

# Deep Learning & Applied AI

PCA and VAEs

Emanuele Rodolà  
[rodola@di.uniroma1.it](mailto:rodola@di.uniroma1.it)



SAPIENZA  
UNIVERSITÀ DI ROMA

2nd semester a.y. 2023/2024 · April 22, 2024

# Generative models

Overall idea:

Learn a **distribution** from some given training samples, and generate new samples from the same distribution.

# Generative models

Overall idea:

Learn a **distribution** from some given training samples, and generate new samples from the same distribution.



<https://this-person-does-not-exist.com/>

# Generative models

What does it mean to learn a **distribution**?

$$\mathbb{R}^{320 \times 240}$$

# Generative models

What does it mean to learn a **distribution**?

$x$   
•

$\mathbb{R}^{320 \times 240}$

# Generative models

What does it mean to learn a **distribution**?



$\mathbb{R}^{320 \times 240}$

# Generative models

What does it mean to learn a **distribution**?



$\mathbb{R}^{320 \times 240}$

# Generative models

What does it mean to learn a **distribution**?



$\mathbb{R}^{320 \times 240}$

# Generative models

What does it mean to learn a **distribution**?



•



$\mathbb{R}^{320 \times 240}$

# Generative models

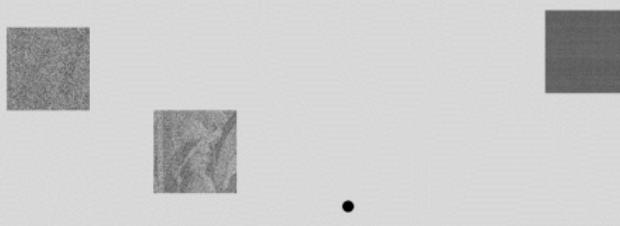
What does it mean to learn a **distribution**?



$\mathbb{R}^{320 \times 240}$

# Generative models

What does it mean to learn a **distribution**?



$\mathbb{R}^{320 \times 240}$

# Generative models

What does it mean to learn a **distribution**?



$\mathbb{R}^{320 \times 240}$

# Generative models

What does it mean to learn a **distribution**?



$\mathbb{R}^{320 \times 240}$

# Generative models

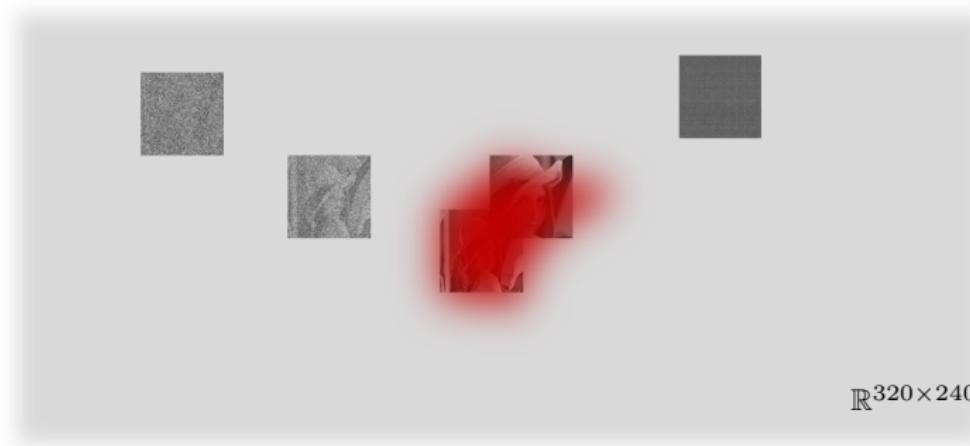
What does it mean to learn a **distribution**?



$\mathbb{R}^{320 \times 240}$

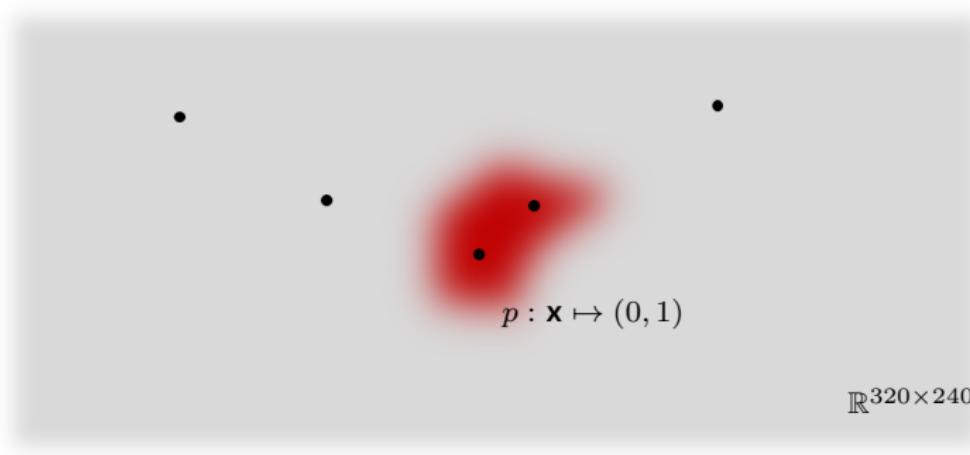
# Generative models

What does it mean to learn a **distribution**?



# Generative models

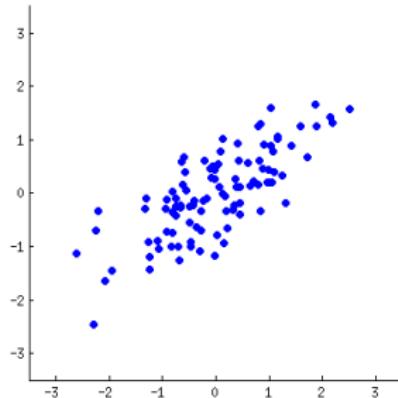
What does it mean to learn a **distribution**?



Probability distribution  $p$  defined on the entire data space.

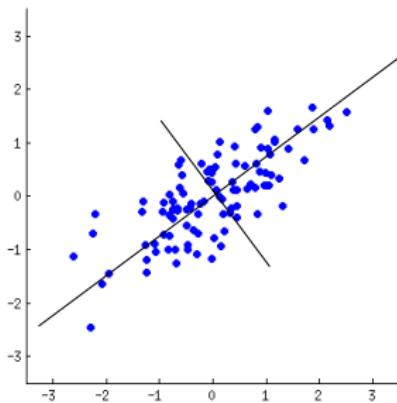
# Principal component analysis (PCA)

Regard our data as  $n$  points in  $\mathbb{R}^d$ :



# Principal component analysis (PCA)

Regard our data as  $n$  points in  $\mathbb{R}^d$ :

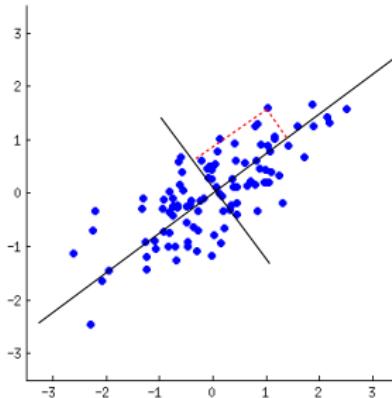


Overall idea:

- Find  $k \leq d$  **orthogonal directions** with the most variance.

# Principal component analysis (PCA)

Regard our data as  $n$  points in  $\mathbb{R}^d$ :



Overall idea:

- Find  $k \leq d$  orthogonal directions with the most variance.
- Project all the data points onto these directions.

