

Deep Learning

A course about theory & practice



Generative Networks: Variational Auto-Encoders (VAE)

Marco Piastra

Auto-Encoders

Auto-Encoders

▪ Auto-Encoder

Two main (composite) layers: **encoder** and **decoder**

One **hidden** or **latent** layer z

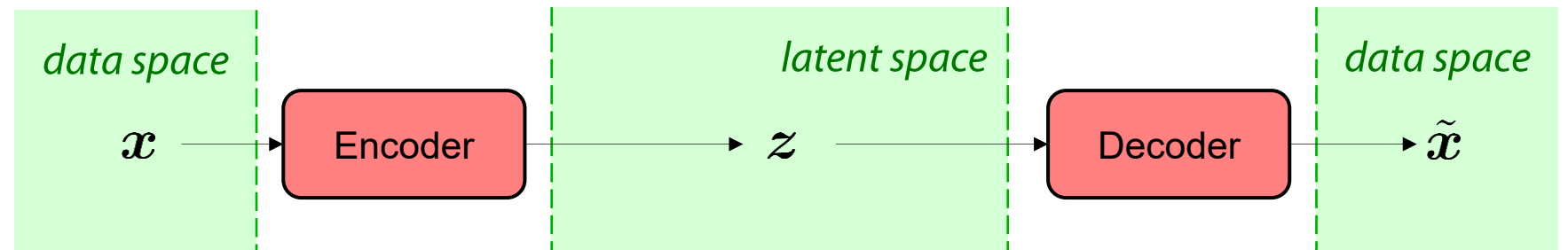
Each item in the dataset comprises the input only (*Unsupervised Learning*)

$$D := \{(\mathbf{x}^{(i)})\}_{i=1}^N,$$

The result of the optimization is z :

a compact (i.e. lower-dimensional) representation of the input \mathbf{x}

This representation
is also called
the *latent space*



Generative Adversarial Networks

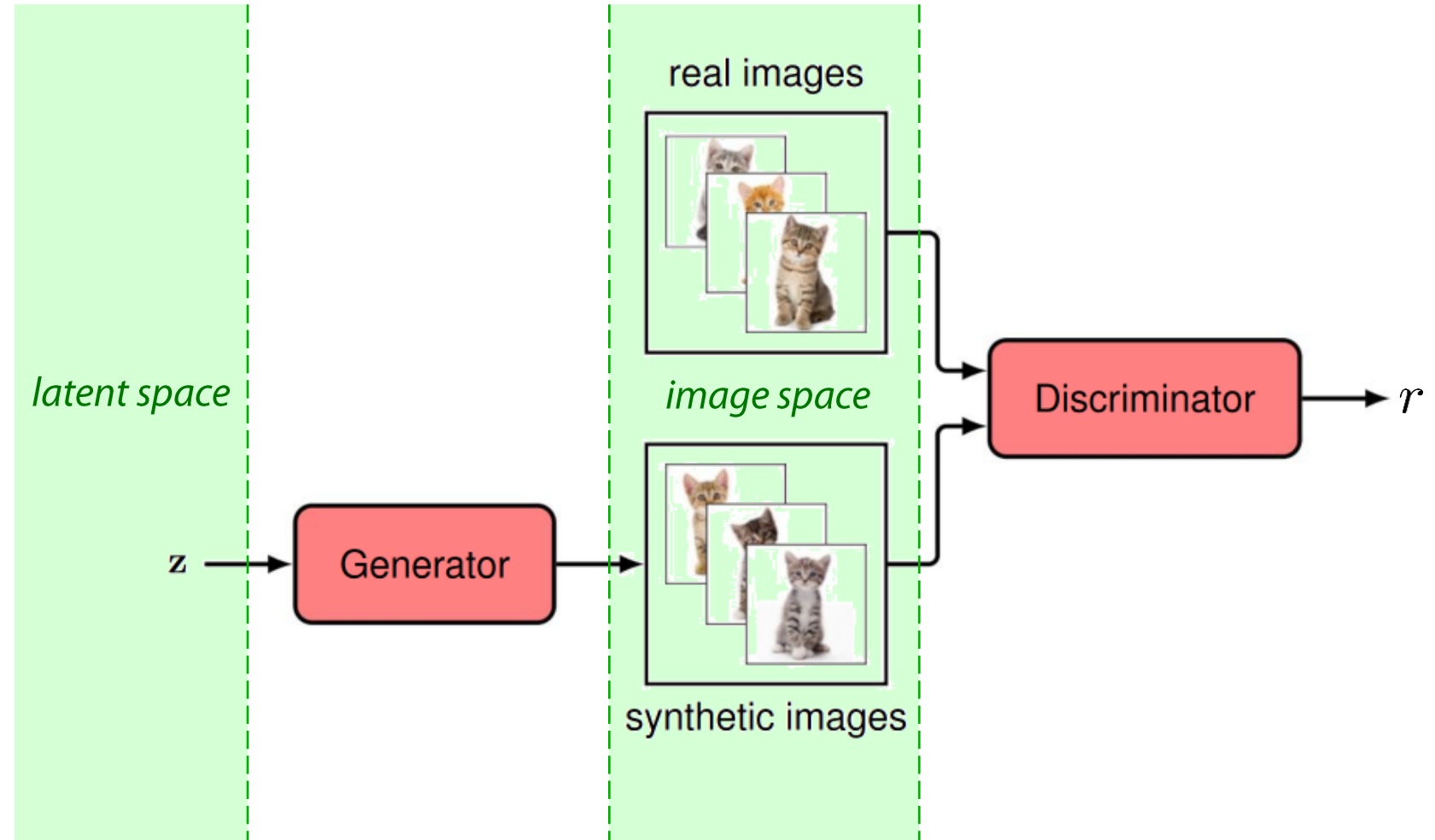
Basic idea: Decoder + Classifier

Objective:

creating a non-linear transformation from a latent space to a data space

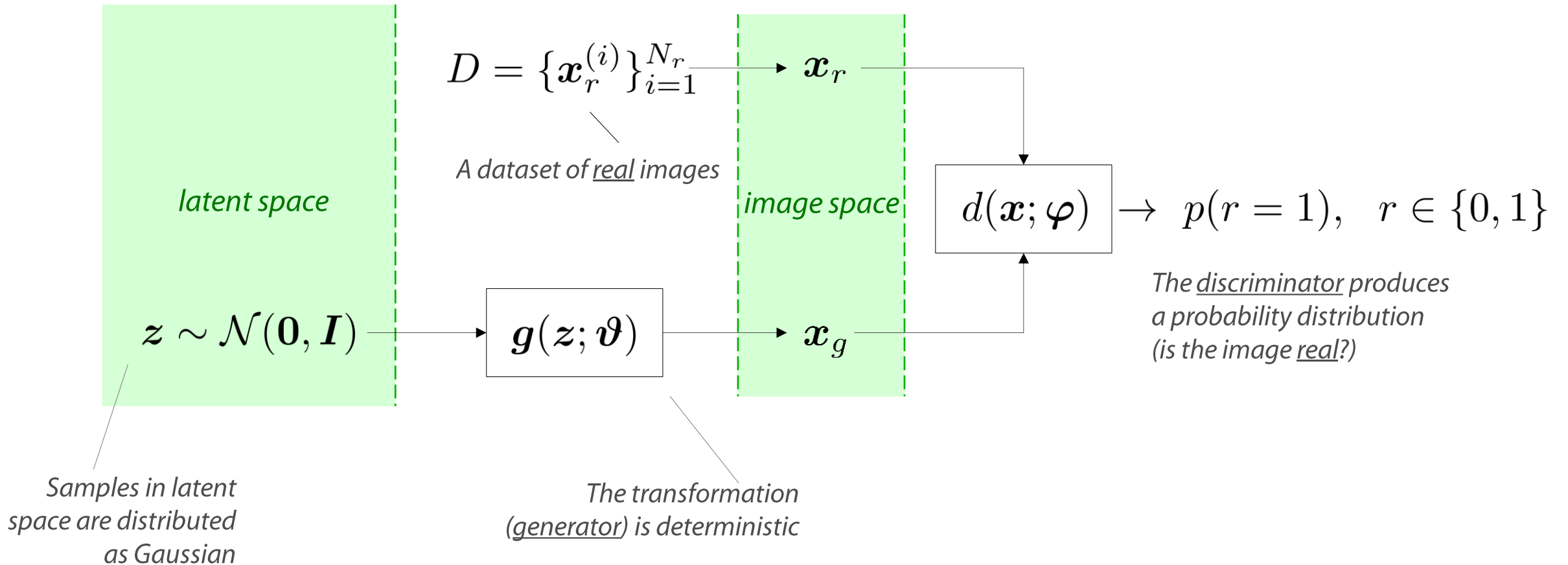
Method:

training together a generator and a discriminator using a real dataset

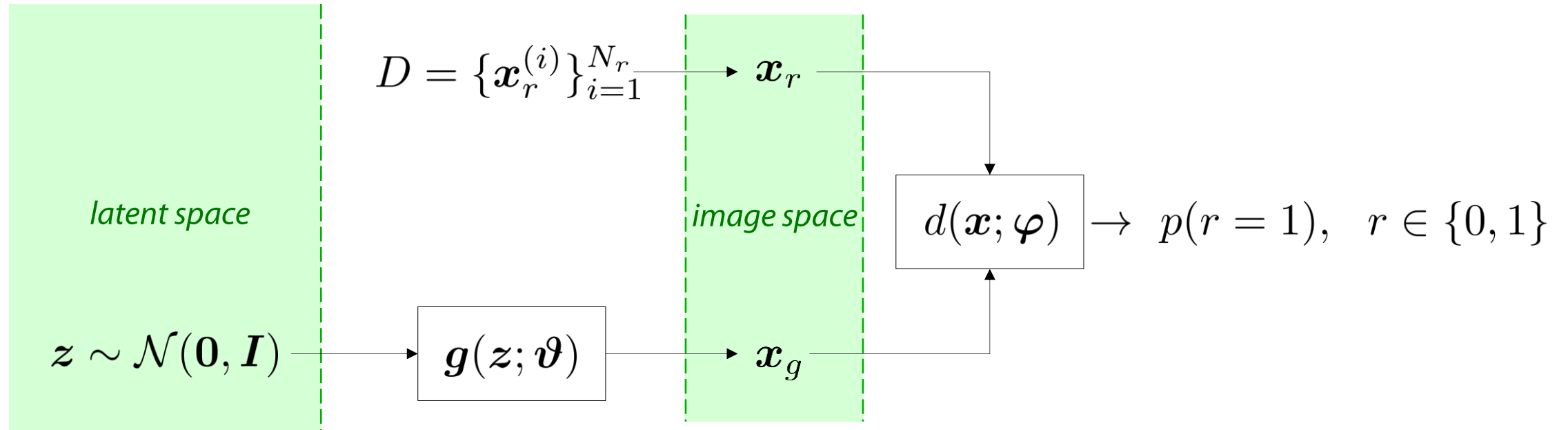


[Image from <https://www.bishopbook.com/>]

