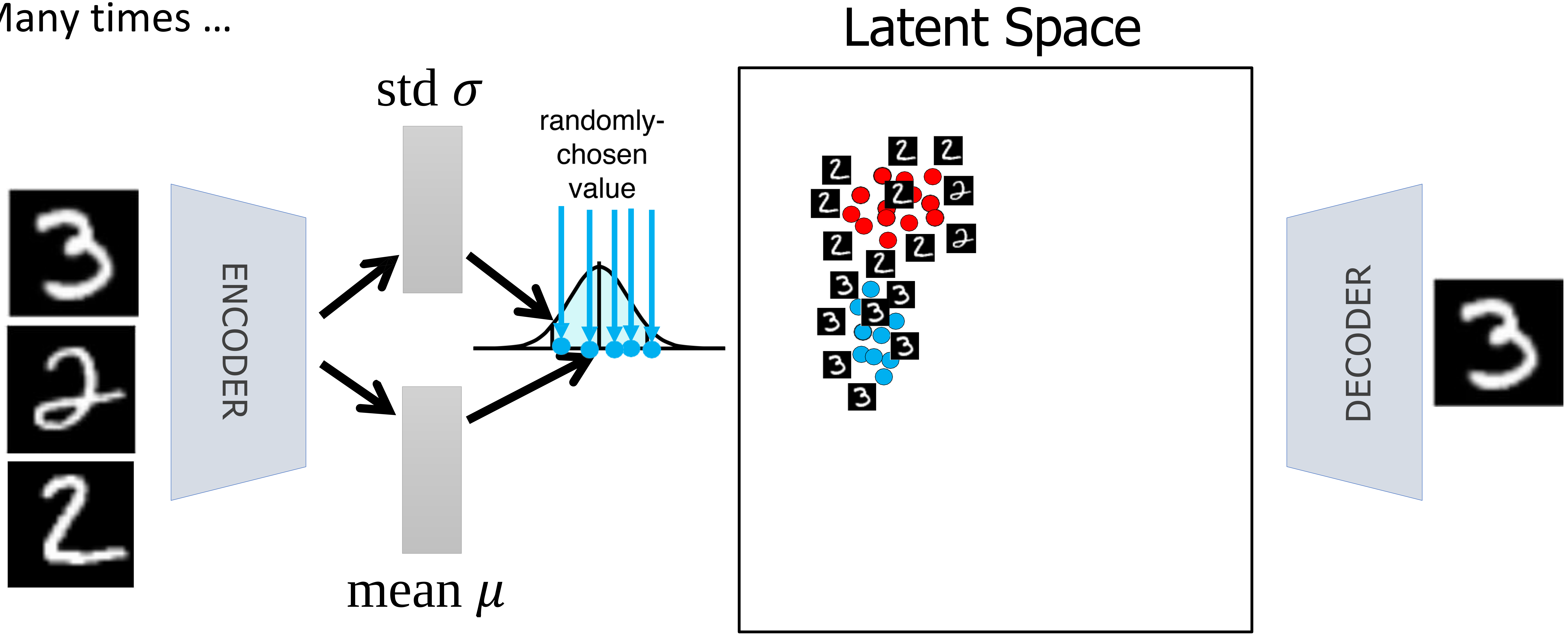
Now, let's test again



Try on 3's again



Many times …

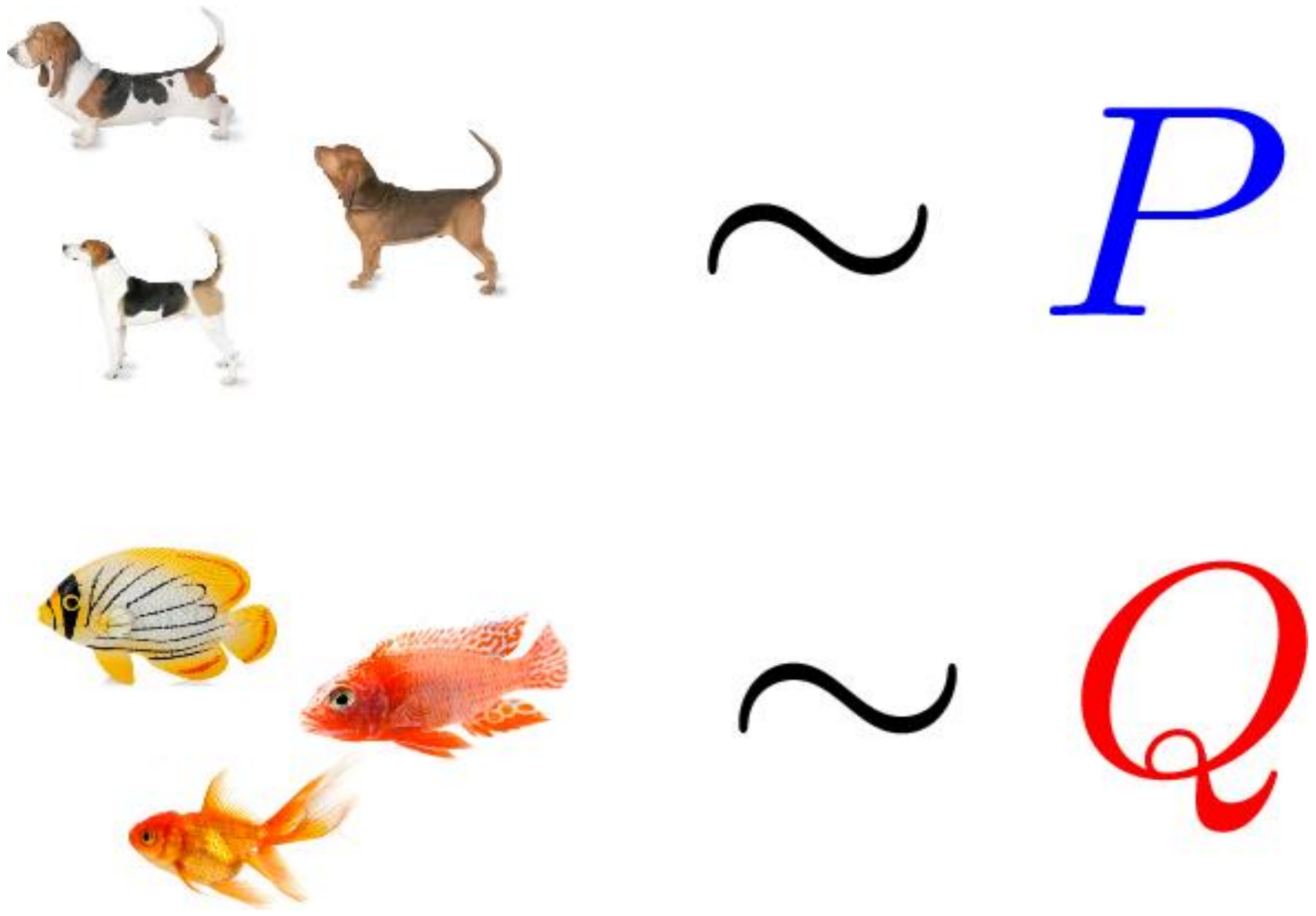

Latent Space
std σ
randomly-
chosen
value
ENCODER
mean μ
DECODER

PART TWO: KL DIVERGENCE AND MAXIMUM MEAN DISCREPANCY

How to measure the distance/divergence?

Given: samples from unknown distributions P and Q

Goal: do P and Q differ?



COMPARING TWO DISTRIBUTIONS

Given: samples from unknown distributions P and Q

Goal: do P and Q differ?