# Principal component analysis (PCA)

In matrix notation:

$$\underbrace{\begin{pmatrix} \text{---} & \mathbf{x}_1^\top & \text{---} \\ \text{---} & \vdots & \text{---} \\ \text{---} & \mathbf{x}_n^\top & \text{---} \end{pmatrix}}_{n \times d}$$

# Principal component analysis (PCA)

In matrix notation:

$$\underbrace{\begin{pmatrix} \text{---} & \mathbf{x}_1^\top & \text{---} \\ \vdots & & \\ \text{---} & \mathbf{x}_n^\top & \text{---} \end{pmatrix}}_{n \times d} \quad \underbrace{\begin{pmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_k \\ | & & | \end{pmatrix}}_{d \times k}$$

# Principal component analysis (PCA)

In matrix notation:

$$\underbrace{\begin{pmatrix} \text{---} & \mathbf{x}_1^\top & \text{---} \\ \text{---} & \vdots & \text{---} \\ \text{---} & \mathbf{x}_n^\top & \text{---} \end{pmatrix}}_{n \times d} \underbrace{\begin{pmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_k \\ | & & | \end{pmatrix}}_{d \times k} = \underbrace{\begin{pmatrix} \text{---} & \mathbf{z}_1^\top & \text{---} \\ \text{---} & \vdots & \text{---} \\ \text{---} & \mathbf{z}_n^\top & \text{---} \end{pmatrix}}_{n \times k}$$

# Principal component analysis (PCA)

In matrix notation:

$$\underbrace{\begin{pmatrix} \text{---} & \mathbf{x}_1^\top & \text{---} \\ \text{---} & \vdots & \text{---} \\ \text{---} & \mathbf{x}_n^\top & \text{---} \end{pmatrix}}_{n \times d} \underbrace{\begin{pmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_k \\ | & & | \end{pmatrix}}_{d \times k} = \underbrace{\begin{pmatrix} \text{---} & \mathbf{z}_1^\top & \text{---} \\ \text{---} & \vdots & \text{---} \\ \text{---} & \mathbf{z}_n^\top & \text{---} \end{pmatrix}}_{n \times k}$$

Assuming  $\mathbf{W}^\top \mathbf{W} = \mathbf{I}$ , for  $k = d$  we get:

$$\mathbf{X}^\top \mathbf{W} = \mathbf{Z}^\top$$

$$\mathbf{X} = \mathbf{WZ}$$

# Principal component analysis (PCA)

In matrix notation:

$$\underbrace{\begin{pmatrix} \text{---} & \mathbf{x}_1^\top & \text{---} \\ \text{---} & \vdots & \text{---} \\ \text{---} & \mathbf{x}_n^\top & \text{---} \end{pmatrix}}_{n \times d} \underbrace{\begin{pmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_k \\ | & & | \end{pmatrix}}_{d \times k} = \underbrace{\begin{pmatrix} \text{---} & \mathbf{z}_1^\top & \text{---} \\ \text{---} & \vdots & \text{---} \\ \text{---} & \mathbf{z}_n^\top & \text{---} \end{pmatrix}}_{n \times k}$$

Assuming  $\mathbf{W}^\top \mathbf{W} = \mathbf{I}$ , for  $k < d$  we get:

$$\mathbf{X}^\top \mathbf{W} = \mathbf{Z}^\top$$

$$\mathbf{X} \approx \mathbf{W}\mathbf{Z}$$

# Principal component analysis (PCA)

In matrix notation:

$$\underbrace{\begin{pmatrix} \_ & \mathbf{x}_1^\top & \_ \\ \_ & \vdots & \_ \\ \_ & \mathbf{x}_n^\top & \_ \end{pmatrix}}_{n \times d} \underbrace{\begin{pmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_k \\ | & & | \end{pmatrix}}_{d \times k} = \underbrace{\begin{pmatrix} \_ & \mathbf{z}_1^\top & \_ \\ \_ & \vdots & \_ \\ \_ & \mathbf{z}_n^\top & \_ \end{pmatrix}}_{n \times k}$$

Assuming  $\mathbf{W}^\top \mathbf{W} = \mathbf{I}$ , for  $k < d$  we get:

$$\mathbf{X}^\top \mathbf{W} = \mathbf{Z}^\top \quad \text{projection}$$

$$\mathbf{X} \approx \mathbf{W}\mathbf{Z} \quad \text{reconstruction}$$

# Principal component analysis (PCA)

In matrix notation:

$$\underbrace{\begin{pmatrix} \text{---} & \mathbf{x}_1^\top & \text{---} \\ \vdots & & \vdots \\ \text{---} & \mathbf{x}_n^\top & \text{---} \end{pmatrix}}_{n \times d} \underbrace{\begin{pmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_k \\ | & & | \end{pmatrix}}_{d \times k} = \underbrace{\begin{pmatrix} \text{---} & \mathbf{z}_1^\top & \text{---} \\ \vdots & & \vdots \\ \text{---} & \mathbf{z}_n^\top & \text{---} \end{pmatrix}}_{n \times k}$$

Assuming  $\mathbf{W}^\top \mathbf{W} = \mathbf{I}$ , for  $k < d$  we get:

$$\mathbf{X}^\top \mathbf{W} = \mathbf{Z}^\top \quad \text{projection}$$

$$\mathbf{X} \approx \mathbf{WZ} \quad \text{reconstruction}$$

We call the columns of  $\mathbf{W}$  principal components.

They are unknown and must be computed.

# Principal component analysis (PCA)

We seek the **direction  $w$**  (a column of  $\mathbf{W}$ ) that:

- Minimizes the **projection/reconstruction error**.
- Maximizes the **variance** of the projected data.

# Principal component analysis (PCA)

We seek the **direction  $w$**  (a column of  $W$ ) that:

- Minimizes the **projection/reconstruction error**.
- Maximizes the **variance** of the projected data.

## Principal component analysis (PCA)

Assume the data points  $\mathbf{X}$  are **centered** at zero.

For a given  $\mathbf{w}$ , the projection of all  $n$  points onto  $\mathbf{w}$  is  $\mathbf{X}^T \mathbf{w}$ .

# Principal component analysis (PCA)

Assume the data points  $\mathbf{X}$  are **centered** at zero.

For a given  $\mathbf{w}$ , the projection of all  $n$  points onto  $\mathbf{w}$  is  $\mathbf{X}^\top \mathbf{w}$ .

The **variance** to maximize is  $\|\mathbf{X}^\top \mathbf{w}\|_2^2$ :

$$(\mathbf{X}^\top \mathbf{w})^\top (\mathbf{X}^\top \mathbf{w})$$

# Principal component analysis (PCA)

Assume the data points  $\mathbf{X}$  are **centered** at zero.

For a given  $\mathbf{w}$ , the projection of all  $n$  points onto  $\mathbf{w}$  is  $\mathbf{X}^\top \mathbf{w}$ .

The **variance** to maximize is  $\|\mathbf{X}^\top \mathbf{w}\|_2^2$ :

$$(\mathbf{X}^\top \mathbf{w})^\top (\mathbf{X}^\top \mathbf{w}) = \mathbf{w}^\top (\mathbf{X} \mathbf{X}^\top) \mathbf{w}$$

# Principal component analysis (PCA)

Assume the data points  $\mathbf{X}$  are **centered** at zero.

For a given  $\mathbf{w}$ , the projection of all  $n$  points onto  $\mathbf{w}$  is  $\mathbf{X}^\top \mathbf{w}$ .

The **variance** to maximize is  $\|\mathbf{X}^\top \mathbf{w}\|_2^2$ :

$$(\mathbf{X}^\top \mathbf{w})^\top (\mathbf{X}^\top \mathbf{w}) = \mathbf{w}^\top \underbrace{(\mathbf{X} \mathbf{X}^\top)}_{\mathbf{C}} \mathbf{w}$$

where  $\mathbf{C} \in \mathbb{R}^{d \times d}$  is the symmetric **covariance matrix**.

# Principal component analysis (PCA)

Assume the data points  $\mathbf{X}$  are **centered** at zero.

For a given  $\mathbf{w}$ , the projection of all  $n$  points onto  $\mathbf{w}$  is  $\mathbf{X}^\top \mathbf{w}$ .

The **variance** to maximize is  $\|\mathbf{X}^\top \mathbf{w}\|_2^2$ :

$$(\mathbf{X}^\top \mathbf{w})^\top (\mathbf{X}^\top \mathbf{w}) = \mathbf{w}^\top \underbrace{(\mathbf{X} \mathbf{X}^\top)}_{\mathbf{C}} \mathbf{w}$$

where  $\mathbf{C} \in \mathbb{R}^{d \times d}$  is the symmetric **covariance matrix**.

We want to solve the problem:

$$\max_{\mathbf{w}} \mathbf{w}^\top \mathbf{C} \mathbf{w} \quad \text{s.t. } \|\mathbf{w}\|_2 = 1$$

# Principal component analysis (PCA)

Assume the data points  $\mathbf{X}$  are **centered** at zero.

For a given  $\mathbf{w}$ , the projection of all  $n$  points onto  $\mathbf{w}$  is  $\mathbf{X}^\top \mathbf{w}$ .

The **variance** to maximize is  $\|\mathbf{X}^\top \mathbf{w}\|_2^2$ :

$$(\mathbf{X}^\top \mathbf{w})^\top (\mathbf{X}^\top \mathbf{w}) = \mathbf{w}^\top \underbrace{(\mathbf{X} \mathbf{X}^\top)}_{\mathbf{C}} \mathbf{w}$$

where  $\mathbf{C} \in \mathbb{R}^{d \times d}$  is the symmetric **covariance matrix**.