Generative Adversarial Network (GAN)



Generative Adversarial Network (GAN)



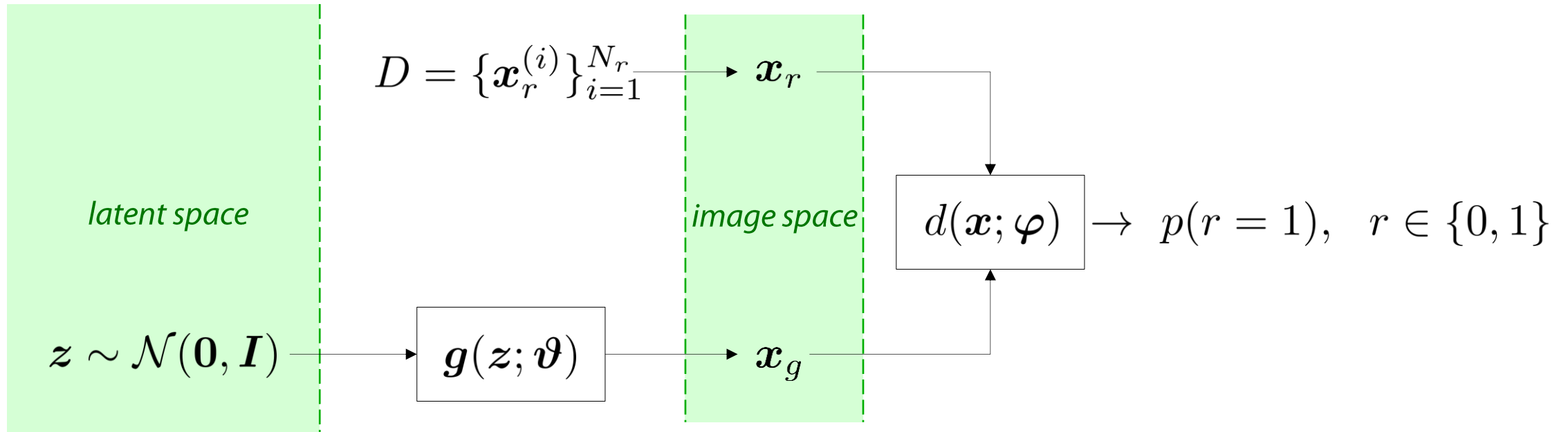
Loss function

$$L(\vartheta, \varphi) := -\frac{1}{N_r} \sum_{i \in \mathcal{R}} \ln(d(\mathbf{x}_r^{(i)}; \varphi)) - \frac{1}{N_g} \sum_{j \in \mathcal{G}} \ln(1 - d(\mathbf{g}(z^{(j)}; \vartheta); \varphi))$$

Cross-entropy
(d should recognize real images)

Cross-entropy
(d should recognize 'false' images)

Generative Adversarial Network (GAN)



Loss function

$$L(\vartheta, \varphi) := -\frac{1}{N_r} \sum_{i \in \mathcal{R}} \ln(d(\mathbf{x}_r^{(i)}; \varphi)) - \frac{1}{N_g} \sum_{j \in \mathcal{G}} \ln(1 - d(\mathbf{g}(z^{(j)}; \vartheta); \varphi))$$

Gradients

$$\Delta \varphi = -\eta \frac{\partial}{\partial \varphi} L(\vartheta, \varphi)$$

Make the discriminator smarter

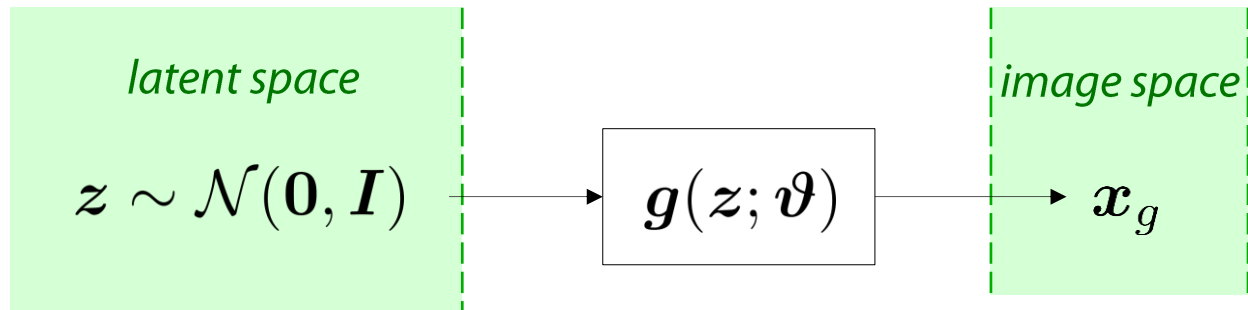
$$\Delta \vartheta = +\eta \frac{\partial}{\partial \vartheta} L(\vartheta, \varphi)$$

Make the generator smarter: the discriminator should be fooled

Generative Adversarial Network (GAN)

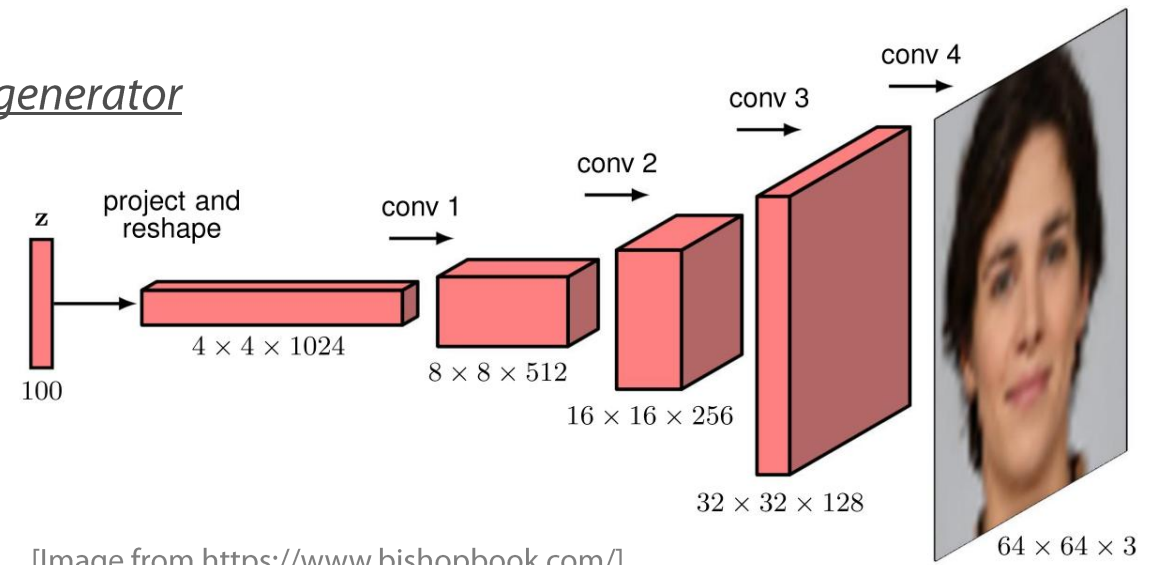
After training

The generator can be used to transform random samples in latent space into realistic data items



ImageGAN

Typically, a (de)convolutional network is used for the generator

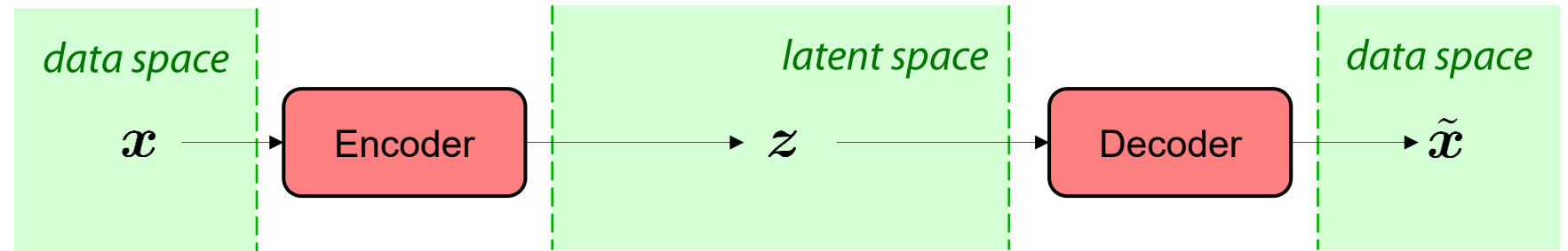


[Image from <https://www.bishopbook.com/>]