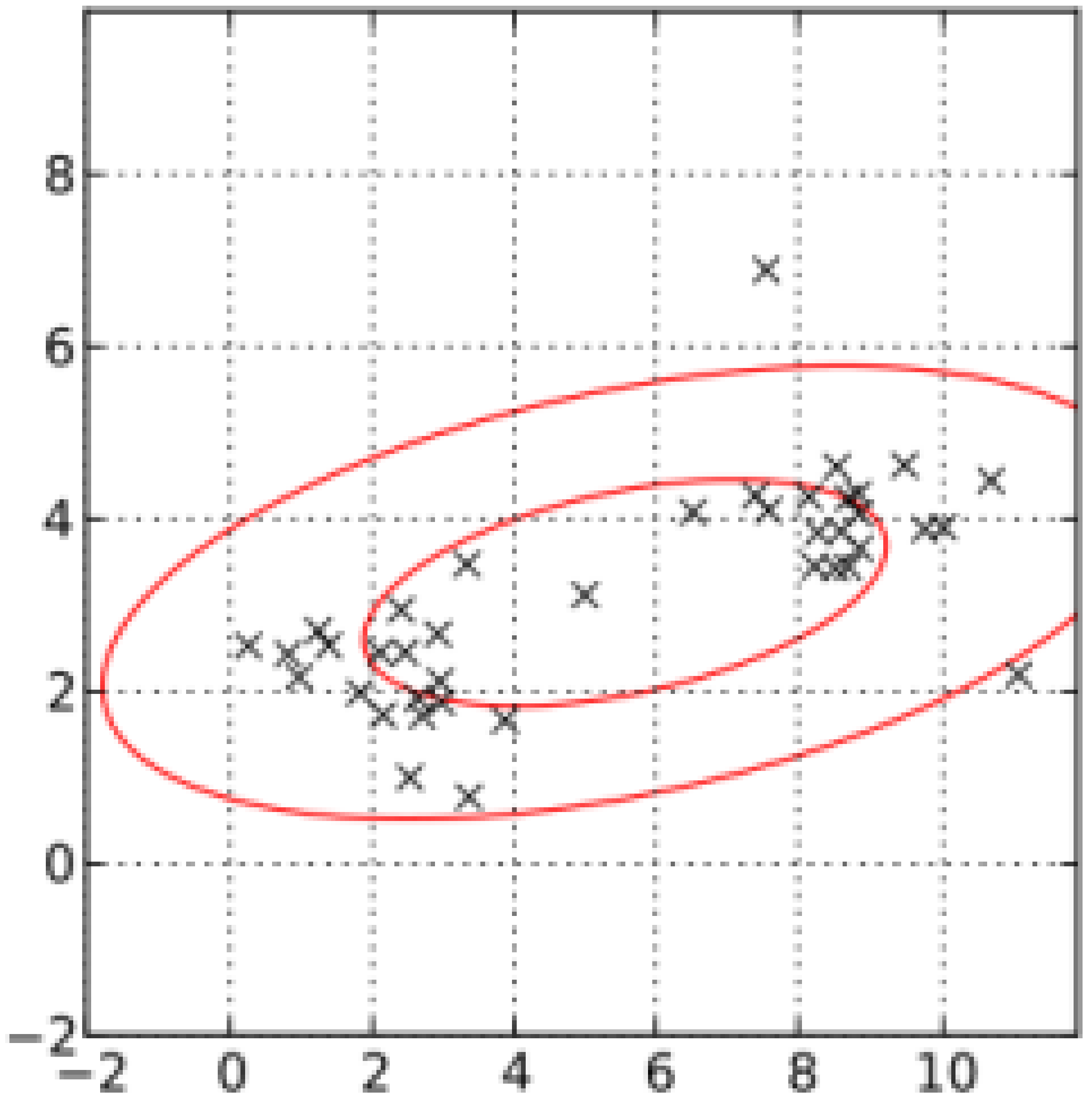
real MNIST samples

generated samples from VAE

Are there significant difference in VAE and MNIST?

Given: A model P and samples from Q

Goal: is one Gaussian or two Gaussians more fit for Q?



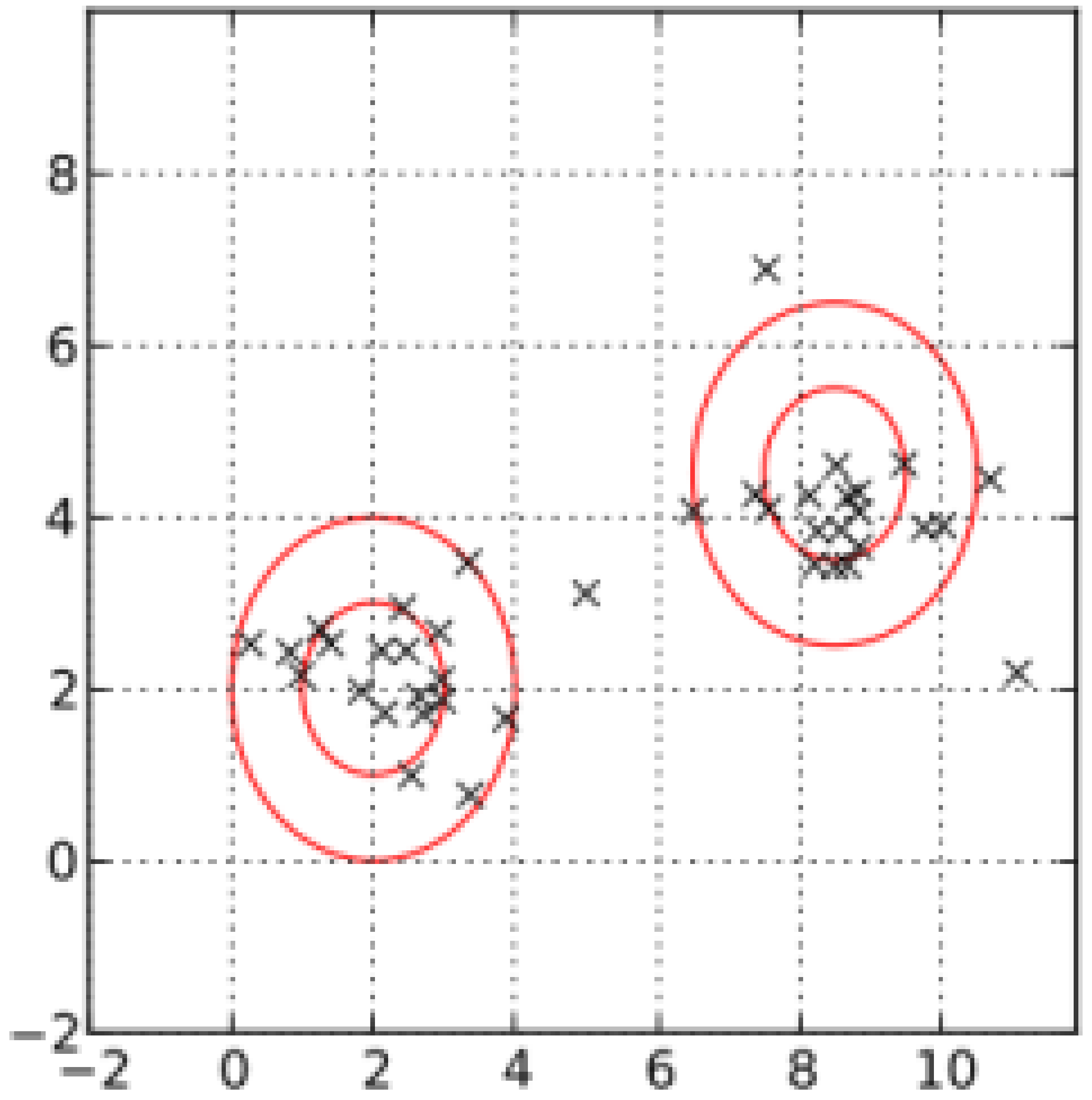
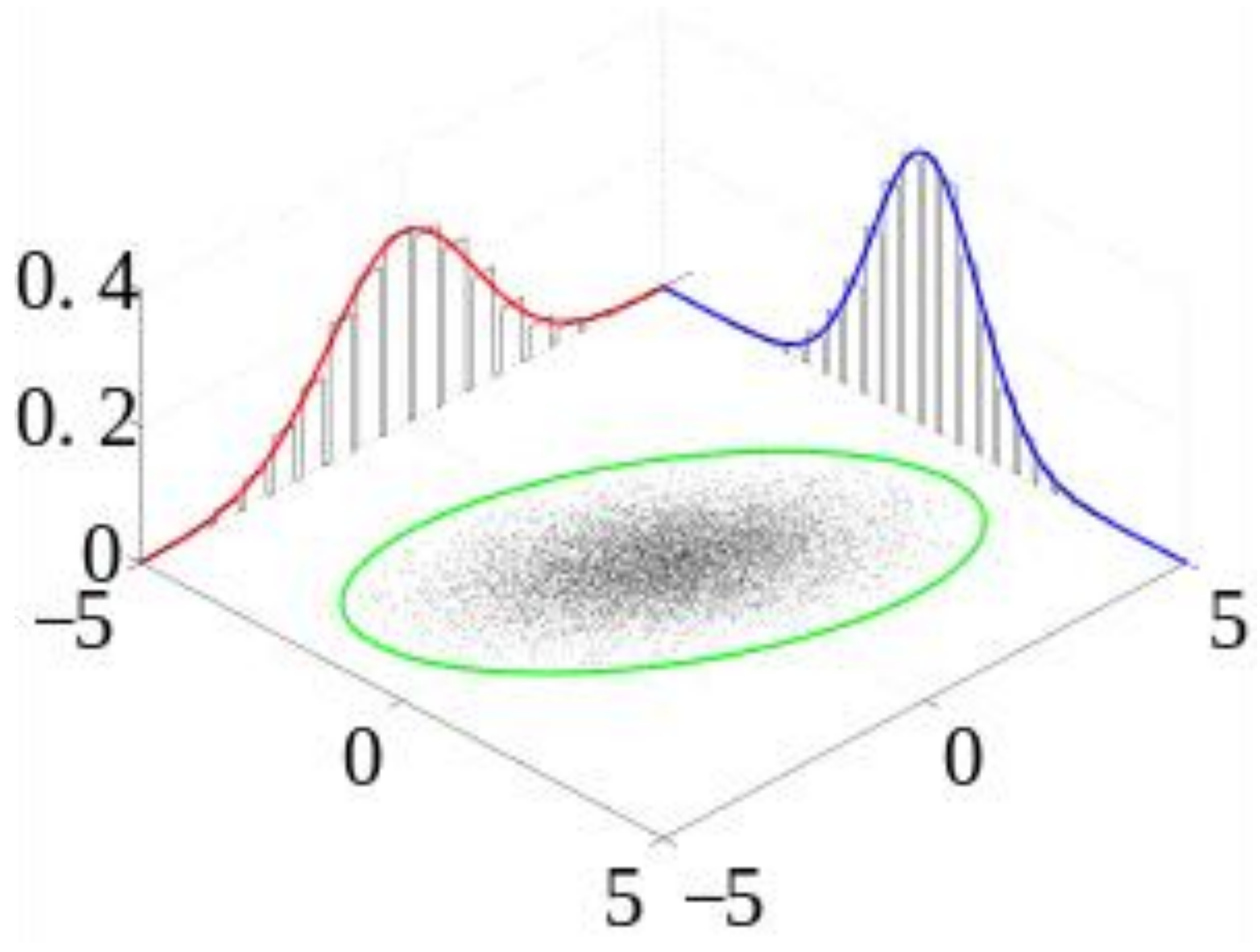