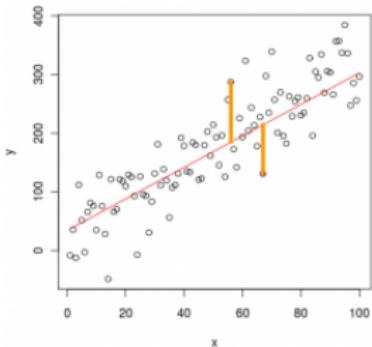
We want to solve the problem:

$$\max_{\mathbf{w}} \mathbf{w}^\top \mathbf{C} \mathbf{w} \quad \text{s.t. } \|\mathbf{w}\|_2 = 1$$

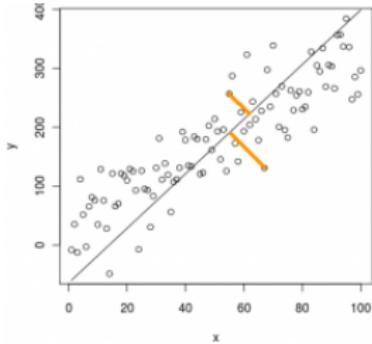
Solution:  $\mathbf{w}$  = principal **eigenvector** of  $\mathbf{C}$

# PCA is not linear regression

Linear regression: measure along the  $y$  coordinate:



PCA: measure orthogonal to the principal direction:



# PCA as a generative model

Given the  $\mathbf{W}$  satisfying:

$$\mathbf{X}^\top \mathbf{W} = \mathbf{Z}^\top \quad \text{projection}$$

$$\mathbf{X} \approx \mathbf{W}\mathbf{Z} \quad \text{reconstruction}$$

Generate new data just by sampling  $\mathbf{z}_{\text{new}} \in \mathbb{R}^k$ :

$$\mathbf{x}_{\text{new}} = \mathbf{W}\mathbf{z}_{\text{new}}$$

# PCA as a generative model

Given the  $\mathbf{W}$  satisfying:

$$\mathbf{x}^\top \mathbf{W} = \mathbf{z}^\top \quad \text{projection}$$

$$\mathbf{x} \approx \mathbf{W}\mathbf{z} \quad \text{reconstruction}$$

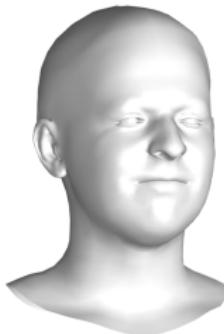
Generate new data just by sampling  $\mathbf{z}_{\text{new}} \in \mathbb{R}^k$ :

$$\mathbf{x}_{\text{new}} = \mathbf{W}\mathbf{z}_{\text{new}}$$

Example:



Data point  $\mathbf{x}_1$



Generated

$$\mathbf{x}_{\text{new}} = \frac{1}{2}\mathbf{W}(\mathbf{z}_1 + \mathbf{z}_2)$$



Data point  $\mathbf{x}_2$

# Codes

Consider again the relations:

$$\mathbf{W}^\top \mathbf{x} = \mathbf{z} \quad \text{projection}$$

$$\mathbf{x} \approx \mathbf{W}\mathbf{z} \quad \text{reconstruction}$$

New perspective: PCA as a [parametric model](#).

# Codes

Consider again the relations:

$$\begin{aligned} \mathbf{W}^\top \mathbf{x} &= \mathbf{z} && \text{encoding} \\ \mathbf{x} &\approx \mathbf{W}\mathbf{z} && \text{decoding} \end{aligned}$$

New perspective: PCA as a [parametric model](#).

Each data point  $\mathbf{x}$  is transformed into a low-dimensional code  $\mathbf{z} \in \mathbb{R}^k$ , where the dimension  $k < d$  is fixed.

[Encoding](#) and [decoding](#) are linear.

# Codes

Consider again the relations:

$$\begin{aligned} \mathbf{W}^\top \mathbf{x} &= \mathbf{z} && \text{encoding} \\ \mathbf{x} &\approx \mathbf{W}\mathbf{z} && \text{decoding} \end{aligned}$$

New perspective: PCA as a [parametric model](#).

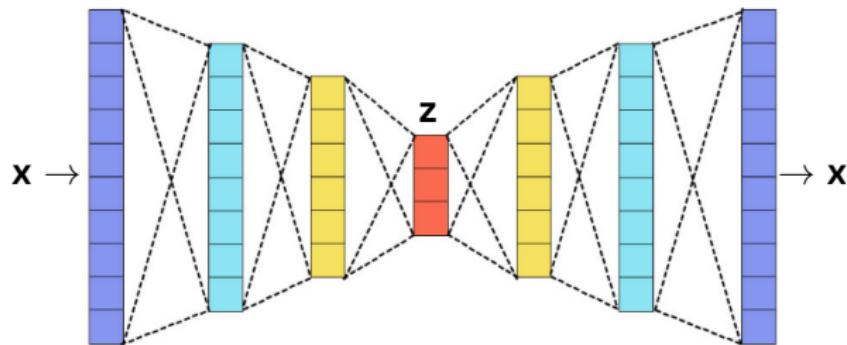
Each data point  $\mathbf{x}$  is transformed into a low-dimensional code  $\mathbf{z} \in \mathbb{R}^k$ , where the dimension  $k < d$  is fixed.

[Encoding](#) and [decoding](#) are linear.

How to generalize this idea?

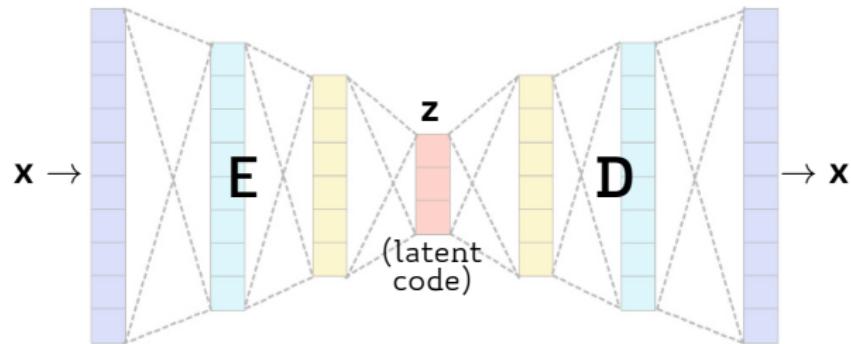
# Autoencoders (AE)

Implement encoding and decoding via deep nets.



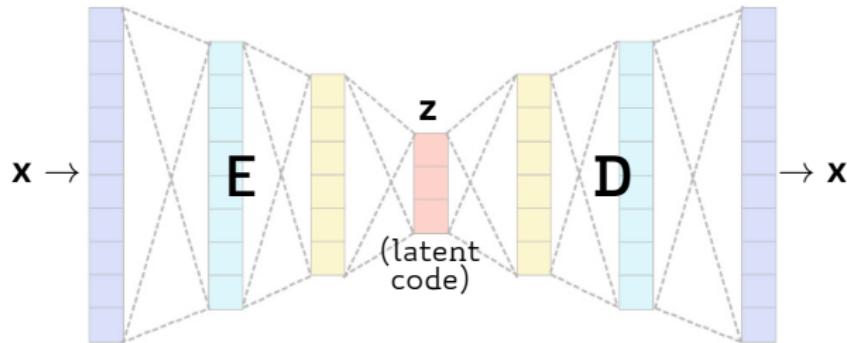
# Autoencoders (AE)

Implement encoding and decoding via deep nets.



# Autoencoders (AE)

Implement encoding and decoding via deep nets.

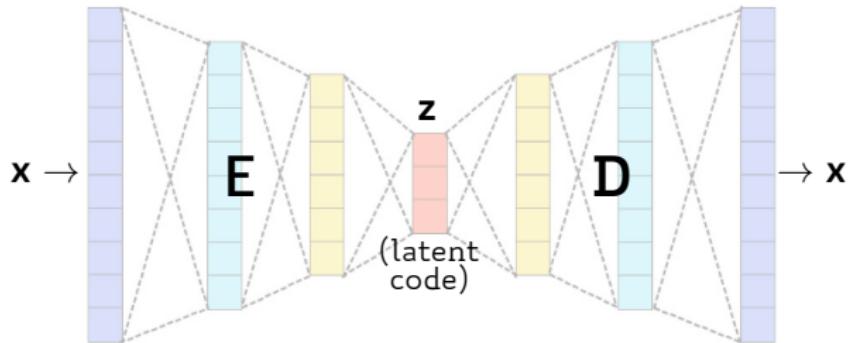


Given a dataset  $\{\mathbf{x}_i\}$ , require encoder  $E$  and decoder (or generator)  $D$  to minimize the **reconstruction loss**:

$$\ell_\Theta = \sum_i \|\mathbf{x}_i - D_\Theta(E_\Theta(\mathbf{x}_i))\|$$

# Autoencoders (AE)

Implement encoding and decoding via deep nets.