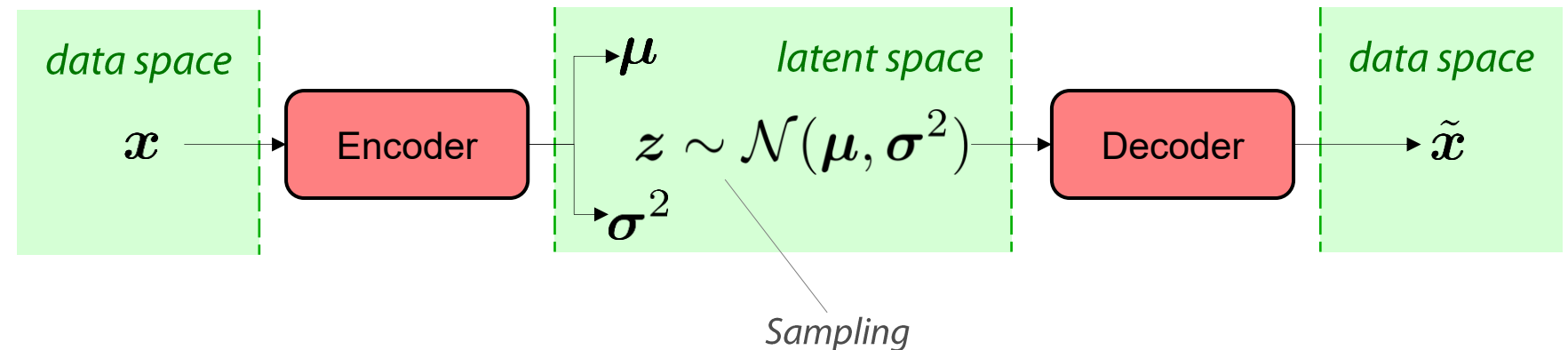
Variational Auto-Encoders

Basic idea

Auto-Encoder:
from data space
into latent space
then back

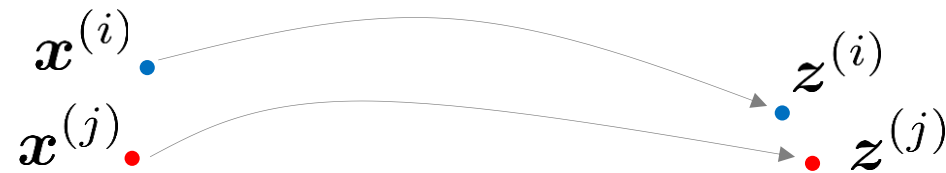
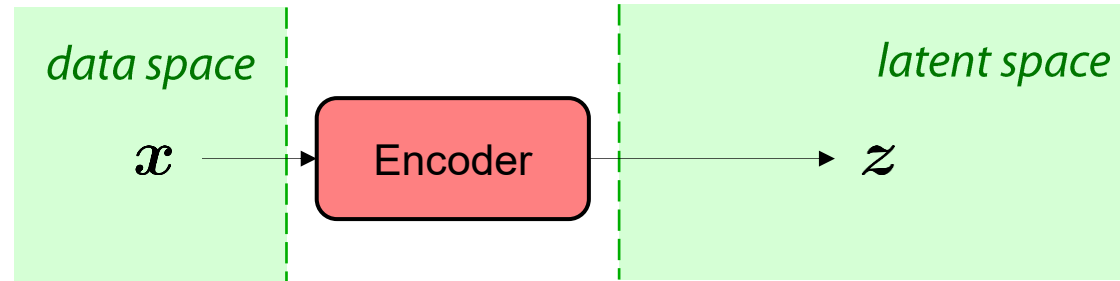


**Variational
Auto-Encoder:**
use a Gaussian spread
function to **organize**
the latent space

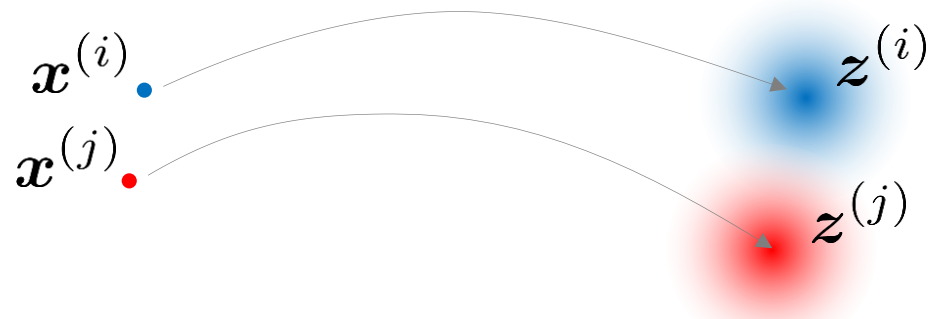
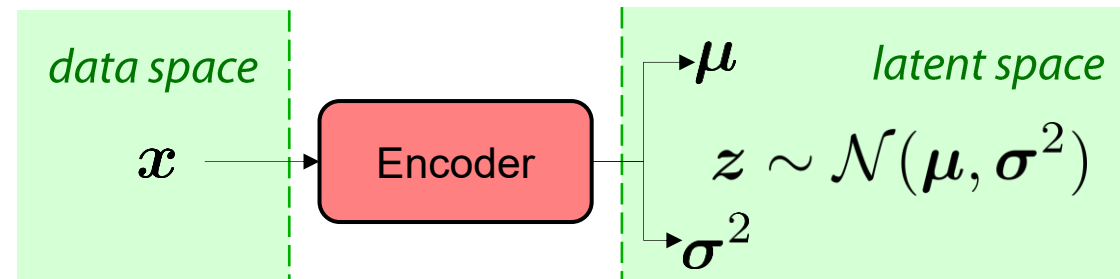


Organizing the latent space

Auto-Encoder:
the correspondence
between data space
and latent space
is one to one



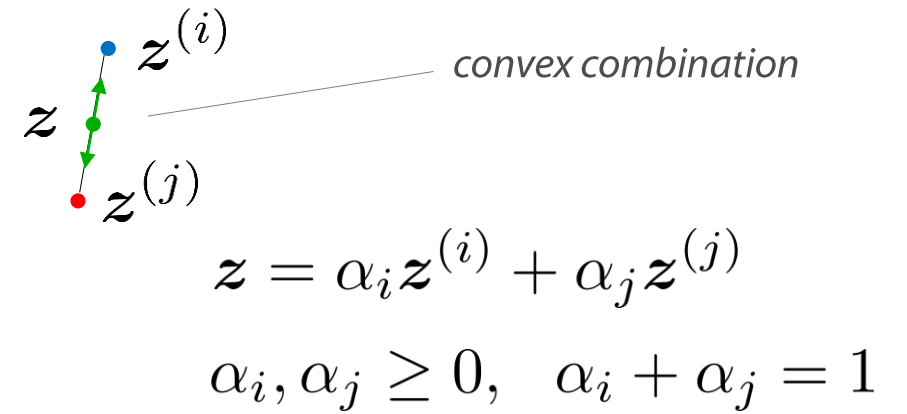
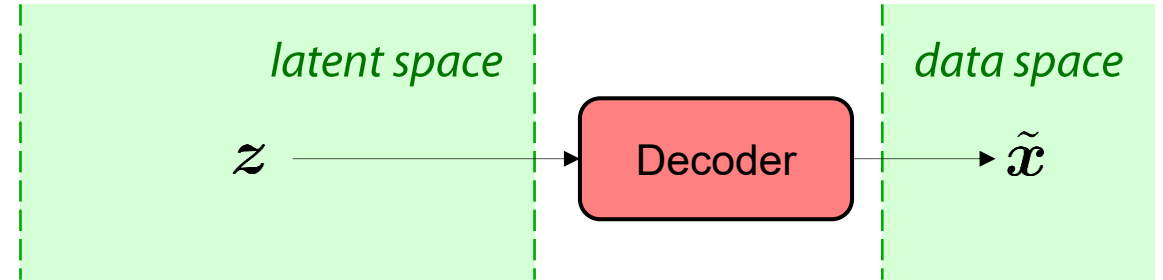
**Variational
Auto-Encoder:**
the correspondence
is one to many



Smooth generation

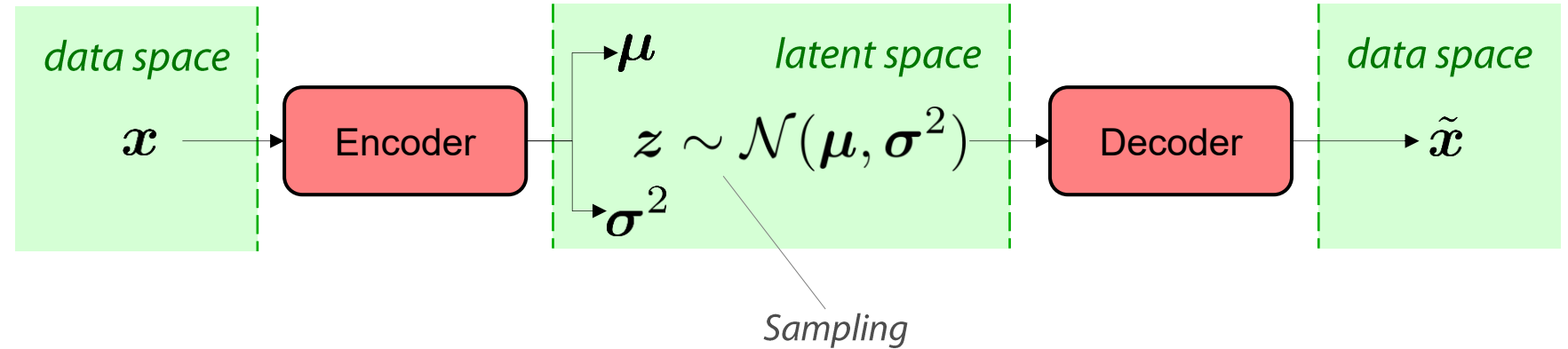
Variational Auto-Encoder:

after training, any convex combination of two points in latent space will generate a data item that changes smoothly from one extreme to the other



A problem: which loss function?