 Image taken from https://brilliant.org/wiki/gaussian-mixture-model/

Given: samples from a (joint) distribution $P_{X,Y}$

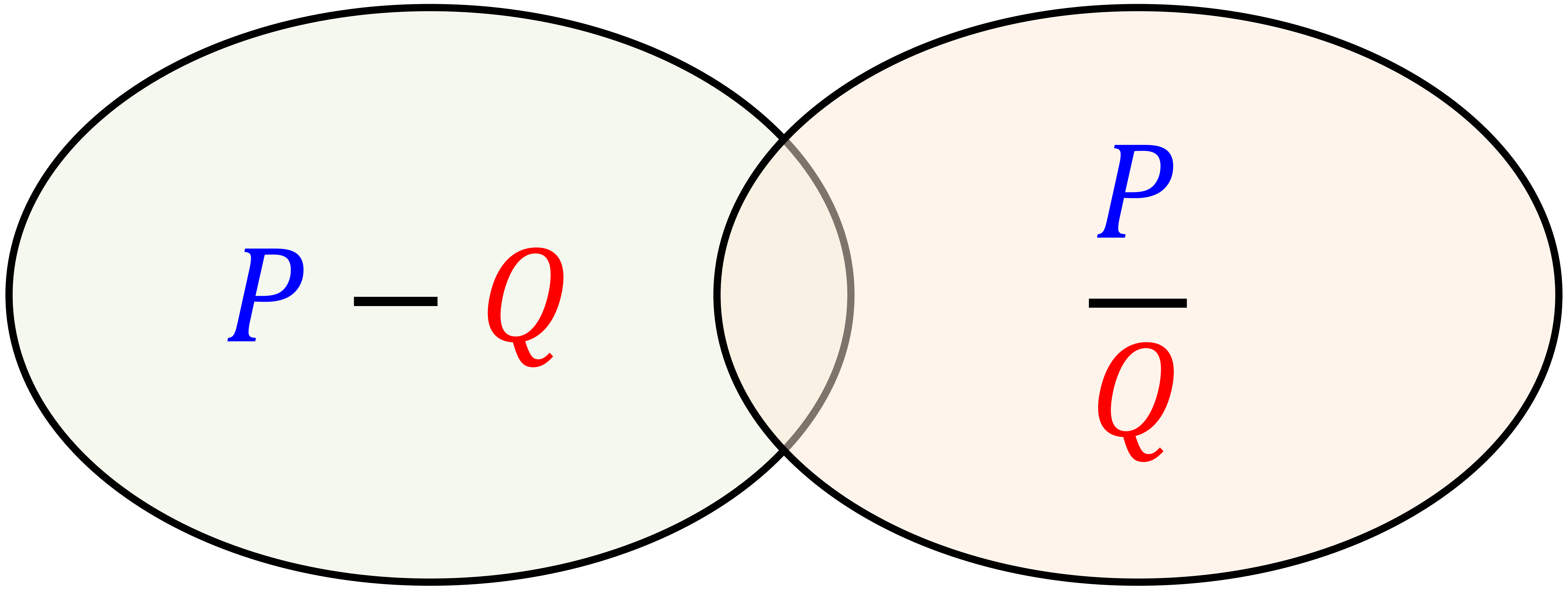
Goal: are X and Y independent? If not, the strength of dependence?



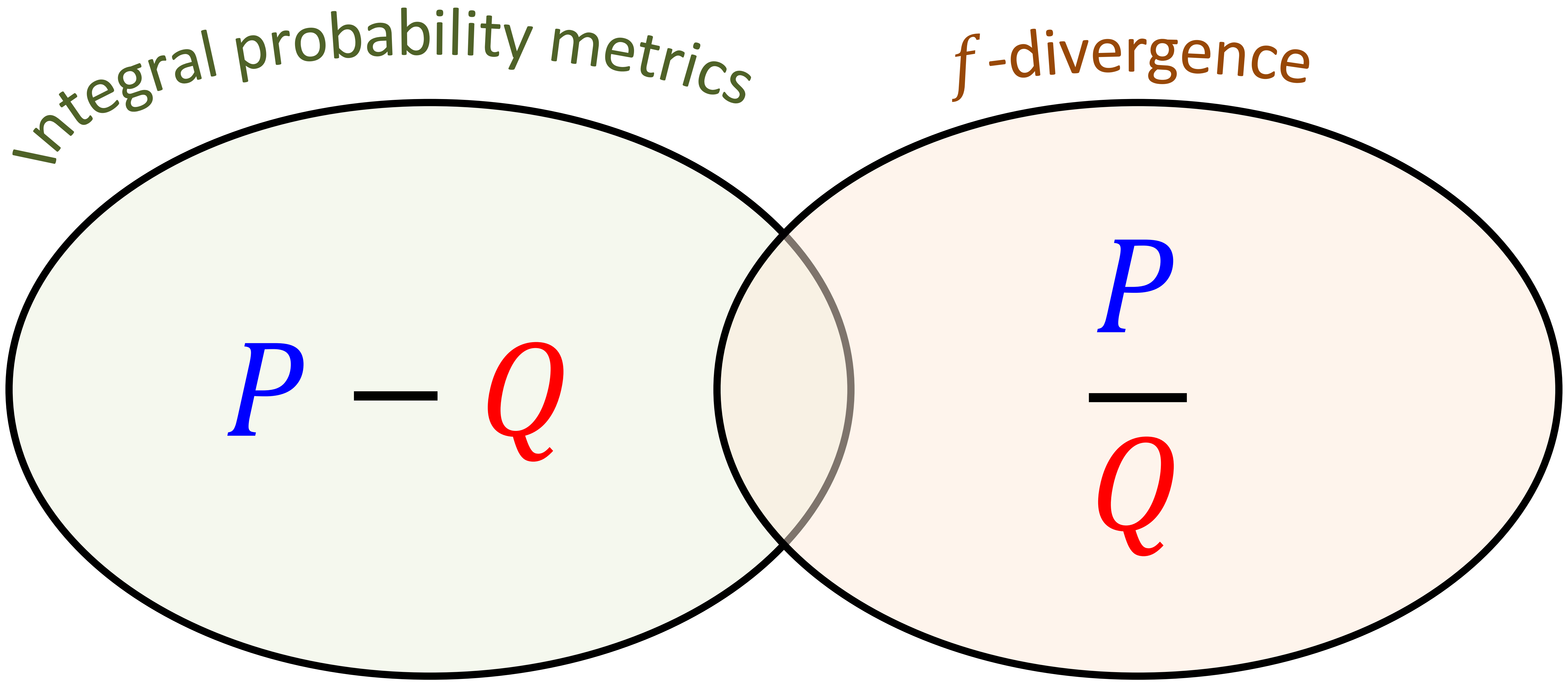
independent $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$