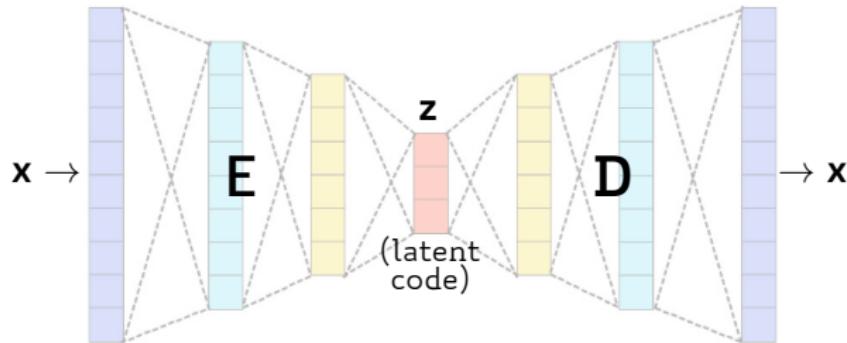
Given a dataset  $\{\mathbf{x}_i\}$ , require encoder  $E$  and decoder (or generator)  $D$  to minimize the **reconstruction loss**:

$$\ell_\Theta = \sum_i \|\mathbf{x}_i - D_\Theta(E_\Theta(\mathbf{x}_i))\|$$

The choice of the metric depends on the data.

# Autoencoders (AE)

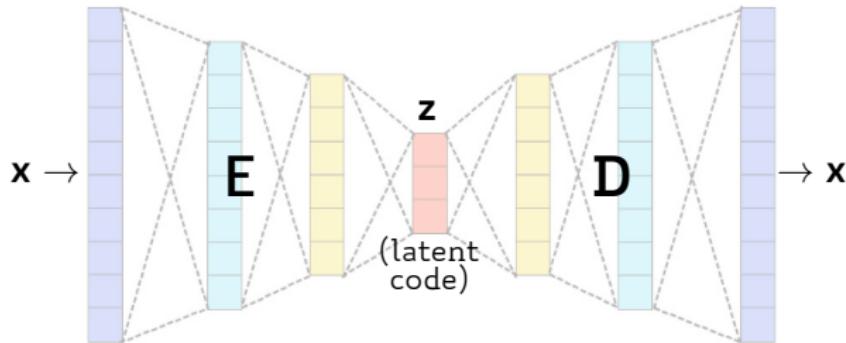
Implement encoding and decoding via deep nets.



- The **bottleneck** prevents trivial solutions.

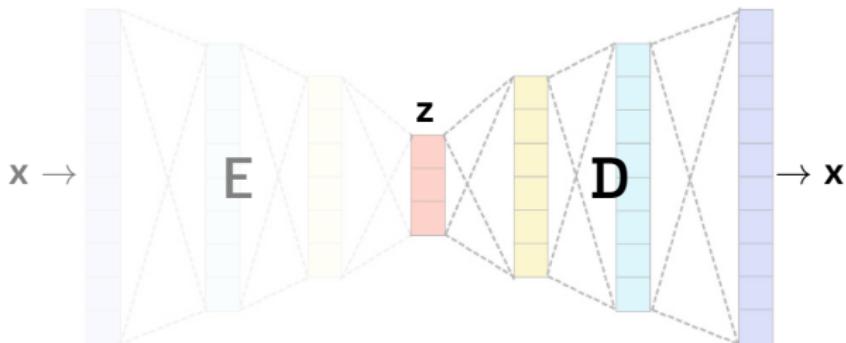
# Autoencoders (AE)

Implement encoding and decoding via deep nets.



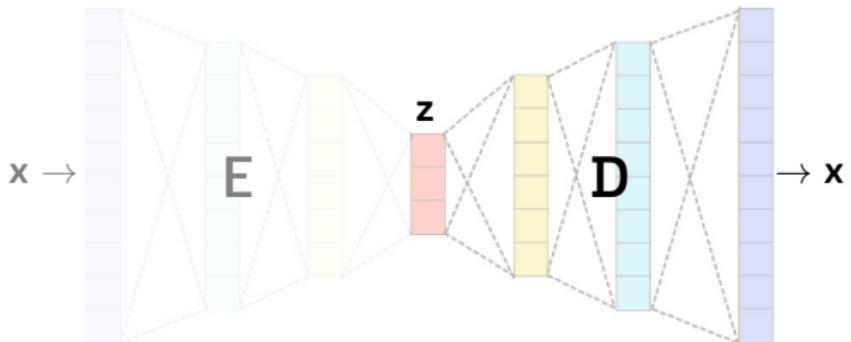
- The **bottleneck** prevents trivial solutions.
- **Task:** reconstruction.  
Once the AE is trained, study the structure of the **latent space** and use  $E, D$  for new tasks.

# Manifold hypothesis



The decoder maps from a low-dimensional **latent space** to a high-dimensional **embedding** of observed data.

# Manifold hypothesis



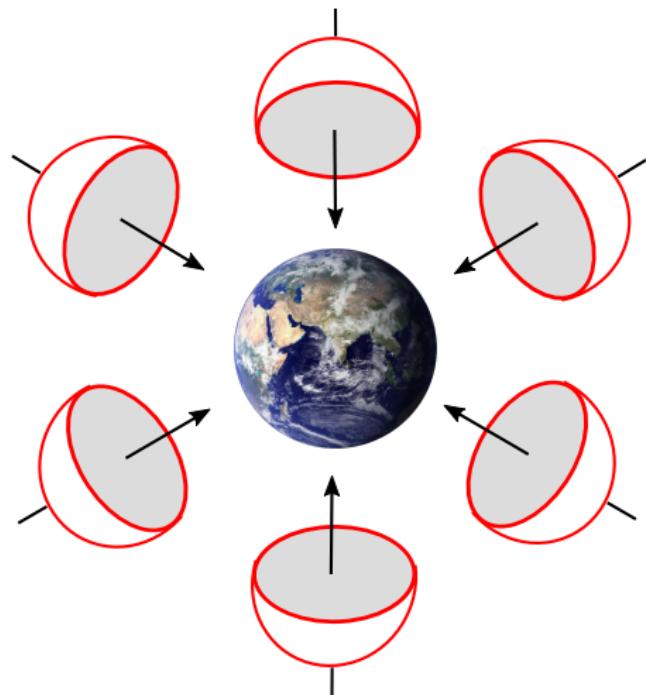
The decoder maps from a low-dimensional **latent space** to a high-dimensional **embedding** of observed data.

The latent space is Euclidean.

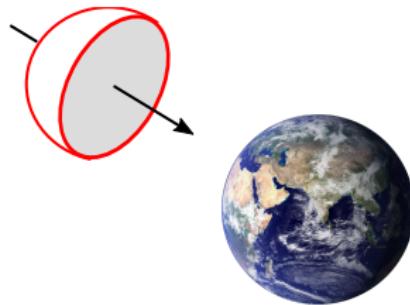
The data embedding space is curved (**manifold hypothesis**).

# Manifolds

Manifolds are unions of **charts**:



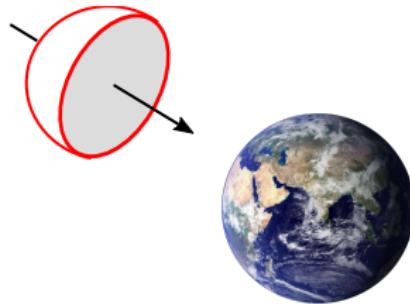
# 2D manifolds (surfaces)



chart

Each **chart** can be seen as a mapping  $\phi: \mathbb{R}^2 \rightarrow \mathcal{S} \subset \mathbb{R}^3$ .

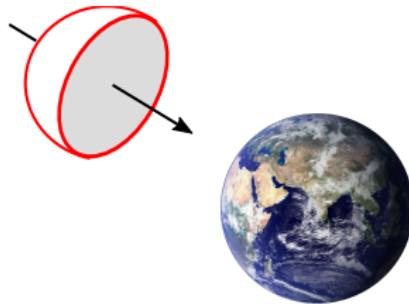
# 2D manifolds (surfaces)



chart

Each **chart** can be seen as a mapping  $\phi: \mathbb{R}^2 \rightarrow \mathcal{S} \subset \mathbb{R}^3$ .  
 $\phi$  must be **smooth** and **invertible** (diffeomorphism).