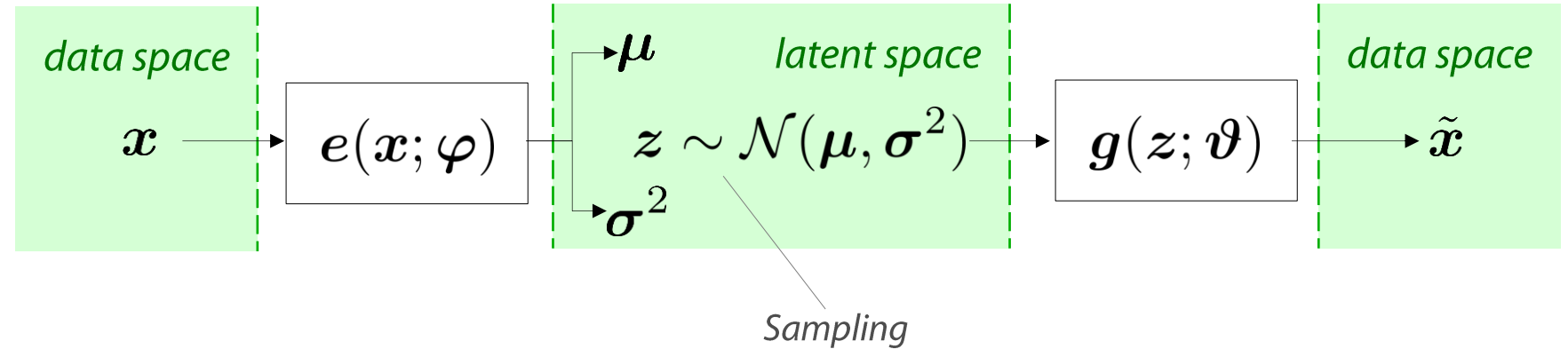
Variational Auto-Encoder:
use a Gaussian spread function to **organize** the latent space



This is what we want to train from a real dataset $D = \{\mathbf{x}_r^{(i)}\}_{i=1}^{N_r}$

A problem: which loss function?

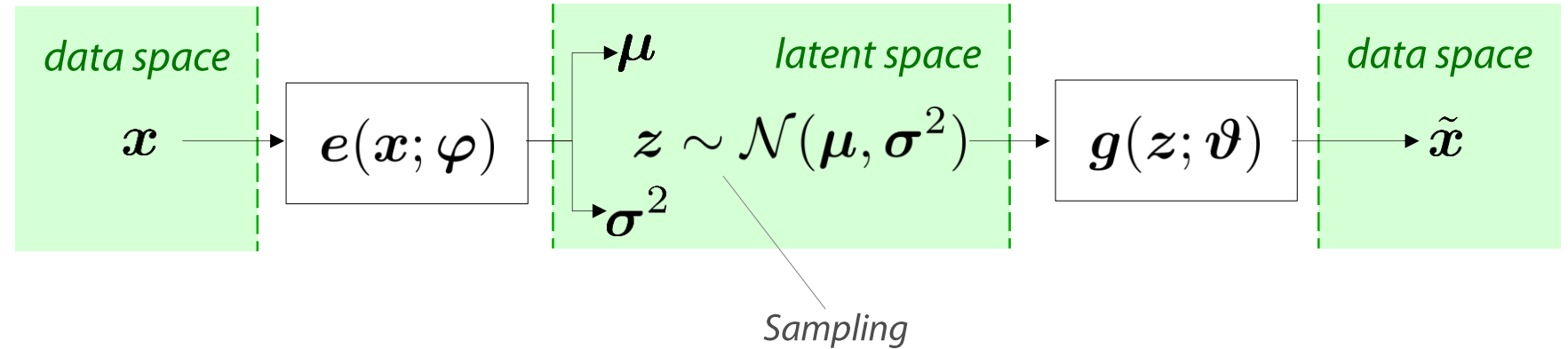
Variational Auto-Encoder:
use a Gaussian spread function to **organize** the latent space



This is what we want to train from a real dataset $D = \{\mathbf{x}^{(i)}\}_{i=1}^N$

A problem: which loss function?

Variational Auto-Encoder:
use a Gaussian spread function to **organize** the latent space



This is what we want to train from a real dataset $D = \{\mathbf{x}^{(i)}\}_{i=1}^N$

This is similar to the loss of a standard autoencoder

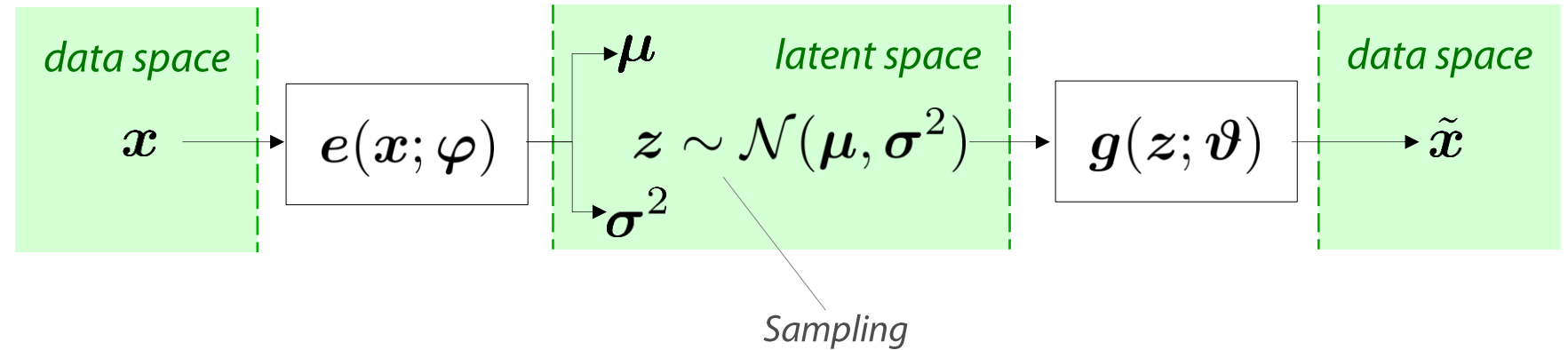
$$L(\mathbf{x}; \varphi, \vartheta) := \text{KL}(q(\mathbf{z} | \mathbf{x}, \varphi) \| p(\mathbf{z})) + \frac{1}{2} \frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|^2}{c}$$

Kullback-Leibler divergence

This is an hyperparameter (see later)

A problem: which loss function?

Variational Auto-Encoder:
use a Gaussian spread function to **organize** the latent space



This is what we want to train from a real dataset $D = \{\mathbf{x}^{(i)}\}_{i=1}^N$

$$L(\mathbf{x}; \varphi, \vartheta) := \text{KL}(q(\mathbf{z} | \mathbf{x}, \varphi) \| p(\mathbf{z})) + \frac{1}{2} \frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|^2}{c}$$

Design choices

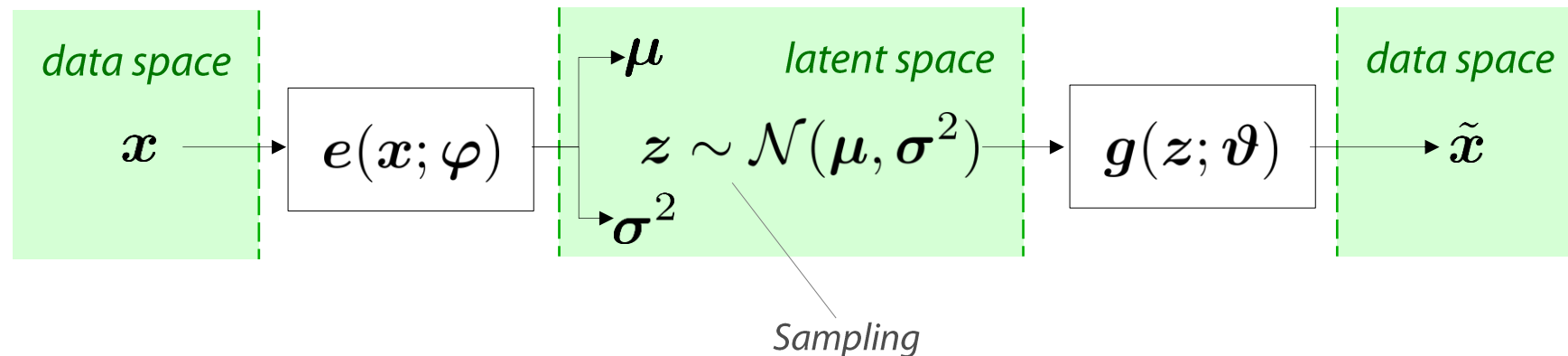
$$q(\mathbf{z} | \mathbf{x}, \varphi) := \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}; \varphi), \boldsymbol{\sigma}^2(\mathbf{x}; \varphi) \mathbf{I})$$

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

Normalization constraint: a soft limit against overspreading latent values

A problem: which loss function?

Variational Auto-Encoder:
use a Gaussian spread function to **organize** the latent space



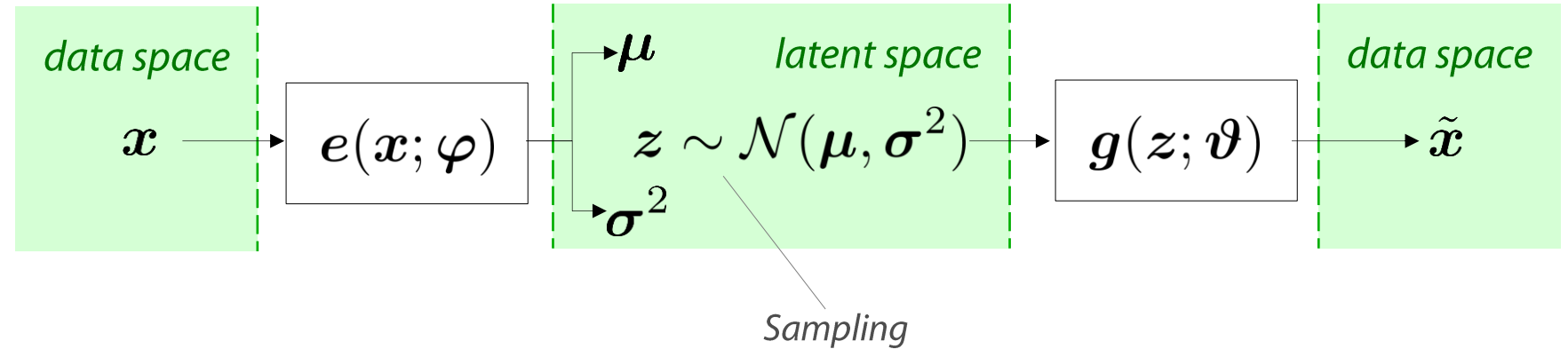
In general: $\text{KL}(q(\mathbf{z}) \parallel p(\mathbf{z})) := \int q(\mathbf{z}) \ln \frac{q(\mathbf{z})}{p(\mathbf{z})} d\mathbf{z}$ ——— Kullback-Leibler divergence; always positive, zero when the two distributions are identical

Since both distributions are normal:

$$\text{KL}(q(\mathbf{z} \mid \mathbf{x}, \varphi) \parallel p(\mathbf{z})) = -\frac{1}{2} \sum_{j=1}^{\dim(\mathbf{z})} (1 + \ln \sigma_j^2(\mathbf{x}; \varphi) - \mu_j^2(\mathbf{x}; \varphi) - \sigma_j^2(\mathbf{x}; \varphi))$$

A problem: which loss function?