dependent $p(\mathbf{x}, \mathbf{y}) \neq p(\mathbf{x})p(\mathbf{y})$

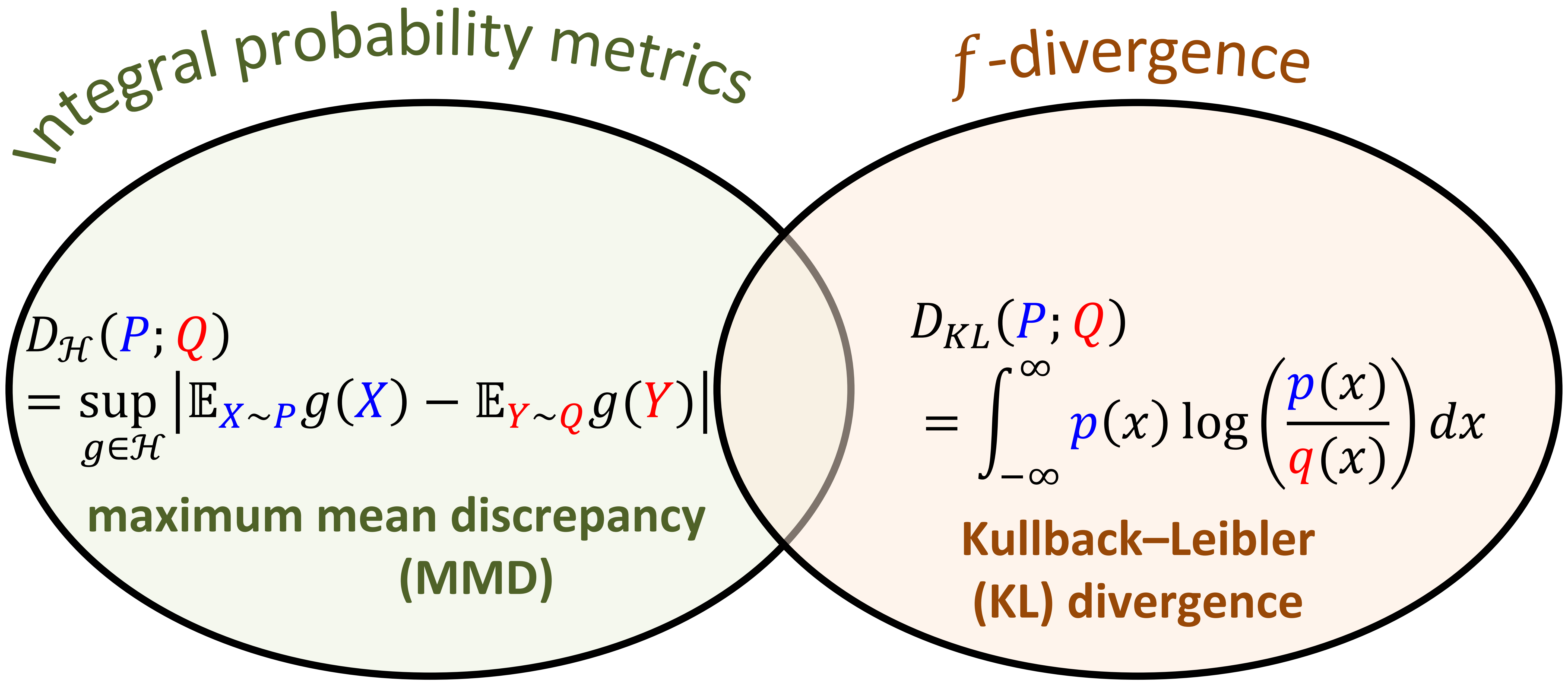
$$I = D_{KL}(p(\mathbf{x}, \mathbf{y}); p(\mathbf{x})p(\mathbf{y}))$$


$P - Q$
$\frac{P}{Q}$

DIVERGENCE

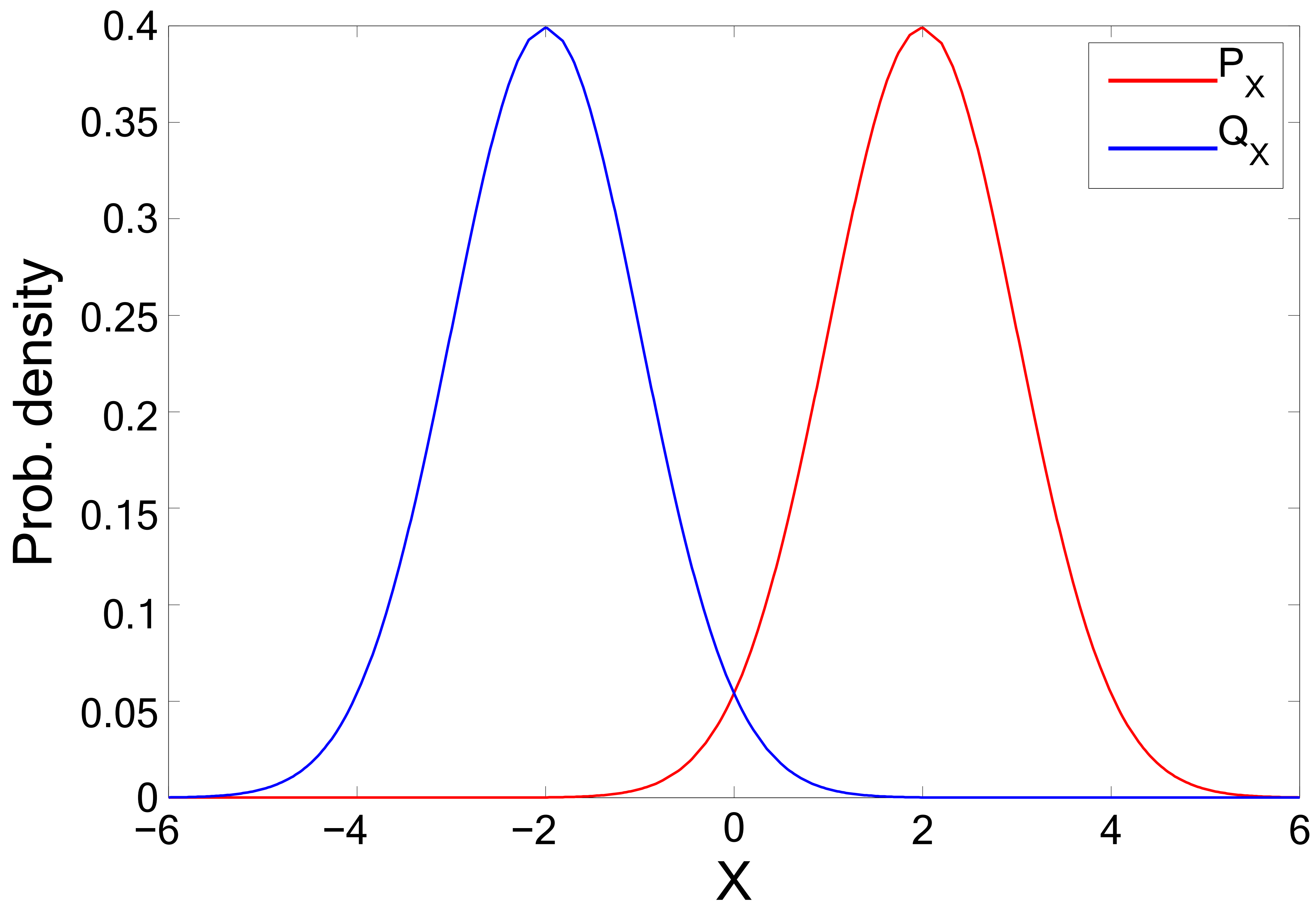


DIVERGENCE



Gretton, Arthur, et al. "A kernel two-sample test." *The Journal of Machine Learning Research* 13.1 (2012): 723-773. https://www.jmlr.org/papers/volume13/gretton12a/gretton12a.pdf