# 2D manifolds (surfaces)

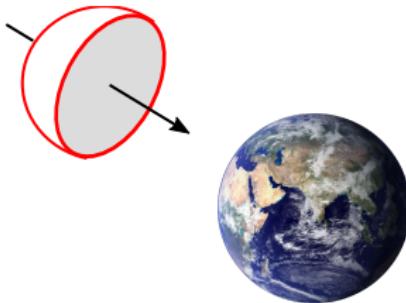


chart

Each **chart** can be seen as a mapping  $\phi: \mathbb{R}^2 \rightarrow \mathcal{S} \subset \mathbb{R}^3$ .  
 $\phi$  must be **smooth** and **invertible** (diffeomorphism).

- The domain of  $\phi$  is the **parametric space** and is Euclidean.

# 2D manifolds (surfaces)



chart

Each **chart** can be seen as a mapping  $\phi: \mathbb{R}^2 \rightarrow \mathcal{S} \subset \mathbb{R}^3$ .  
 $\phi$  must be **smooth** and **invertible** (diffeomorphism).

- The domain of  $\phi$  is the **parametric space** and is Euclidean.
- The image of  $\phi$  is the **embedding** and is a surface.

# Manifolds

Manifolds can be  $k$ -dimensional:

$$\phi : \mathbb{R}^k \rightarrow \mathcal{M} \subset \mathbb{R}^d \quad \text{with } k < d$$

# Manifolds

Manifolds can be  $k$ -dimensional:

$$\phi : \mathbb{R}^k \rightarrow \mathcal{M} \subset \mathbb{R}^d \quad \text{with } k < d$$

The parametrization is **not unique**:

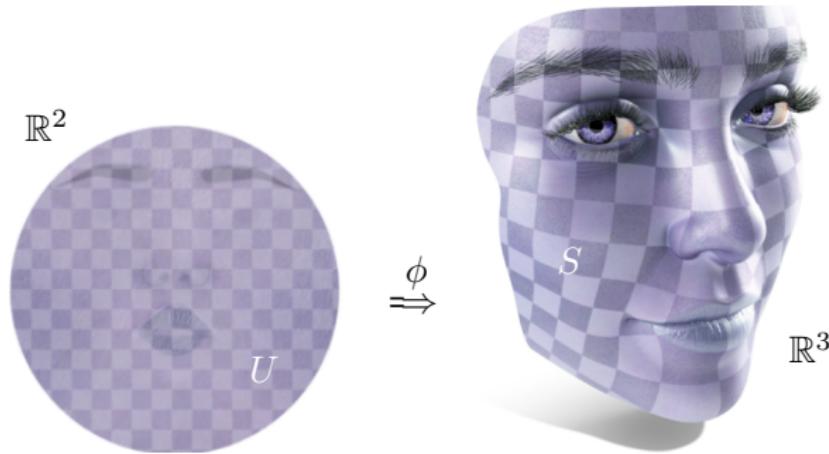


Figure by Keenan Crane

# Manifolds

Manifolds can be  $k$ -dimensional:

$$\phi : \mathbb{R}^k \rightarrow \mathcal{M} \subset \mathbb{R}^d \quad \text{with } k < d$$

The parametrization is **not unique**:

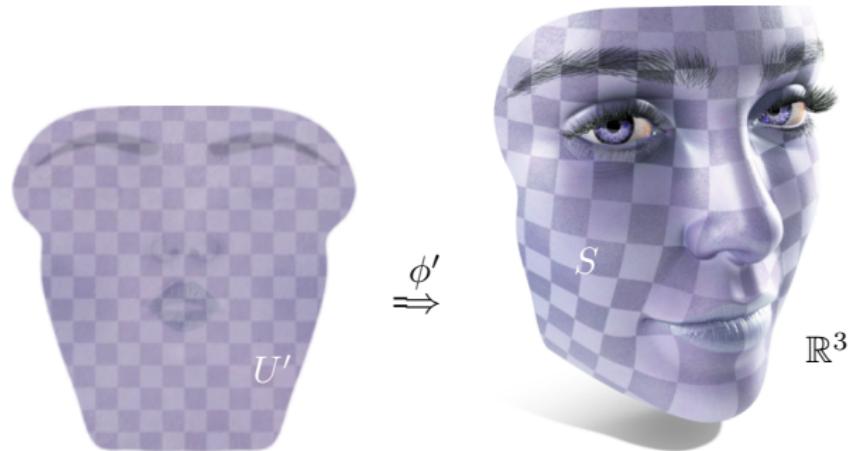


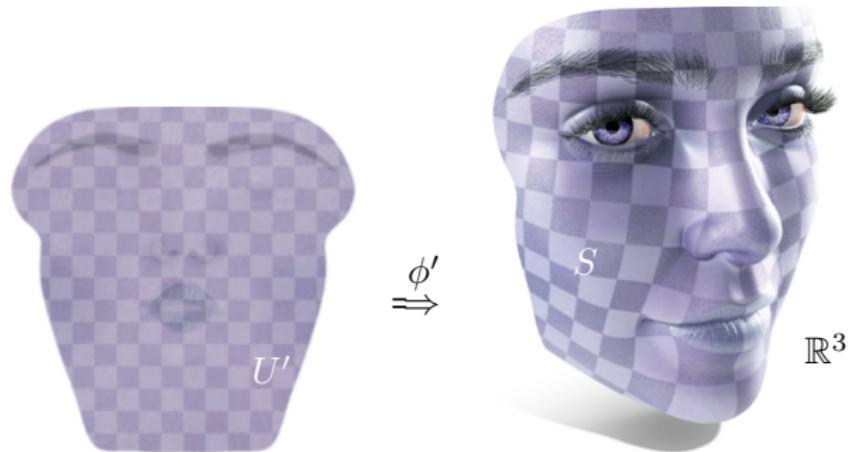
Figure by Keenan Crane

# Manifolds

Manifolds can be  $k$ -dimensional:

$$\phi : \mathbb{R}^k \rightarrow \mathcal{M} \subset \mathbb{R}^d \quad \text{with } k < d$$

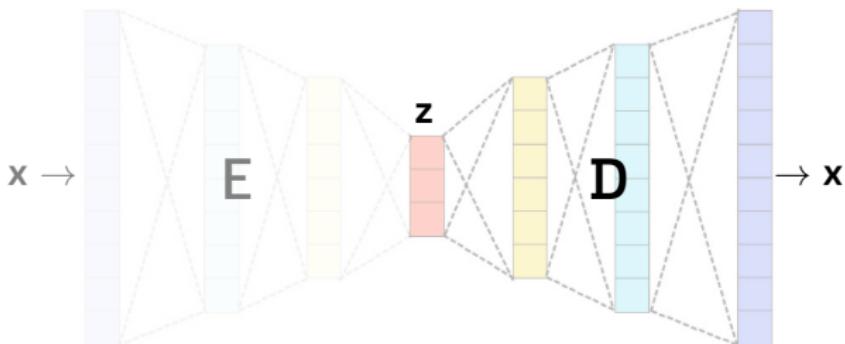
The parametrization is **not unique**:



They all encode **the same** geometric information.

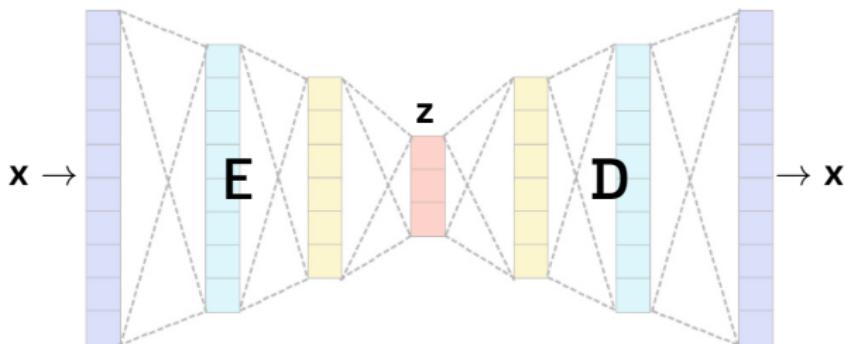
Figure by Keenan Crane

# Manifolds and generative models



The decoder  $D : \mathbb{R}^k \rightarrow \mathbb{R}^d$  is a chart from the latent space spanned by the codes  $\mathbf{z}$  to the data space of the inputs  $\mathbf{x}$ .