Variational Auto-Encoder:
use a Gaussian spread function to **organize** the latent space



With a bit more of mathematics (omitted) it can be shown that the second term in the loss function

$$\frac{1}{2} \frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|^2}{c}$$

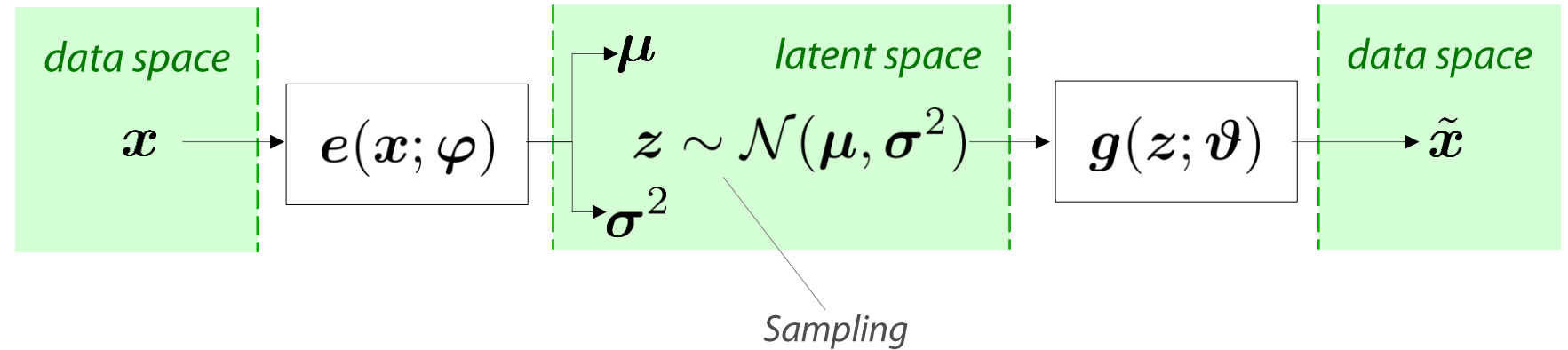
relates to an assumption of:

$$p(\tilde{\mathbf{x}}) := \mathcal{N}(\mathbf{x}, c\mathbf{I}), \quad c > 0$$

Annotations: "hyperparameter" points to c ; "Design choice" points to the \mathcal{N} symbol; "(hyper) spherical normal" points to the \mathbf{I} symbol.

Reparametrization Trick

Variational Auto-Encoder:
use a Gaussian spread function to **organize** the latent space



$$L(\mathbf{x}; \varphi, \vartheta) := -\frac{1}{2} \sum_{j=1}^{\dim(\mathbf{z})} (1 + \ln \sigma_j^2(\mathbf{x}; \varphi) - \mu_j^2(\mathbf{x}; \varphi) - \sigma_j^2(\mathbf{x}; \varphi)) + \frac{1}{2} \frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|^2}{c}$$

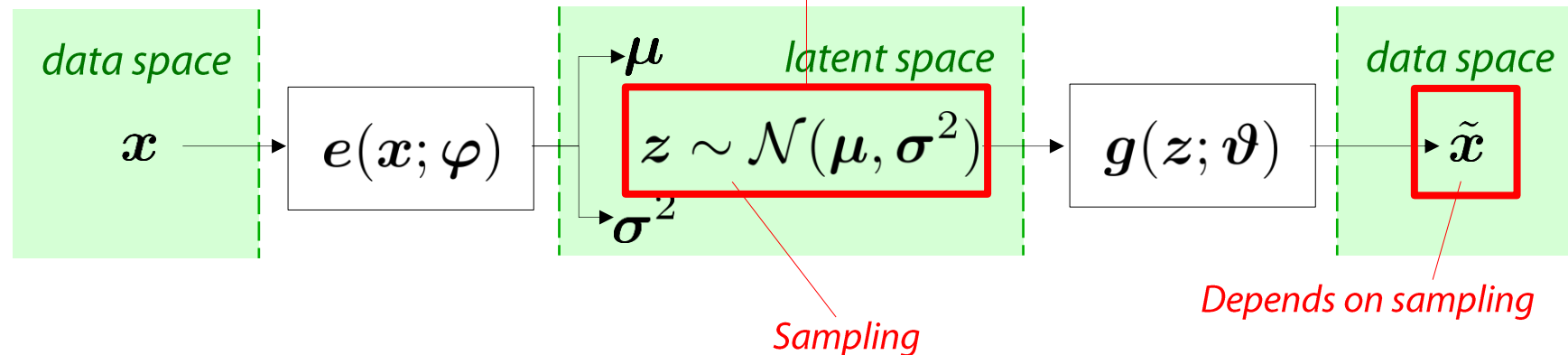
$$\Delta \varphi = -\eta \frac{\partial}{\partial \varphi} L(\mathbf{x}; \vartheta, \varphi)$$

$$\Delta \vartheta = -\eta \frac{\partial}{\partial \vartheta} L(\mathbf{x}; \vartheta, \varphi)$$

Reparametrization Trick

\tilde{x} depends on both ϑ and φ via z
 yet, when z is **sampled**, the derivative in φ is blocked

Variational Auto-Encoder:
 use a Gaussian spread function to **organize**
 the latent space



$$L(\mathbf{x}; \varphi, \vartheta) := -\frac{1}{2} \sum_{j=1}^{\dim(\mathbf{z})} (1 + \ln \sigma_j^2(\mathbf{x}; \varphi) - \mu_j^2(\mathbf{x}; \varphi) - \sigma_j^2(\mathbf{x}; \varphi)) + \frac{1}{2} \frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|^2}{c}$$

The trick is:

$$\mathbf{z} = \boldsymbol{\mu}(\mathbf{x}; \varphi) + \varepsilon \boldsymbol{\sigma}(\mathbf{x}; \varphi)$$

where:

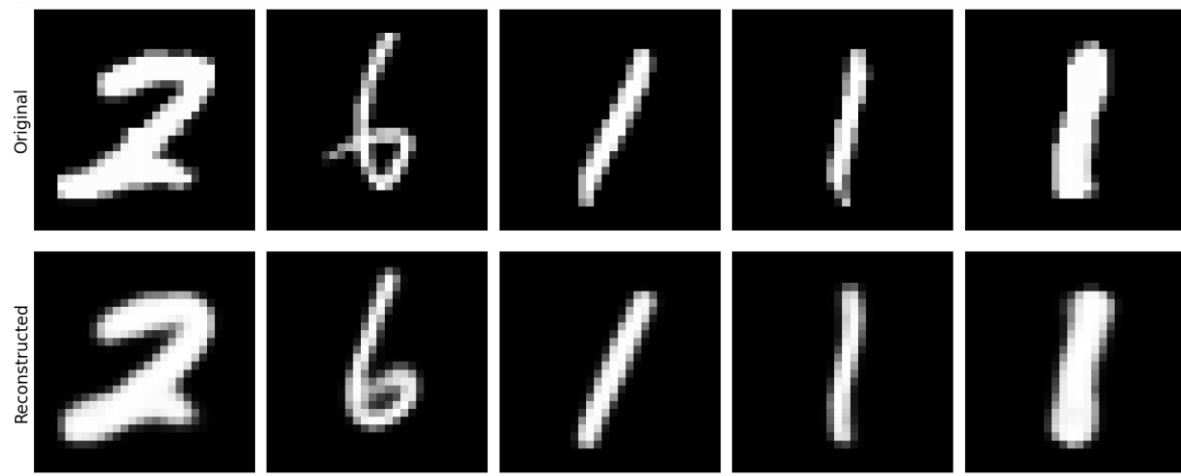
$$\varepsilon \sim \mathcal{N}(0, 1)$$

is an hyperparameter (although it changes at each forward pass) therefore it is constant to the derivative.