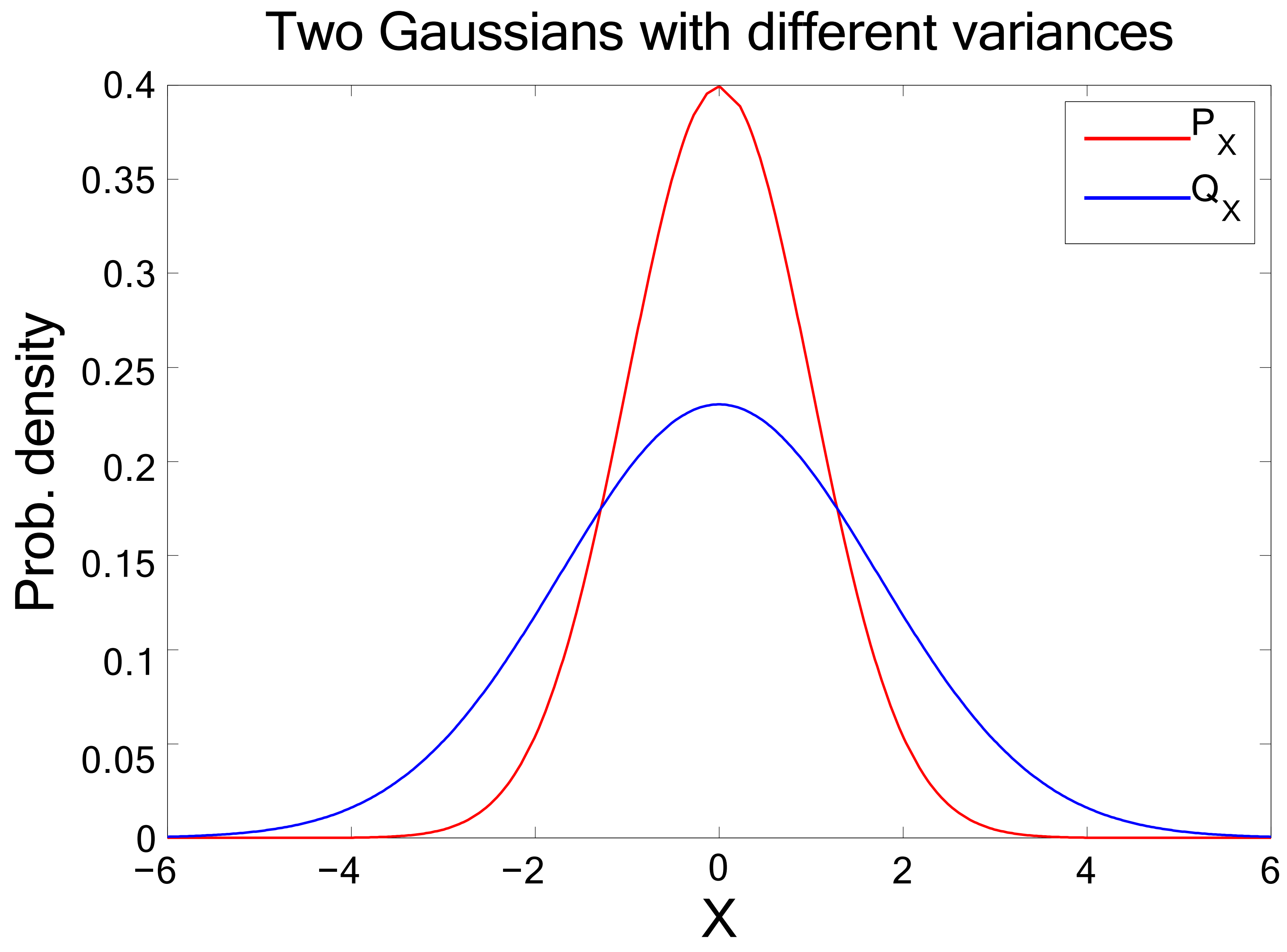
Two Gaussians with different means

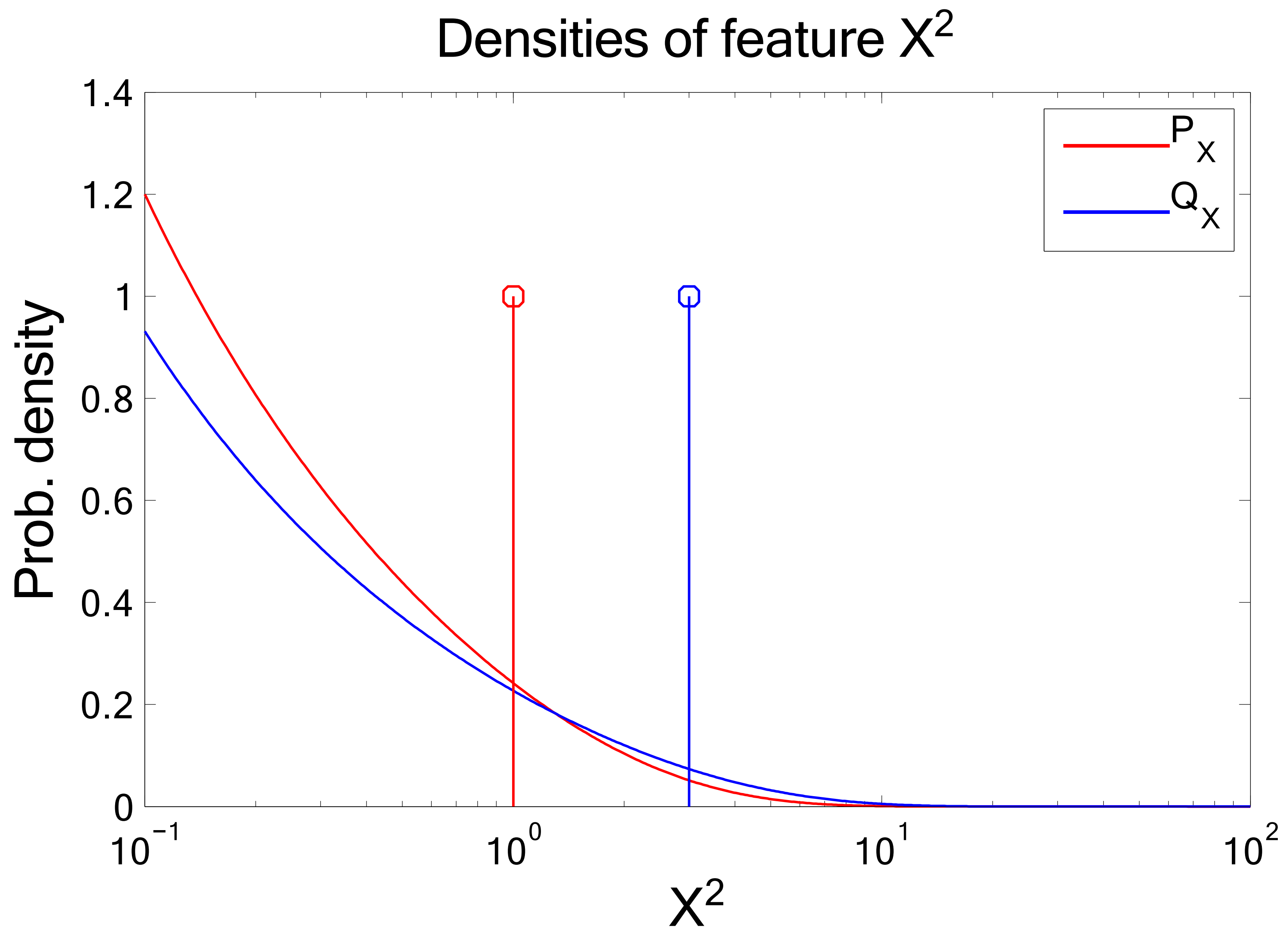


Two Gaussians with same mean but different variances

Idea: look at difference in means of features of the random variables

In Gaussian case: second order features of form $\varphi(x) = x^2$

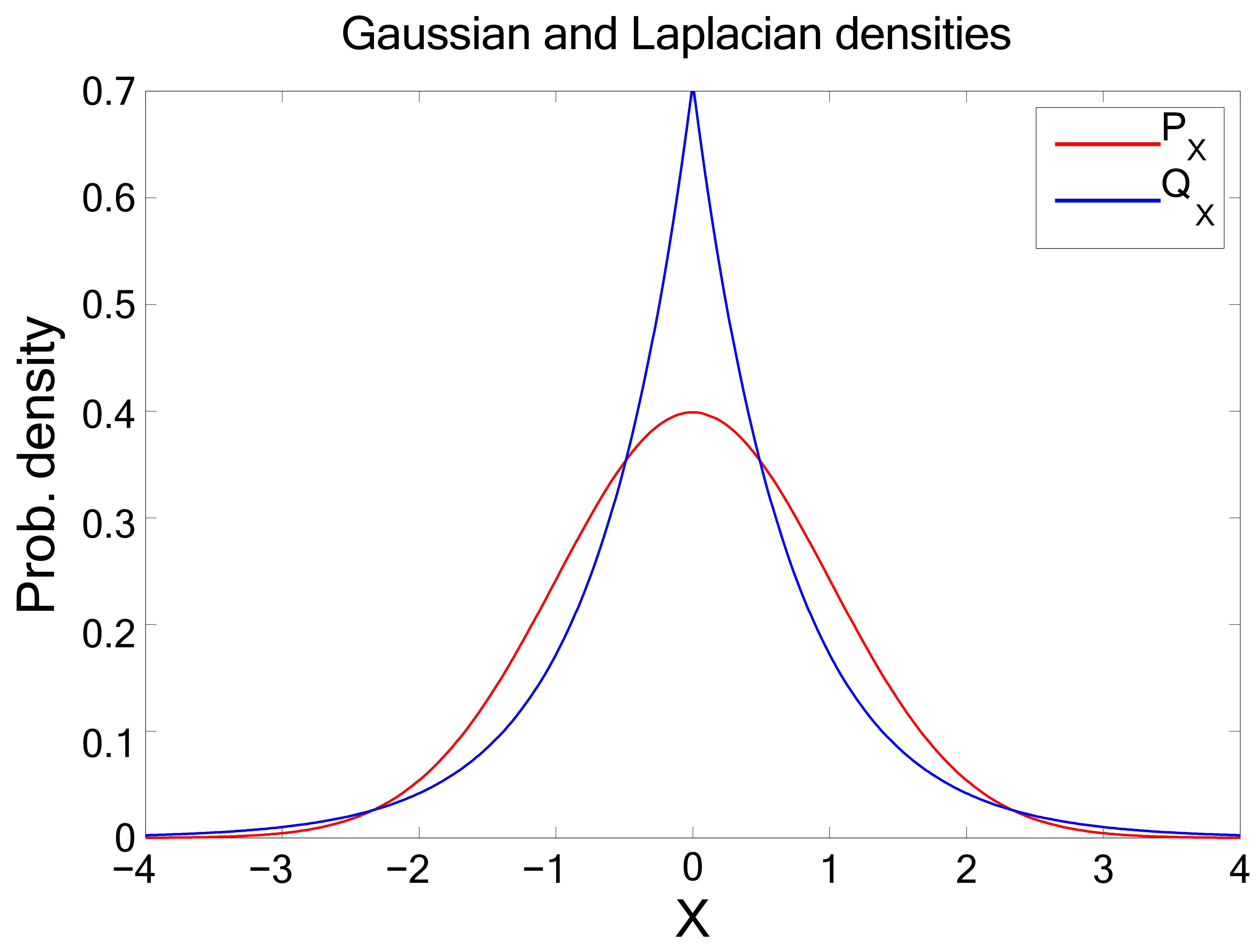




Gaussian and Laplacian distributions

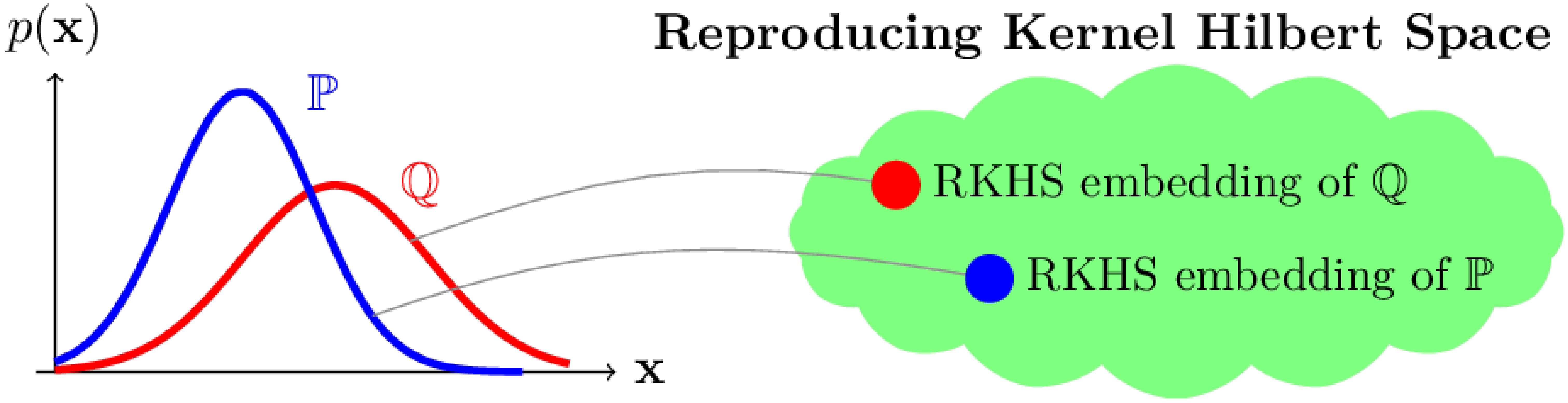
Same mean *and* same variance

Difference in means using higher order features



MAXIMUM MEAN DISCREPANCY

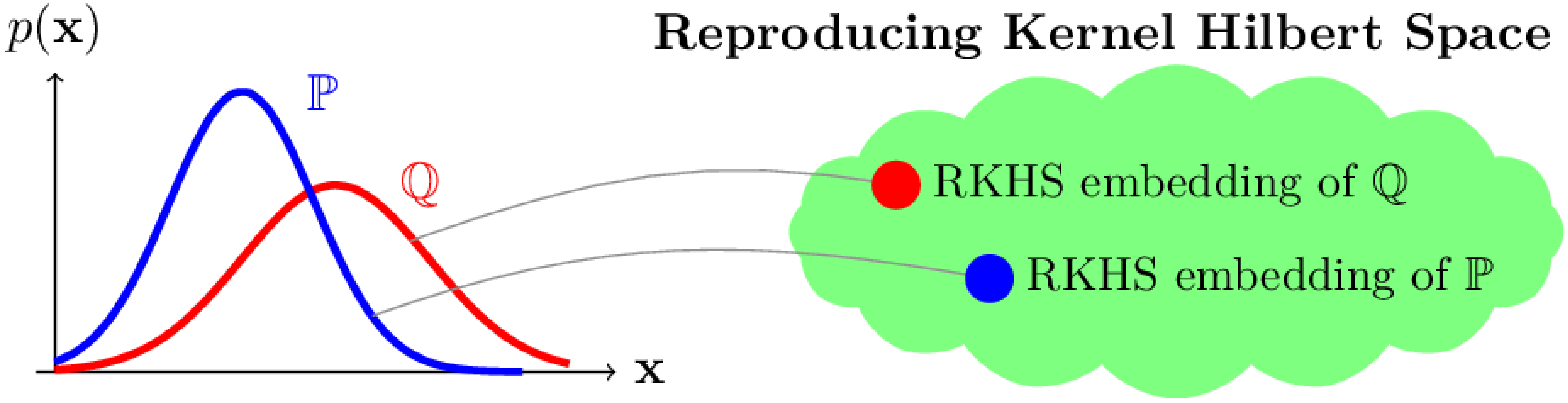
For a feature map $\varphi: \mathcal{X} \rightarrow \mathcal{H}$, representing distances between distributions as distances between mean embeddings of features



$$\text{MMD}^2(P; Q) = \left\| \mathbb{E}_{X\sim P}\varphi(X) - \mathbb{E}_{Y\sim Q}\varphi(Y) \right\|_{\mathcal{H}}^2$$

Muandet, Krikamol, et al. "Kernel mean embedding of distributions: A review and beyond." *Foundations and Trends® in Machine Learning* 10.1-2 (2017): 1-141. https://www.nowpublishers.com/article/Details/MAL-060

For a feature map $\varphi\colon \mathcal{X} \to \mathcal{H}$, representing distances between distributions as distances between mean embeddings of features



$$
\begin{aligned}
\mathrm{MMD}^2(P;Q) &= \left\| \mathbb{E}_{X\sim P}\varphi(X) - \mathbb{E}_{Y\sim Q}\varphi(Y) \right\|_{\mathcal{H}}^2 \\
&= \left\langle \mathbb{E}_{X\sim P}\varphi(X), \mathbb{E}_{X'\sim P}\varphi(X') \right\rangle_{\mathcal{H}} + \left\langle \mathbb{E}_{Y\sim Q}\varphi(Y), \mathbb{E}_{Y'\sim Q}\varphi(Y') \right\rangle_{\mathcal{H}} \\
&\quad -2\left\langle \mathbb{E}_{X\sim P}\varphi(X), \mathbb{E}_{Y\sim Q}\varphi(Y) \right\rangle_{\mathcal{H}} \\
&= \mathbb{E}_{X,X'\sim P}\, \kappa(X,X') + \mathbb{E}_{Y,Y'\sim Q}\kappa(Y,Y') - 2\mathbb{E}_{X\sim P,Y\sim Q}\kappa(X,Y)
\end{aligned}
$$

$$(x-y)^2 = x^T x + y^T y - 2x^T y$$

The kernel trick: $\kappa(x,y) = \langle \varphi(x), \varphi(y) \rangle_{\mathcal{H}}$

For a feature map $\varphi\colon \mathcal{X} \to \mathcal{H}$, representing distances between distributions as distances between mean embeddings of features