# Manifolds and generative models

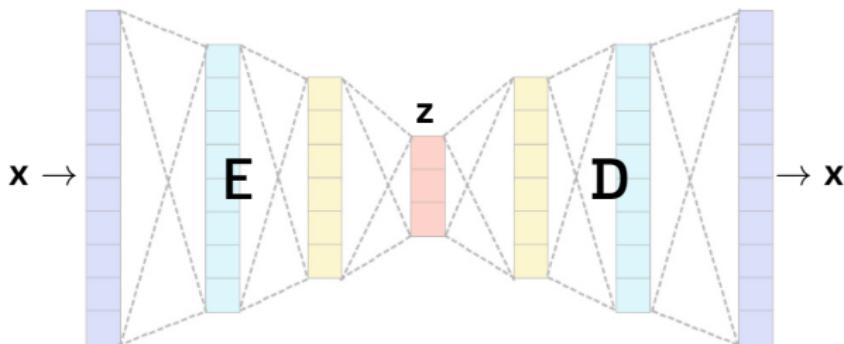


The decoder  $D : \mathbb{R}^k \rightarrow \mathbb{R}^d$  is a chart from the latent space spanned by the codes  $\mathbf{z}$  to the data space of the inputs  $\mathbf{x}$ .

It is differentiable.

It is invertible via the encoder  $E$ .

# Manifolds and generative models



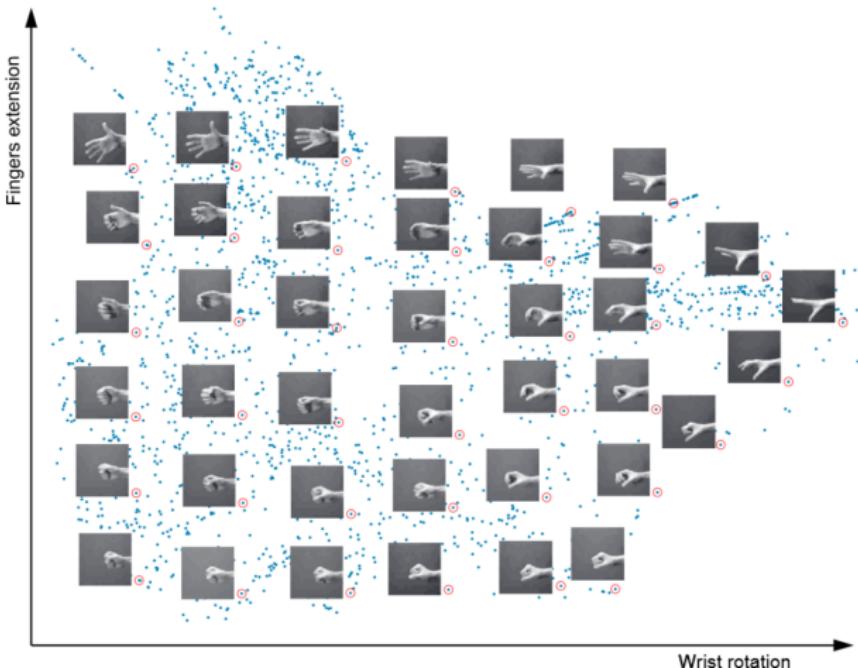
The decoder  $D : \mathbb{R}^k \rightarrow \mathbb{R}^d$  is a chart from the latent space spanned by the codes  $\mathbf{z}$  to the data space of the inputs  $\mathbf{x}$ .

It is differentiable.

It is invertible via the encoder  $E$ .

PCA puts the data on a **linear** (flat) manifold, since  $D$  simply performs a linear combination of orthogonal vectors.

# Manifolds and generative models



Have a look at the notebook:

[https://colab.research.google.com/github/erodola/ML-s2-2024/blob/main/labs/05\\_PCA.ipynb](https://colab.research.google.com/github/erodola/ML-s2-2024/blob/main/labs/05_PCA.ipynb)

# Variational autoencoders (VAE)

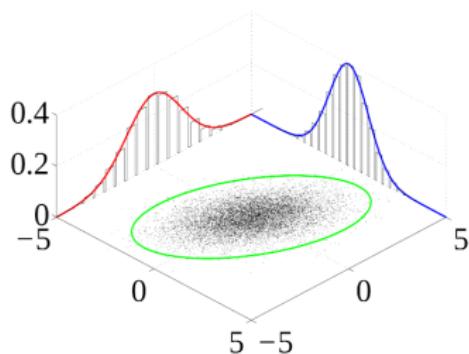
AEs can map similar inputs to dissimilar latent codes.

# Variational autoencoders (VAE)

AEs can map similar inputs to dissimilar latent codes.

A **variational** AE learns a probability distribution on the latent space.

- The data is seen as a **sampling** of the distribution.
- The distribution is fixed and decided **a priori** (e.g. Gaussian).



# Entropy and divergence

The information carried by an event  $\mathbf{x}$  can be quantified as:

$$I(\mathbf{x}) = -\log p(\mathbf{x})$$

# Entropy and divergence

The information carried by an event  $\mathbf{x}$  can be quantified as:

$$I(\mathbf{x}) = -\log p(\mathbf{x})$$

The average information encoded in  $p$  is its **entropy**:

$$H(p) = - \sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

# Entropy and divergence

The information carried by an event  $\mathbf{x}$  can be quantified as:

$$I(\mathbf{x}) = -\log p(\mathbf{x})$$

The average information encoded in  $p$  is its [entropy](#):

$$H(p) = - \sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

Given two distributions  $p$  and  $q$ , the [Kullback-Leibler divergence](#):

$$KL(p\|q) \approx H(q) - H(p)$$

measures their dissimilarity in terms of their entropy.

# Entropy and divergence

The information carried by an event  $\mathbf{x}$  can be quantified as:

$$I(\mathbf{x}) = -\log p(\mathbf{x})$$

The average information encoded in  $p$  is its **entropy**:

$$H(p) = - \sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

Given two distributions  $p$  and  $q$ , the **Kullback-Leibler divergence**:

$$KL(p\|q) \approx - \sum q(\mathbf{x}) \log q(\mathbf{x}) + \sum p(\mathbf{x}) \log p(\mathbf{x})$$

measures their dissimilarity in terms of their entropy.

# Entropy and divergence

The information carried by an event  $\mathbf{x}$  can be quantified as:

$$I(\mathbf{x}) = -\log p(\mathbf{x})$$

The average information encoded in  $p$  is its **entropy**:

$$H(p) = - \sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

Given two distributions  $p$  and  $q$ , the **Kullback-Leibler divergence**:

$$KL(p\|q) = - \sum_{\mathbf{x}} p(\mathbf{x}) \log q(\mathbf{x}) + \sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x}) \geq 0$$

measures their dissimilarity in terms of their entropy.

# Entropy and divergence

The information carried by an event  $\mathbf{x}$  can be quantified as:

$$I(\mathbf{x}) = -\log p(\mathbf{x})$$

The average information encoded in  $p$  is its [entropy](#):

$$H(p) = - \sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

Given two distributions  $p$  and  $q$ , the [Kullback-Leibler divergence](#):

$$KL(p\|q) = \sum_{\mathbf{x}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} \geq 0$$

measures their dissimilarity in terms of their entropy.

# Entropy and divergence

The information carried by an event  $\mathbf{x}$  can be quantified as:

$$I(\mathbf{x}) = -\log p(\mathbf{x})$$

The average information encoded in  $p$  is its **entropy**:

$$H(p) = - \sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

Given two distributions  $p$  and  $q$ , the **Kullback-Leibler divergence**:

$$KL(p\|q) = - \sum_{\mathbf{x}} p(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} \geq 0$$

measures their dissimilarity in terms of their entropy.

# Variational inference

In our scenario:  $x$  is a data point, and  $z$  is a latent code.

## Variational inference

In our scenario:  $\textcolor{green}{x}$  is a data point, and  $\textcolor{red}{z}$  is a latent code.

Define a parametric [probabilistic encoder](#):

$$p_{\theta}(\textcolor{red}{z}|\textcolor{green}{x})$$

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric [probabilistic encoder](#):

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})}$$

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric [probabilistic encoder](#):

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric **probabilistic encoder**:

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

However, this integral:

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

is **intractable** over the entire **latent space**.

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric [probabilistic encoder](#):

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

Instead, compute an approximation:

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

where  $q_{\phi}(\mathbf{z}|\mathbf{x})$  is a neural network with weights  $\phi$ .

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric [probabilistic encoder](#):

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

Instead, compute an approximation  $q_{\phi^*}(\mathbf{z}|\mathbf{x}) \approx p_{\theta}(\mathbf{z}|\mathbf{x})$  with:

$$\phi^* = \arg \min_{\phi, \theta} KL(q_{\phi}(\mathbf{z}|\mathbf{x}) \| p_{\theta}(\mathbf{z}|\mathbf{x}))$$

where  $q_{\phi}(\mathbf{z}|\mathbf{x})$  is a neural network with weights  $\phi$ .