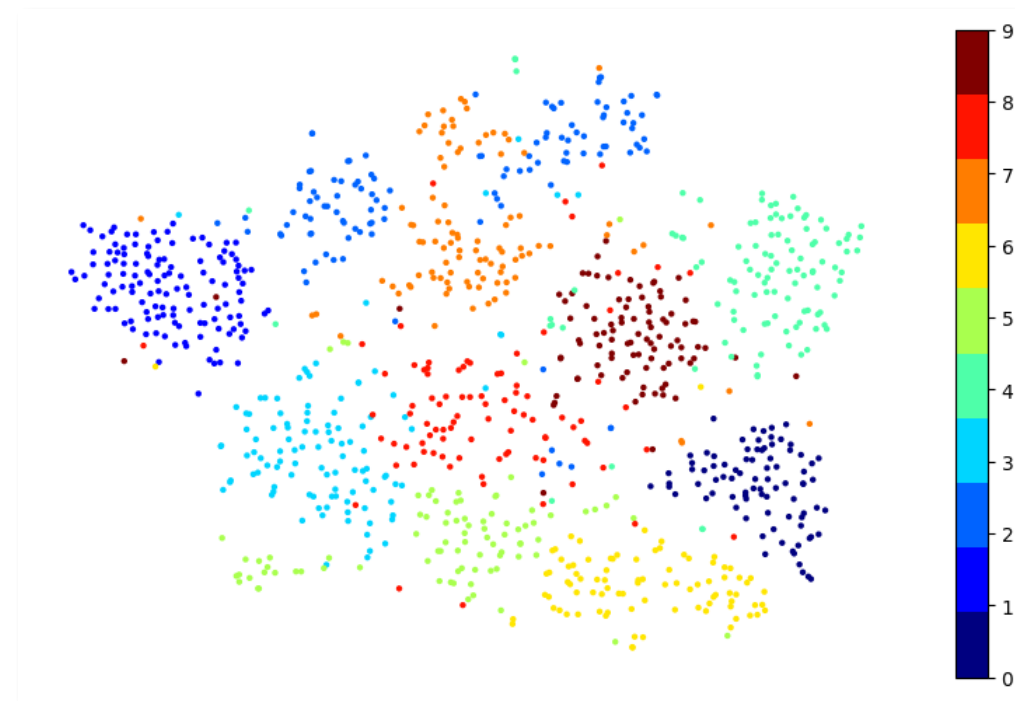
In plain words, during training and per each data item $\mathbf{x}^{(i)}$ the system draws one random value ε and computes the derivatives

VAE MNIST Experiments

Reconstruction



Latent space (2D TSNE)



Links

<https://johfischer.com/2022/09/18/denoising-score-matching/>

<https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>

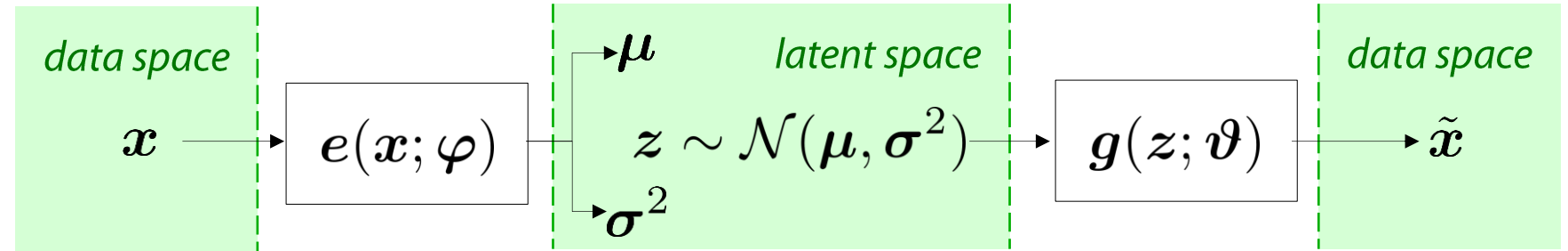
https://en.wikipedia.org/wiki/Variational_autoencoder

<https://mbernste.github.io/posts/vae/>

*Gaussian-Mixture
Variational Auto-Encoder
(GMVAE)*

One VAE limitation: unimodal arrangement of latent space

Variational Auto-Encoder:
use a Gaussian spread function to **organize** the latent space



This is what we want to train from a real dataset $D = \{\mathbf{x}_r^{(i)}\}_{i=1}^{N_r}$

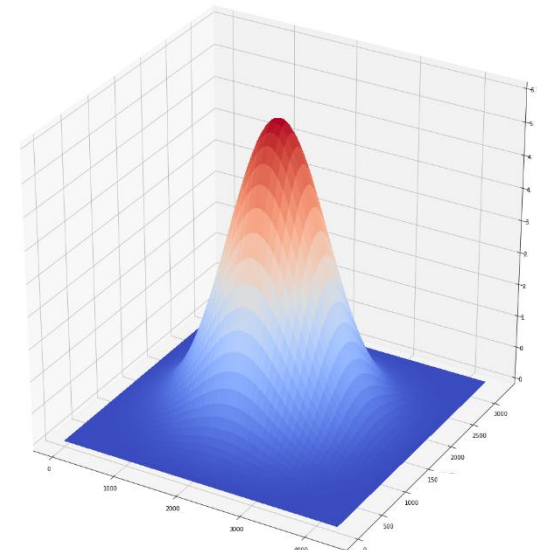
$$L(\mathbf{x}; \varphi, \vartheta) := \text{KL}(q(\mathbf{z} | \mathbf{x}, \varphi) \| p(\mathbf{z})) + \frac{1}{2} \frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|^2}{c}$$

Design choices

$$q(\mathbf{z} | \mathbf{x}, \varphi) := \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}; \varphi), \boldsymbol{\sigma}^2(\mathbf{x}; \varphi)\mathbf{I})$$

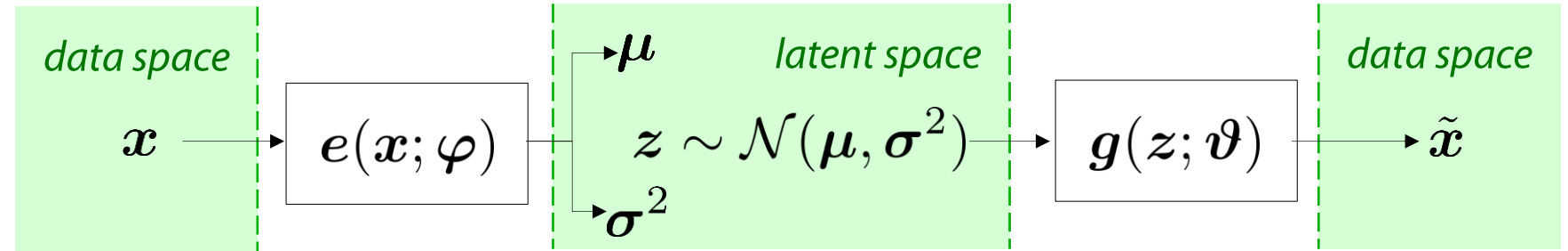
$$p(\mathbf{z}) := \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Unimodal distribution



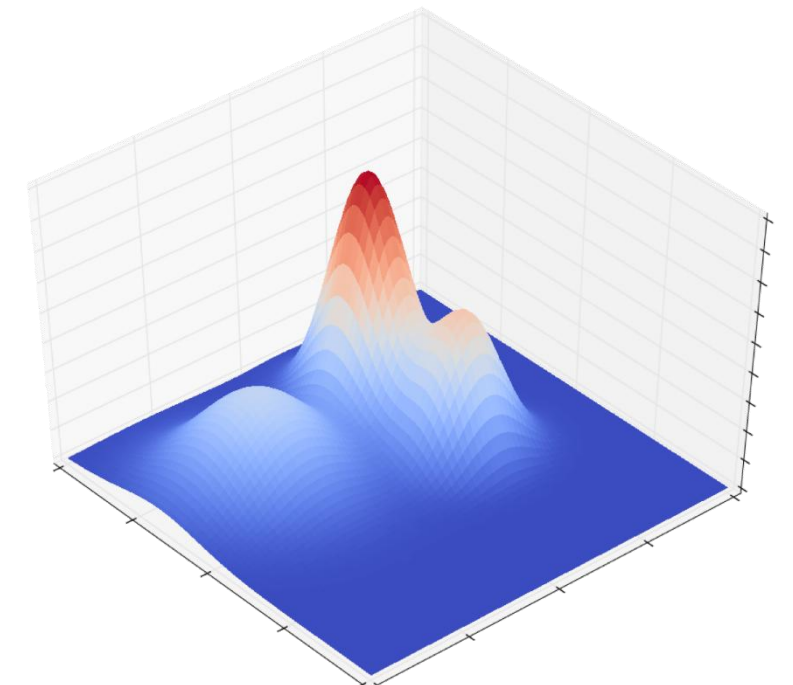
One VAE limitation: unimodal arrangement of latent space

Variational Auto-Encoder:
use a Gaussian spread function to **organize** the latent space