Can be inner product in infinite dimensional space

Assume $x \in R^1$ and $\gamma > 0$.

$$e^{-\gamma\|x_i - x_j\|^2} = e^{-\gamma(x_i - x_j)^2} = e^{-\gamma x_i^2 + 2\gamma x_i x_j - \gamma x_j^2}$$

$$= e^{-\gamma x_i^2 - \gamma x_j^2}\left(1 + \frac{2\gamma x_i x_j}{1!} + \frac{(2\gamma x_i x_j)^2}{2!} + \frac{(2\gamma x_i x_j)^3}{3!} + \cdots\right)$$

$$= e^{-\gamma x_i^2 - \gamma x_j^2}\left(1 \cdot 1 + \sqrt{\frac{2\gamma}{1!}}x_i \cdot \sqrt{\frac{2\gamma}{1!}}x_j + \sqrt{\frac{(2\gamma)^2}{2!}}x_i^2 \cdot \sqrt{\frac{(2\gamma)^2}{2!}}x_j^2 + \sqrt{\frac{(2\gamma)^3}{3!}}x_i^3 \cdot \sqrt{\frac{(2\gamma)^3}{3!}}x_j^3 + \cdots\right) = \phi(x_i)^T\phi(x_j),$$

where

$$\phi(x) = e^{-\gamma x^2}\left[1, \sqrt{\frac{2\gamma}{1!}}x, \sqrt{\frac{(2\gamma)^2}{2!}}x^2, \sqrt{\frac{(2\gamma)^3}{3!}}x^3, \cdots\right]^T$$

MAXIMUM MEAN DISCREPANCY

For a feature map $\varphi: \mathcal{X} \rightarrow \mathcal{H}$, representing distances between distributions as distances between mean embeddings of features

$$\mathrm{MMD}^2(P;Q) = \mathbb{E}_{X,X'\sim P}\kappa(X,X') + \mathbb{E}_{Y,Y'\sim Q}\kappa(Y,Y') - 2\mathbb{E}_{X\sim P,Y\sim Q}\kappa(X,Y)$$

$$\widehat{\mathrm{MMD}}^2(P;Q) = \underbrace{\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} G_\sigma(x_i - x_j)}_{\text{within distribution similarity}} + \underbrace{\frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M} G_\sigma(y_i - y_j)}_{\text{within distribution similarity}} - \underbrace{\frac{2}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} G_\sigma(x_i - y_j)}_{\text{cross-distribution similarity}}$$

Gretton, Arthur, et al. "A kernel two-sample test." *The Journal of Machine Learning Research* 13.1 (2012): 723-773.

PART THREE: MMD-VAE

SOME ISSUES OF VAE OBJECTIVE

Uninformative latent code

- $D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}); p_\lambda(\boldsymbol{z})\right)$ might be too restrictive
- If the decoder is sufficiently flexible, failed to learn meaningful representation

A poor prior distribution $p_\lambda(\boldsymbol{z})$

MMD-VAE

Which divergence to choose?