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric [probabilistic encoder](#):

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

Instead, compute an approximation  $q_{\phi^*}(\mathbf{z}|\mathbf{x}) \approx p_{\theta}(\mathbf{z}|\mathbf{x})$  with:

$$\phi^* = \arg \min_{\phi, \theta} - \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{z}|\mathbf{x})}{q_{\phi}(\mathbf{z}|\mathbf{x})}$$

where  $q_{\phi}(\mathbf{z}|\mathbf{x})$  is a neural network with weights  $\phi$ .

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric [probabilistic encoder](#):

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

Instead, compute an approximation  $q_{\phi^*}(\mathbf{z}|\mathbf{x}) \approx p_{\theta}(\mathbf{z}|\mathbf{x})$  with:

$$\phi^* = \arg \min_{\phi, \theta} - \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})/p_{\theta}(\mathbf{x})}{q_{\phi}(\mathbf{z}|\mathbf{x})}$$

where  $q_{\phi}(\mathbf{z}|\mathbf{x})$  is a neural network with weights  $\phi$ .

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric [probabilistic encoder](#):

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

Instead, compute an approximation  $q_{\phi^*}(\mathbf{z}|\mathbf{x}) \approx p_{\theta}(\mathbf{z}|\mathbf{x})$  with:

$$\phi^* = \arg \min_{\phi, \theta} - \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \frac{1}{p_{\theta}(\mathbf{x})}$$

where  $q_{\phi}(\mathbf{z}|\mathbf{x})$  is a neural network with weights  $\phi$ .

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric [probabilistic encoder](#):

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

Instead, compute an approximation  $q_{\phi^*}(\mathbf{z}|\mathbf{x}) \approx p_{\theta}(\mathbf{z}|\mathbf{x})$  with:

$$\phi^* = \arg \min_{\phi, \theta} - \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \left( \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} + \log \frac{1}{p_{\theta}(\mathbf{x})} \right)$$

where  $q_{\phi}(\mathbf{z}|\mathbf{x})$  is a neural network with weights  $\phi$ .

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric [probabilistic encoder](#):

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

Instead, compute an approximation  $q_{\phi^*}(\mathbf{z}|\mathbf{x}) \approx p_{\theta}(\mathbf{z}|\mathbf{x})$  with:

$$\phi^* = \arg \min_{\phi, \theta} - \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \left( \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} - \log p_{\theta}(\mathbf{x}) \right)$$

where  $q_{\phi}(\mathbf{z}|\mathbf{x})$  is a neural network with weights  $\phi$ .

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric [probabilistic encoder](#):

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

Instead, compute an approximation  $q_{\phi^*}(\mathbf{z}|\mathbf{x}) \approx p_{\theta}(\mathbf{z}|\mathbf{x})$  with:

$$\phi^* = \arg \min_{\phi, \theta} - \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} + \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x})$$

where  $q_{\phi}(\mathbf{z}|\mathbf{x})$  is a neural network with weights  $\phi$ .

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric [probabilistic encoder](#):

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

Instead, compute an approximation  $q_{\phi^*}(\mathbf{z}|\mathbf{x}) \approx p_{\theta}(\mathbf{z}|\mathbf{x})$  with:

$$\phi^* = \arg \min_{\phi, \theta} - \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} + \log p_{\theta}(\mathbf{x}) \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x})$$

where  $q_{\phi}(\mathbf{z}|\mathbf{x})$  is a neural network with weights  $\phi$ .

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric **probabilistic encoder**:

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

Instead, compute an approximation  $q_{\phi^*}(\mathbf{z}|\mathbf{x}) \approx p_{\theta}(\mathbf{z}|\mathbf{x})$  with:

$$\phi^* = \arg \min_{\phi, \theta} - \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} + \log p_{\theta}(\mathbf{x}) \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x})^1$$

where  $q_{\phi}(\mathbf{z}|\mathbf{x})$  is a neural network with weights  $\phi$ .

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric [probabilistic encoder](#):

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

Instead, compute an approximation  $q_{\phi^*}(\mathbf{z}|\mathbf{x}) \approx p_{\theta}(\mathbf{z}|\mathbf{x})$  with:

$$\phi^* = \arg \min_{\phi, \theta} - \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} + \log p_{\theta}(\mathbf{x})$$

where  $q_{\phi}(\mathbf{z}|\mathbf{x})$  is a neural network with weights  $\phi$ .

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric [probabilistic encoder](#):

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

Instead, compute an approximation  $q_{\phi^*}(\mathbf{z}|\mathbf{x}) \approx p_{\theta}(\mathbf{z}|\mathbf{x})$  with:

$$\phi^* = \arg \max_{\phi, \theta} \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} - \log p_{\theta}(\mathbf{x})$$

where  $q_{\phi}(\mathbf{z}|\mathbf{x})$  is a neural network with weights  $\phi$ .

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric **probabilistic encoder**:

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

Instead, compute an approximation  $q_{\phi^*}(\mathbf{z}|\mathbf{x}) \approx p_{\theta}(\mathbf{z}|\mathbf{x})$  with:

$$\phi^* = \arg \max_{\phi, \theta} \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} - \cancel{\log p_{\theta}(\mathbf{x})}^{\text{relax}}$$

where  $q_{\phi}(\mathbf{z}|\mathbf{x})$  is a neural network with weights  $\phi$ .

# Variational inference

In our scenario:  $\mathbf{x}$  is a data point, and  $\mathbf{z}$  is a latent code.

Define a parametric **probabilistic encoder**:

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

Instead, compute an approximation  $q_{\phi^*}(\mathbf{z}|\mathbf{x}) \approx p_{\theta}(\mathbf{z}|\mathbf{x})$  with:

$$\phi^* = \arg \max_{\phi, \theta} ELBO_{\phi, \theta}(\mathbf{x}) - \overbrace{\log p_{\theta}(\mathbf{x})}^{\text{relax}}$$

where  $q_{\phi}(\mathbf{z}|\mathbf{x})$  is a neural network with weights  $\phi$  and:

$$ELBO_{\phi, \theta}(\mathbf{x}) = \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \leq \log p_{\theta}(\mathbf{x})$$

is called the **Evidence variational Lower BOund**.

# Variational autoencoder (VAE)

$$\max_{\phi, \theta} ELBO_{\phi, \theta}(\mathbf{x})$$

# Variational autoencoder (VAE)

$$\max_{\phi, \theta} \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})}$$

# Variational autoencoder (VAE)

$$\max_{\phi, \theta} \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})}$$

# Variational autoencoder (VAE)

$$\max_{\phi, \theta} \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \left( \log p_{\theta}(\mathbf{x}|\mathbf{z}) + \log \frac{p_{\theta}(\mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right)$$

# Variational autoencoder (VAE)

$$\max_{\phi, \theta} \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{x}|\mathbf{z}) + \sum_{\mathbf{z}} q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})}$$

# Variational autoencoder (VAE)

$$\max_{\phi, \theta} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

# Variational autoencoder (VAE)

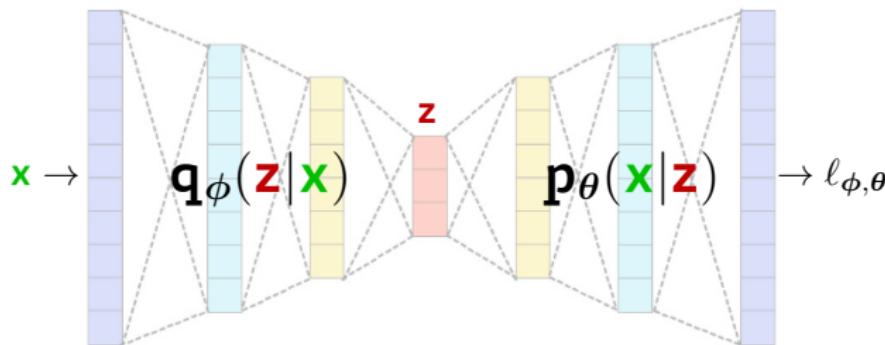
$$\max_{\phi, \theta} \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} - KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

# Variational autoencoder (VAE)

$$\max_{\phi, \theta} \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} - \underbrace{KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))}_{\text{ensures } q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z})}$$

# Variational autoencoder (VAE)

$$\ell_{\phi, \theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} + \underbrace{KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))}_{\text{ensures } q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z})}$$



# Variational autoencoder (VAE)

$$\ell_{\phi, \theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} + \underbrace{KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))}_{\text{ensures } q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z})}$$

Fixed Gaussian prior over the latent variables:

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

# Variational autoencoder (VAE)

$$\ell_{\phi, \theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} + \underbrace{KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))}_{\text{ensures } q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z})}$$

Fixed Gaussian prior over the latent variables:

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

The probabilistic encoder also generates a Gaussian distribution:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}_{\boldsymbol{\mu}, \boldsymbol{\sigma}}(\mathbf{z})$$

# Variational autoencoder (VAE)

$$\ell_{\phi, \theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} + \underbrace{KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))}_{\text{ensures } q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z})}$$

Fixed Gaussian prior over the latent variables:

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

The probabilistic encoder also generates a Gaussian distribution:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}_{\boldsymbol{\mu}, \boldsymbol{\sigma}}(\mathbf{z})$$



# Variational autoencoder (VAE)

$$\ell_{\phi, \theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} + \underbrace{KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))}_{\text{ensures } q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z})}$$

Fixed Gaussian prior over the latent variables:

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

The probabilistic encoder also generates a Gaussian distribution:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}_{\boldsymbol{\mu}, \boldsymbol{\sigma}}(\mathbf{z})$$

$\boldsymbol{\mu}, \boldsymbol{\sigma}$  are functions of the input  $\mathbf{x}$  and the network parameters  $\phi$ .