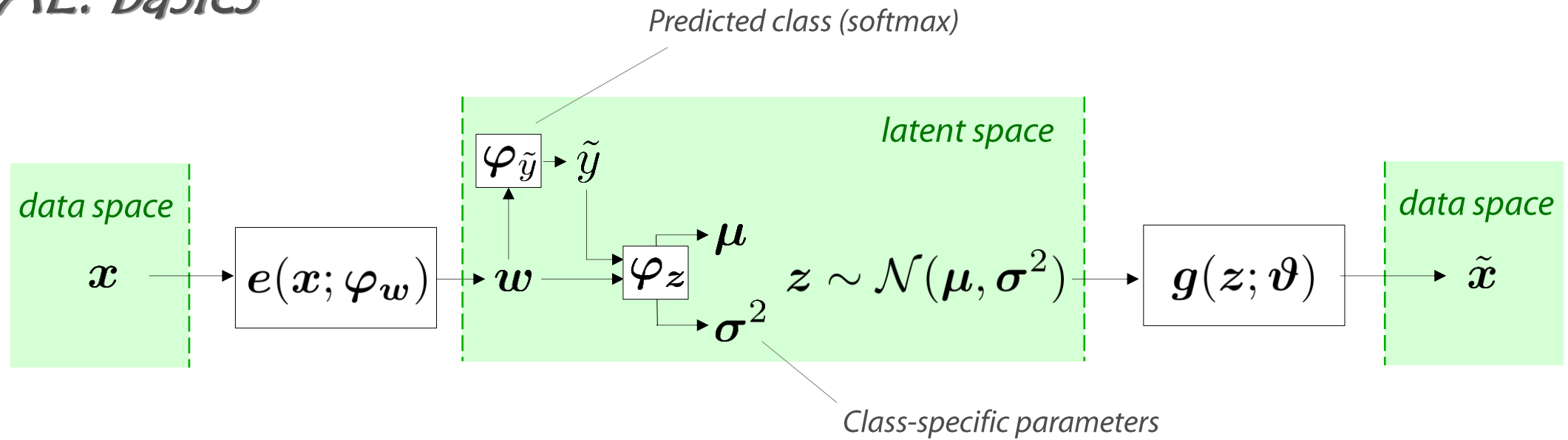


We may be wanting a more flexible arrangement of latent space

Multimodal distribution



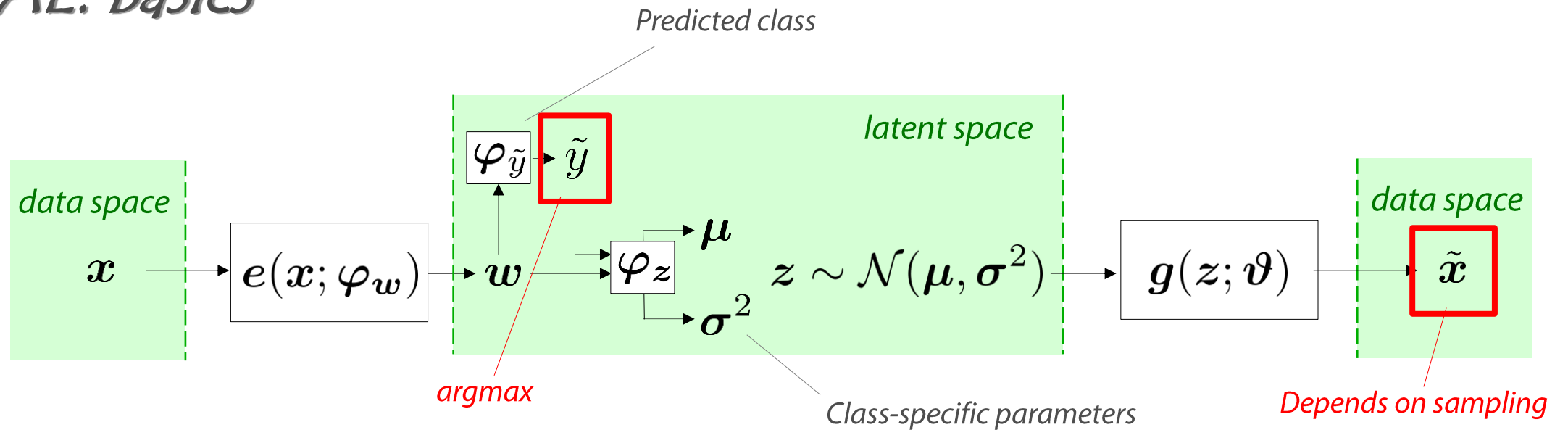
GMVAE: Basics



Training dataset $D = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$

Class label

GMVAE: Basics

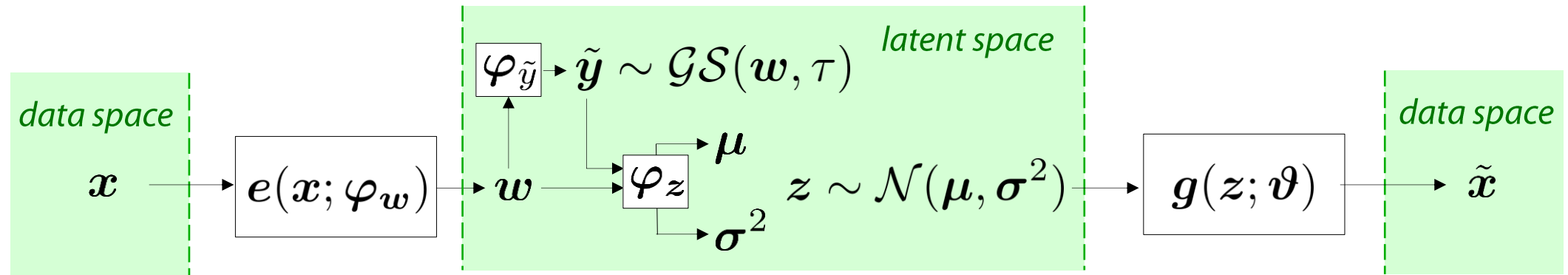


Training dataset $D = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$

Class label

Similar problem as before:
'argmax' prevents backpropagation

GMVAE: Gumbel-Softmax



Yet another reparametrization trick:

$$\lambda = \text{logit}(w, \varphi_{\tilde{y}}) \in \mathbb{R}^k$$

Number of classes

$$\xi = -\log(-\log u), \quad u \sim \mathcal{U}(0, 1; k) \in \mathbb{R}^k$$

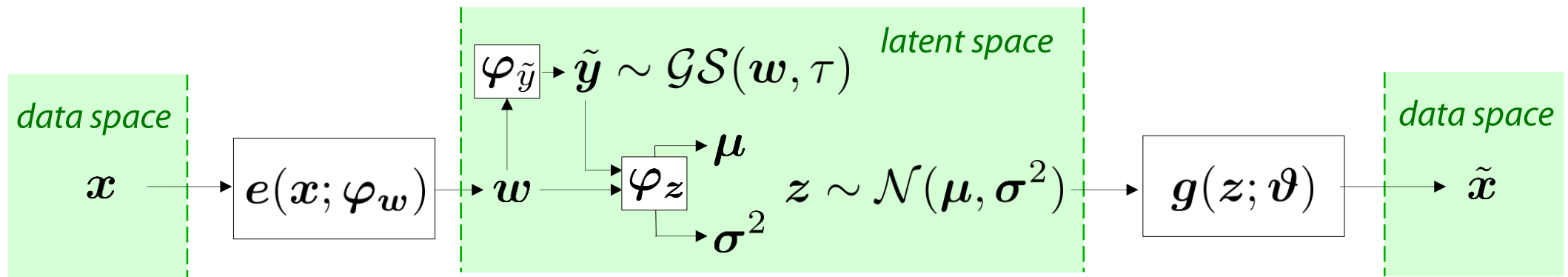
Gumbel distribution

Uniform distribution over [0, 1]

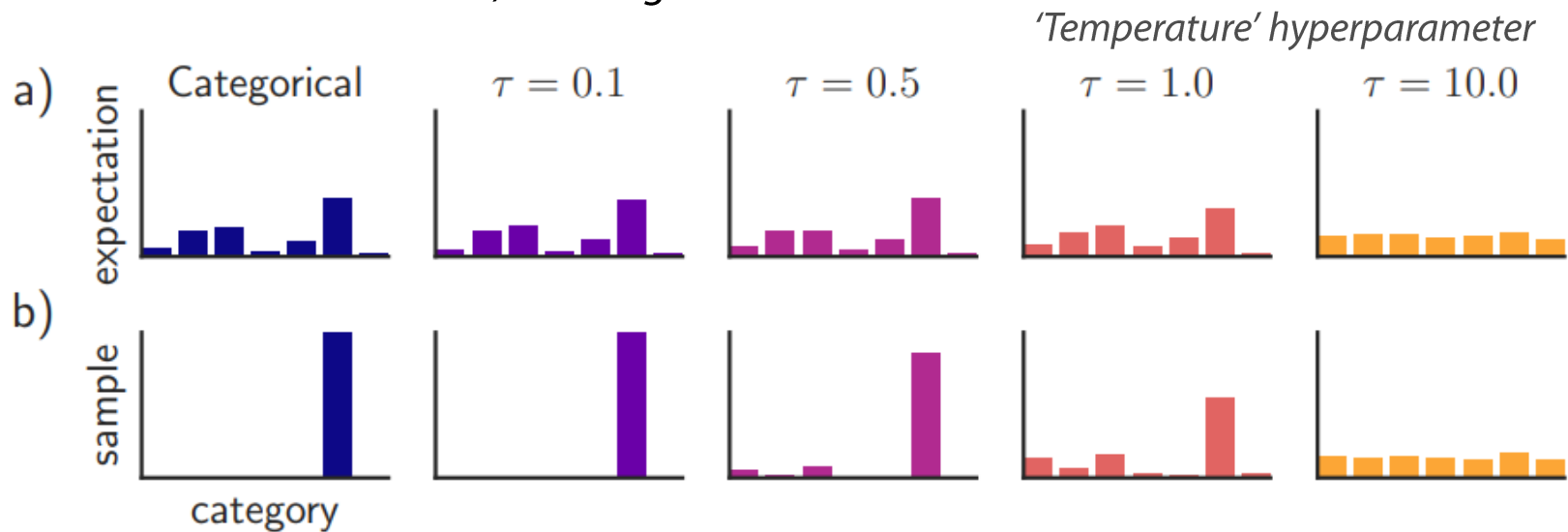
$$\tilde{y} = \mathcal{GS}(w, \tau) = \frac{\exp((\lambda + \xi)/\tau)}{\sum \exp((\lambda + \xi)/\tau)}$$

'Temperature' hyperparameter

GMVAE: Gumbel-Softmax



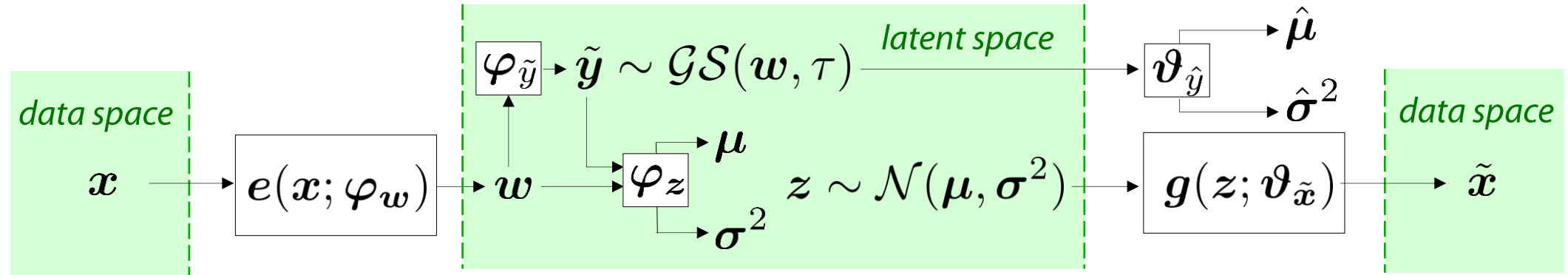
Gumbel-Softmax: derivable, 'soft-argmax'