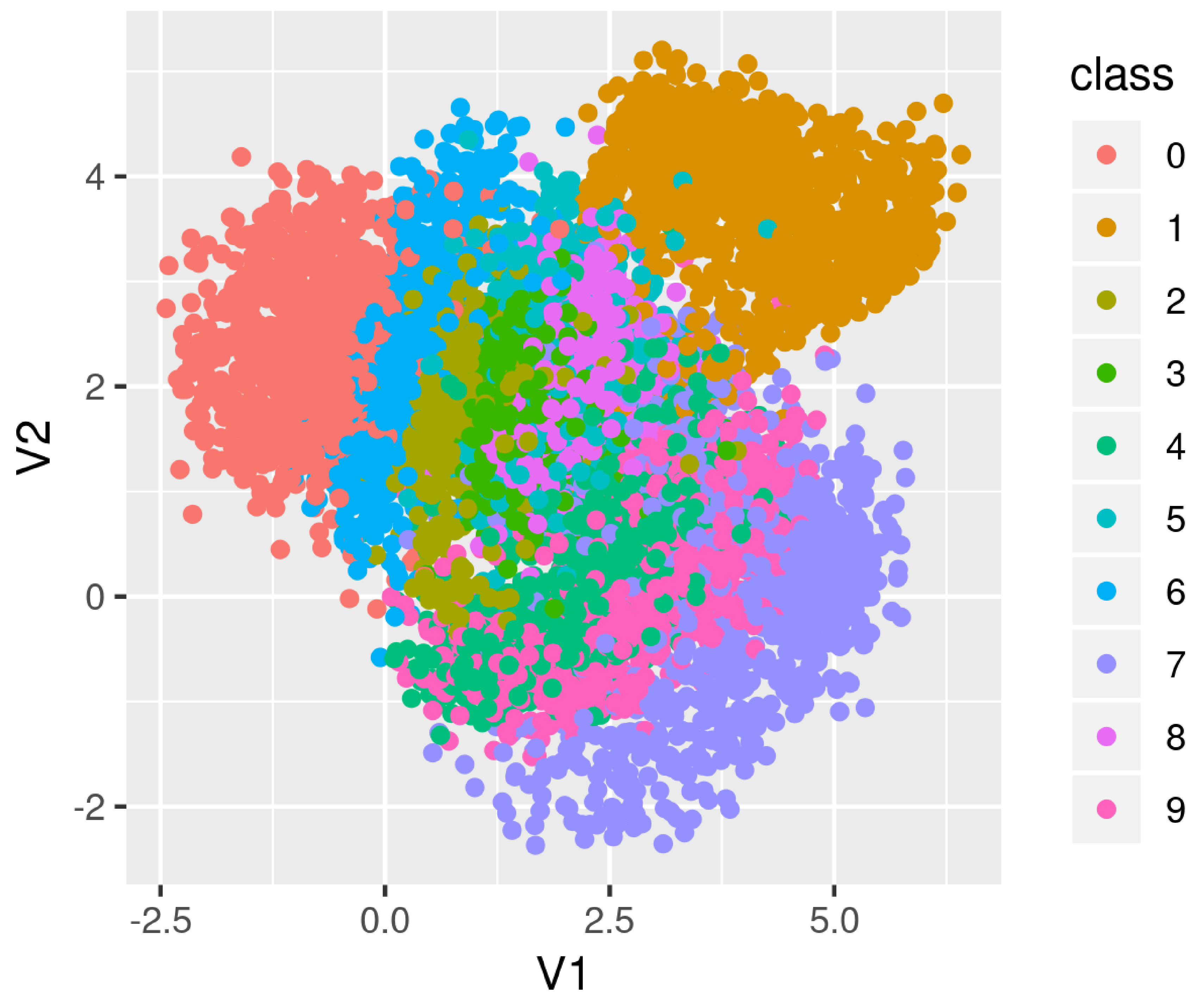
$$\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}); p_\lambda(\boldsymbol{z})\right)$$

MMD-VAE objective:

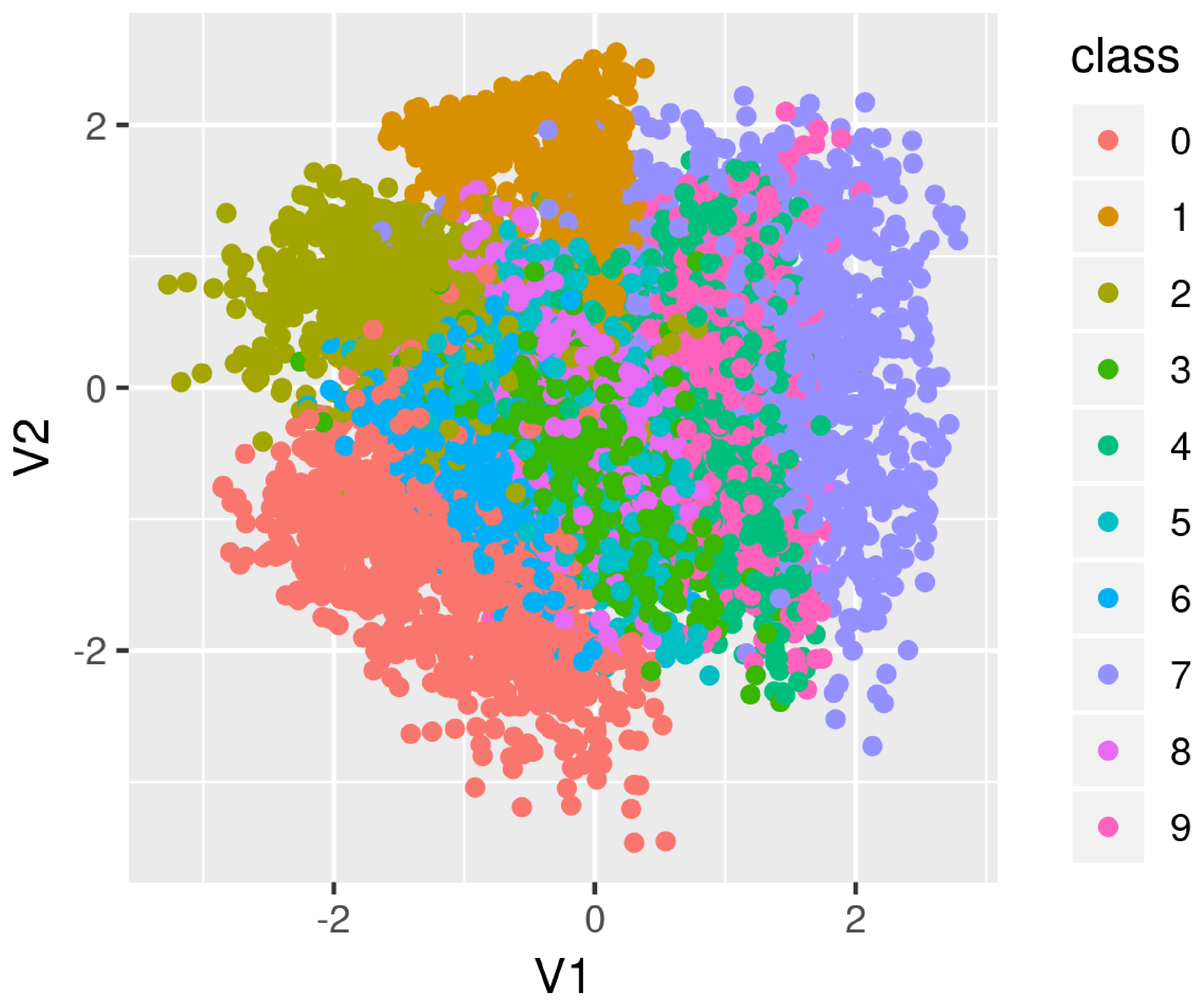
$$\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{MMD}\left(q_\phi(\boldsymbol{z}); p_\lambda(\boldsymbol{z})\right)$$

Zhao, Shengjia, Jiaming Song, and Stefano Ermon. "Infovae: Information maximizing variational autoencoders." *arXiv preprint arXiv:1706.02262* (2017). https://arxiv.org/abs/1706.02262

Which divergence to choose?

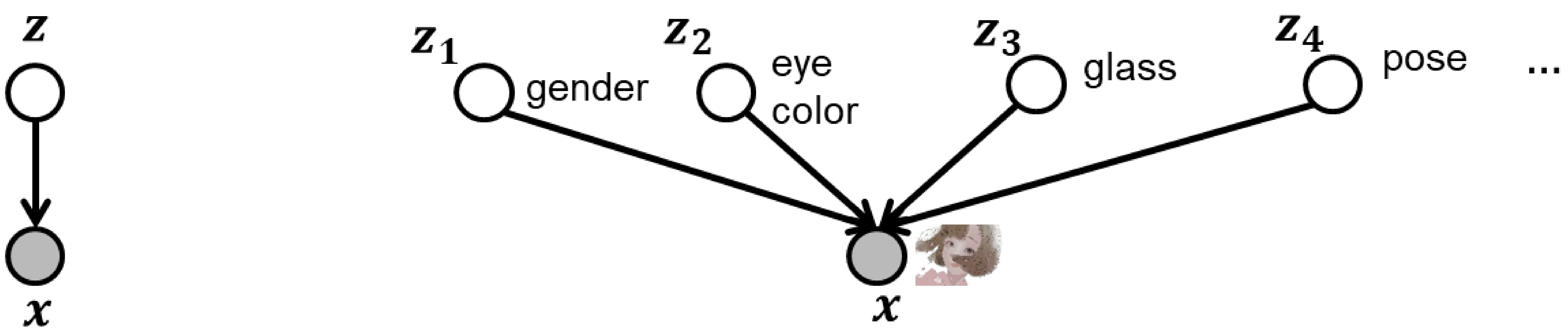


VAE



MMD-VAE

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



VAE AND DISENTANGLEMENT

$$D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}); p_\lambda(\boldsymbol{z})\right)$$