The result is a 'soft one-hot encoding' of the argmax

[image from <https://arxiv.org/pdf/1611.01144>]

GMVAE: Loss Function



$$L(x; \varphi, \vartheta) := \underbrace{c_r \|\mathbf{x} - \tilde{\mathbf{x}}\|^2}_{\text{Reconstruction loss}} + \underbrace{c_g |\log \mathcal{N}(z, \mu, \sigma^2) - \log \mathcal{N}(z, \hat{\mu}, \hat{\sigma}^2)|}_{\text{'Gaussian' loss}} + \underbrace{c_e H(\tilde{\mathbf{y}})}_{\text{Entropy}}$$

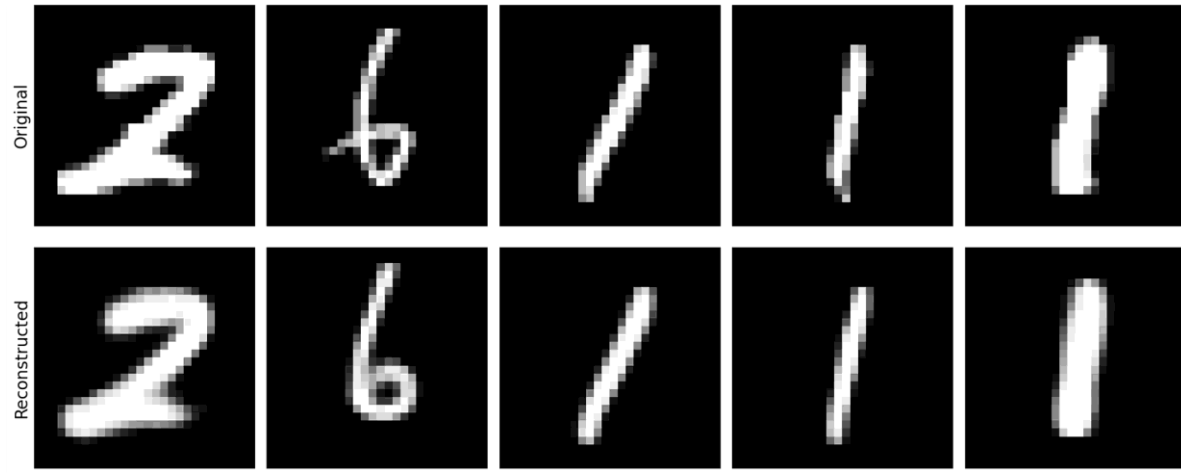
where:

$$H(\tilde{\mathbf{y}}) = - \sum_{i=1}^k \tilde{y}_i \log \tilde{y}_i$$

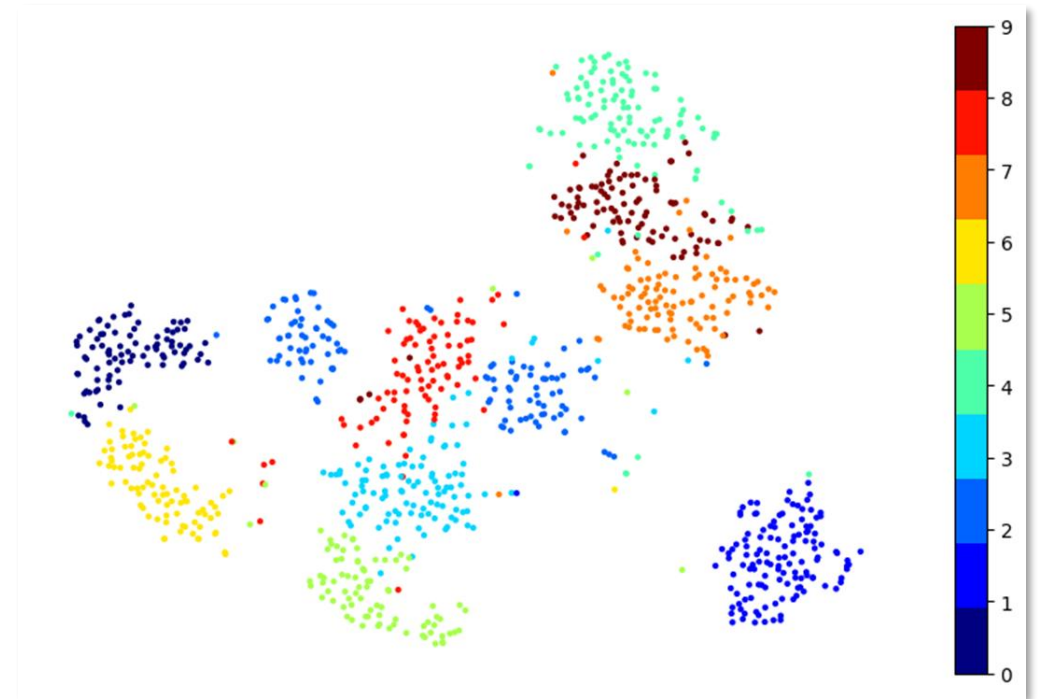
$\tilde{\mathbf{y}}$ is generated by a softmax:
each \tilde{y}_i is a probability

GMVAE MNIST Experiments

Reconstruction

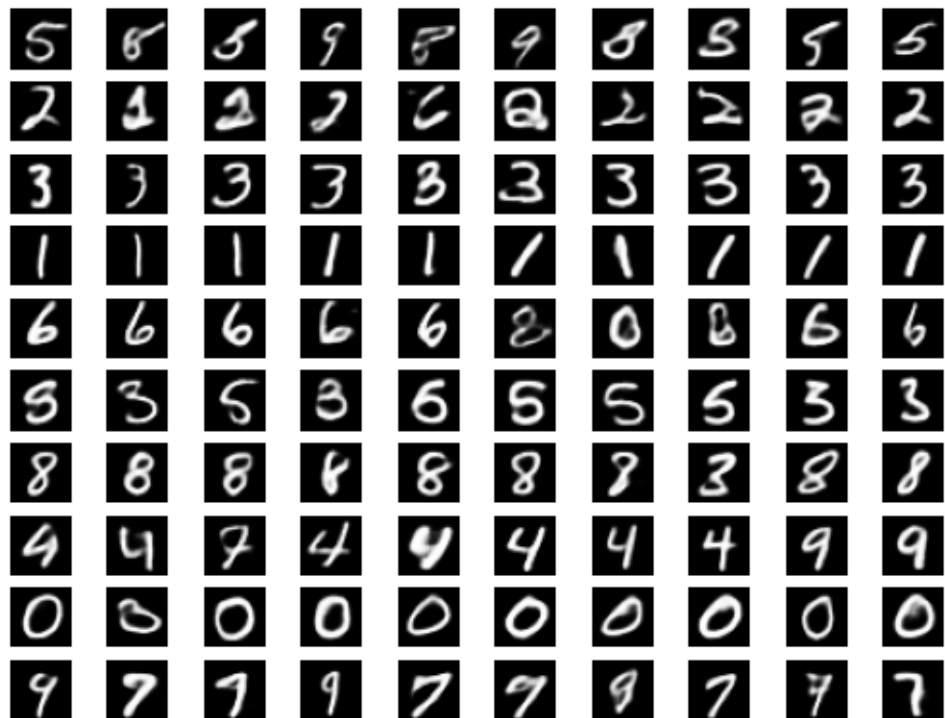


Latent space (2D TSNE)

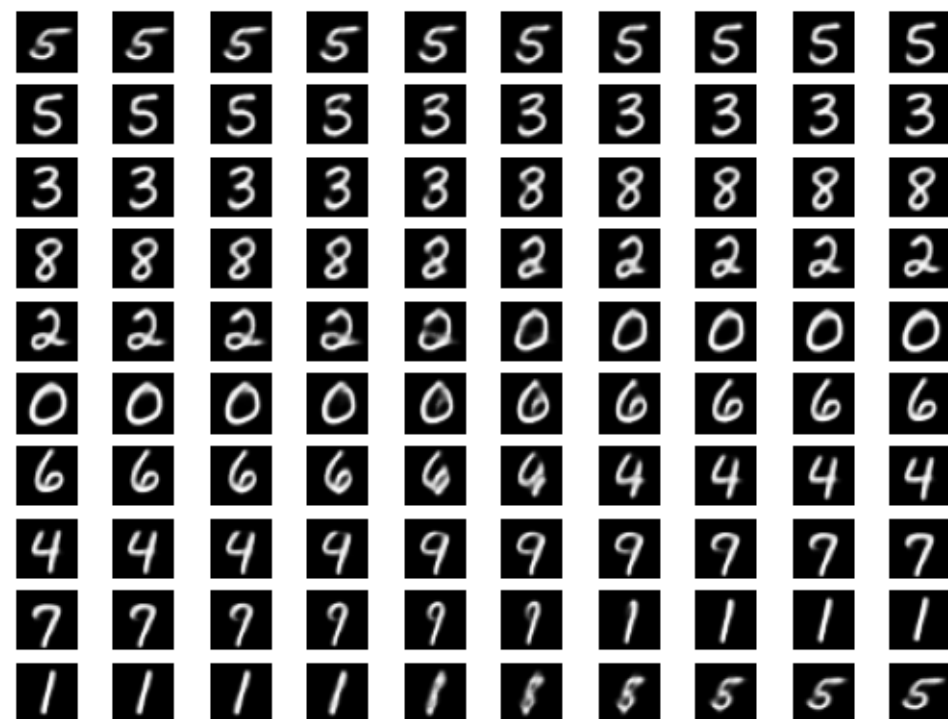


GMVAE MNIST Experiments

Random sampling (for each class μ and σ)



Convex interpolation (TSP tour)



Links

<https://arxiv.org/pdf/1406.5298>

<https://arxiv.org/pdf/1611.05148>

<https://ruishu.io/2016/12/25/gmvae/>

<https://arxiv.org/pdf/2112.00976>