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

$$= \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{q_\phi(\boldsymbol{z})} \frac{q_\phi(\boldsymbol{z})}{\prod_j q_\phi(\boldsymbol{z}_j)} \frac{\prod_j q_\phi(\boldsymbol{z}_j)}{p_\lambda(\boldsymbol{z})} d\boldsymbol{z}$$

$$= \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{q_\phi(\boldsymbol{z})} \frac{q_\phi(\boldsymbol{z})}{\prod_j q_\phi(\boldsymbol{z}_j)} \frac{\prod_j q_\phi(\boldsymbol{z}_j)}{\prod_j p_\lambda(\boldsymbol{z}_j)} d\boldsymbol{z}$$

$$= D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}); q_\phi(\boldsymbol{z})\right) + D_{KL}\left(q_\phi(\boldsymbol{z}); \prod_j q_\phi(\boldsymbol{z}_j)\right) + D_{KL}\left(\prod_j q_\phi(\boldsymbol{z}_j); \prod_j p_\lambda(\boldsymbol{z}_j)\right)$$

$$= I(\boldsymbol{z}; \boldsymbol{x}) + \underbrace{D_{KL}\left(q_\phi(\boldsymbol{z}); \prod_j q_\phi(\boldsymbol{z}_j)\right)}_{\text{Total correlation}} + \underbrace{\sum_{j=1}^{d} D_{KL}\left(q_\phi(\boldsymbol{z}_j); p_\lambda(\boldsymbol{z}_j)\right)}_{\text{dimension-wise KL divergence}}$$

Total correlation

dimension-wise KL divergence

VAE AND DISENTANGLEMENT

$$D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}); p_\lambda(\boldsymbol{z})\right)$$

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

$$= \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{q_\phi(\boldsymbol{z})} \frac{q_\phi(\boldsymbol{z})}{\prod_j q_\phi(\boldsymbol{z}_j)} \frac{\prod_j q_\phi(\boldsymbol{z}_j)}{p_\lambda(\boldsymbol{z})} d\boldsymbol{z}$$

$$= \int q_\phi(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{q_\phi(\boldsymbol{z})} \frac{q_\phi(\boldsymbol{z})}{\prod_j q_\phi(\boldsymbol{z}_j)} \frac{\prod_j q_\phi(\boldsymbol{z}_j)}{\prod_j p_\lambda(\boldsymbol{z}_j)} d\boldsymbol{z}$$

$$= D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}); q_\phi(\boldsymbol{z})\right) + D_{KL}\left(q_\phi(\boldsymbol{z}); \prod_j q_\phi(\boldsymbol{z}_j)\right) + D_{KL}\left(\prod_j q_\phi(\boldsymbol{z}_j); \prod_j p_\lambda(\boldsymbol{z}_j)\right)$$

$$= I(\boldsymbol{z};\boldsymbol{x}) + \underbrace{D_{KL}\left(q_\phi(\boldsymbol{z}); \prod_j q_\phi(\boldsymbol{z}_j)\right)}_{\text{Total correlation (TC)}} + \underbrace{\sum_{j=1}^{d} D_{KL}\left(q_\phi(\boldsymbol{z}_j); p_\lambda(\boldsymbol{z}_j)\right)}_{\text{dimension-wise KL divergence}}$$

independence between each dimension of latent codes

Total correlation (TC)

dimension-wise KL divergence

β-VAE

A β-VAE optimizes the following function:

$$\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \beta D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}); p_\lambda(\boldsymbol{z})\right)$$

$$\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \beta \left\{\underbrace{I(\boldsymbol{z};\boldsymbol{x})}_{\text{Minimality}} + \underbrace{TC(\boldsymbol{z})}_{\text{Disentanglement}} + \underbrace{\sum_{j=1}^{d} D_{KL}\left(q_\phi(\boldsymbol{z}_j); p_\lambda(\boldsymbol{z}_j)\right)}_{\text{closeness to prior distribution}}\right\}$$

Minimality

Disentanglement

closeness to prior distribution

Assuming a factorized prior for $\boldsymbol{z}$, a β-VAE optimizes both for the information bottleneck (IB) Lagrangian and for disentanglement.

Higgins, Irina, et al. "beta-vae: Learning basic visual concepts with a constrained variational framework." *International conference on learning representations*. 2016. https://openreview.net/forum?id=Sy2fzU9gl

Burgess, Christopher P., et al. "Understanding disentangling in $\beta $-VAE." *arXiv preprint arXiv:1804.03599* (2018). https://arxiv.org/abs/1804.03599

VU

β-VAE

Start with very high β and slowly decrease during training.

Beginning: Very strict bottleneck, only encode most important factor

End: Very large bottleneck, encode all remaining factors

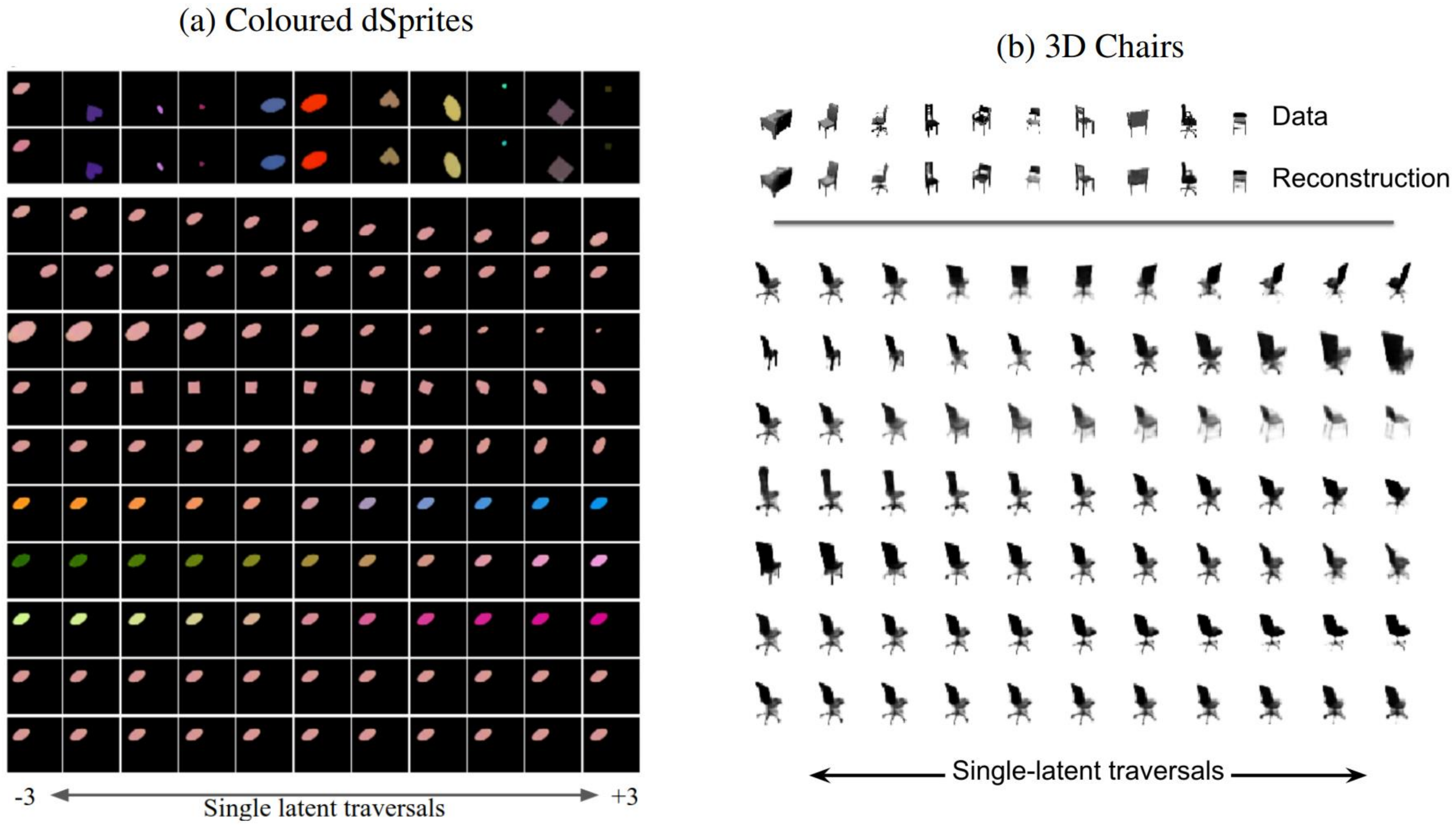


Figure 4: **Disentangling and reconstructions from β-VAE with controlled capacity increase. (a)** Latent traversal plots for a β-VAE trained with controlled capacity increaseon the coloured dSprites dataset. The top two rows show data samples and corresponding reconstructions. Subsequent rows show single latent traversals, ordered by their average KL divergence with the prior (high to low). To generate the traversals, we initialise the latent representation by inferring it from a seed image (left data sample), then traverse a single latent dimension (in $[-3, 3]$), whilst holding the remaining latent dimensions fixed, and plot the resulting reconstruction. The corresponding reconstructions are the rows of this figure. The disentangling is evident: different latent dimensions independently code for position, size, shape, rotation, and colour. **(b)** Latent traversal plots, as in (a), but trained on the Chairs dataset [3].