The probabilistic encoder outputs  $\boldsymbol{\mu}, \boldsymbol{\sigma}$ , not  $\mathbf{z}$ .

# Variational autoencoder (VAE)

$$\ell_{\phi, \theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} + \underbrace{KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))}_{\text{ensures } q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z})}$$

Fixed Gaussian prior over the latent variables:

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

The probabilistic encoder also generates a Gaussian distribution:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}_{\boldsymbol{\mu}, \boldsymbol{\sigma}}(\mathbf{z})$$

$\boldsymbol{\mu}, \boldsymbol{\sigma}$  are functions of the input  $\mathbf{x}$  and the network parameters  $\phi$ .

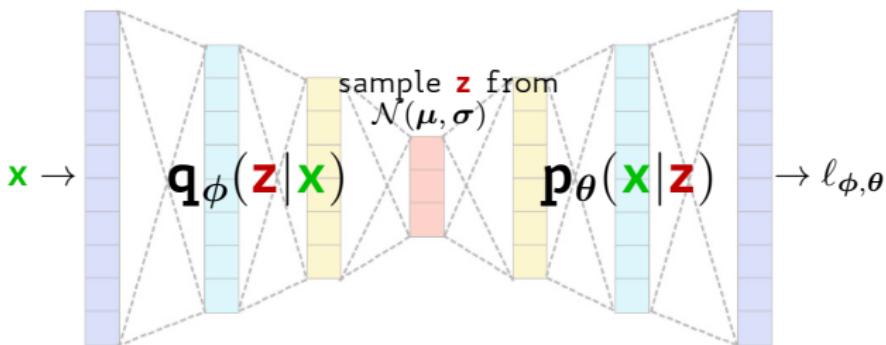
The probabilistic encoder outputs  $\boldsymbol{\mu}, \boldsymbol{\sigma}$ , not  $\mathbf{z}$ .

Using Gaussians, the KL term has a closed form.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

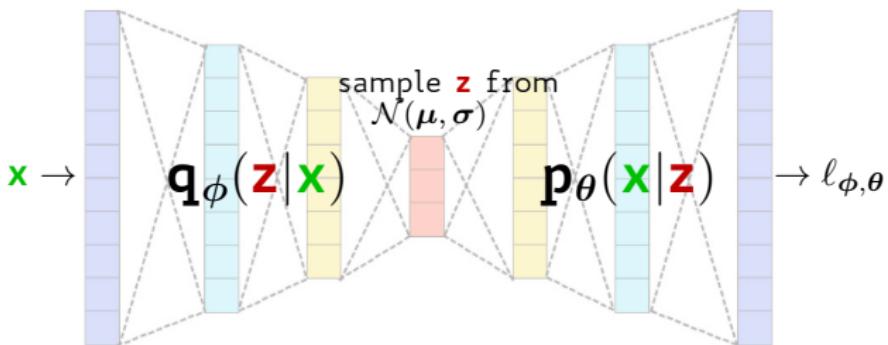
# Variational autoencoder (VAE)

$$\ell_{\phi, \theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} + \underbrace{KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))}_{\text{ensures } q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z})}$$



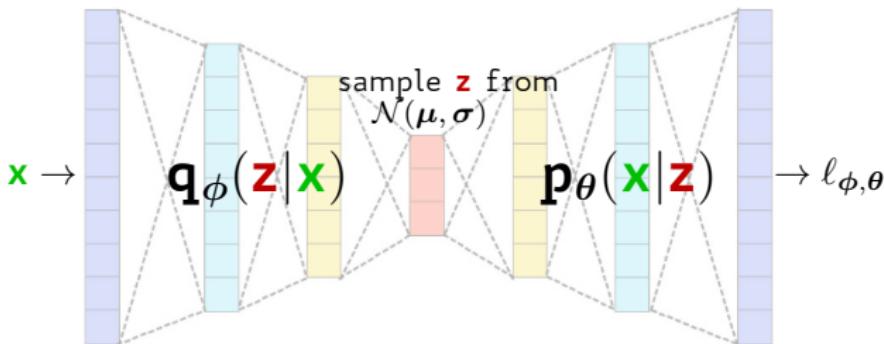
# Variational autoencoder (VAE)

$$\ell_{\phi, \theta} = \underbrace{-\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \log p_{\theta}(\mathbf{x}|\mathbf{z})}_{\text{reconstruction loss} \\ (\text{choose your own})} + \underbrace{KL(q_{\phi}(\mathbf{z}|\mathbf{x}) \| p_{\theta}(\mathbf{z}))}_{\text{regularizer}}$$



# Variational autoencoder (VAE)

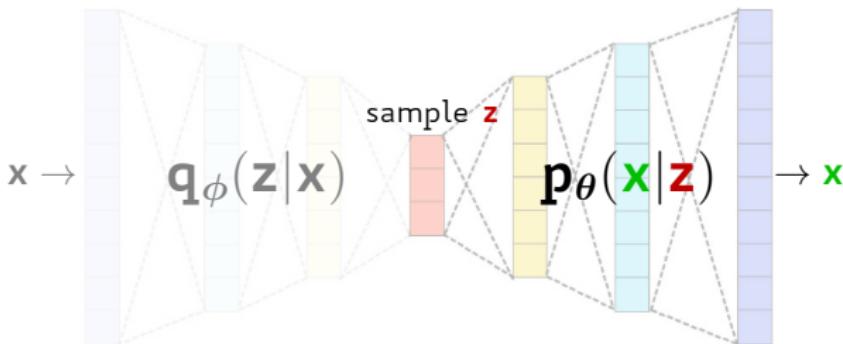
$$\ell_{\phi, \theta} = \underbrace{-\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \log p_{\theta}(\mathbf{x}|\mathbf{z})}_{\text{reconstruction loss} \\ (\text{choose your own})} + \underbrace{KL(q_{\phi}(\mathbf{z}|\mathbf{x}) \| p_{\theta}(\mathbf{z}))}_{\text{regularizer}}$$



For clarity, classical AEs are called **deterministic AEs**.

# Variational autoencoder (VAE)

$$\ell_{\phi, \theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{reconstruction loss} \\ (\text{choose your own})} + \underbrace{KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))}_{\text{regularizer}}$$

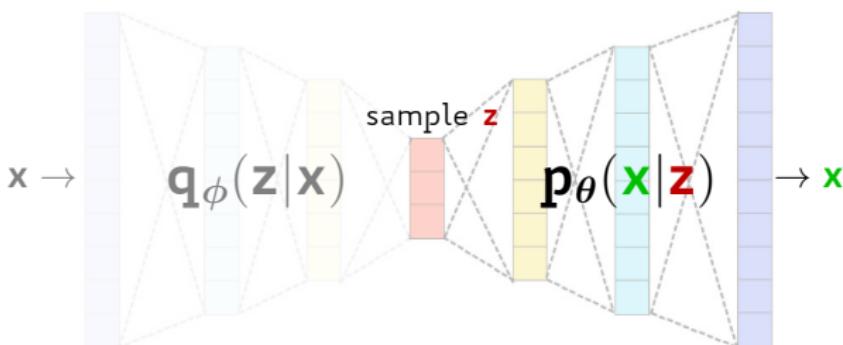


For clarity, classical AEs are called **deterministic AEs**.

**Generation:** Sample  $\mathbf{z}$  according to the learned distribution.

# Variational autoencoder (VAE)

$$\ell_{\phi, \theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{reconstruction loss} \\ (\text{choose your own})} + \underbrace{KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))}_{\text{regularizer}}$$



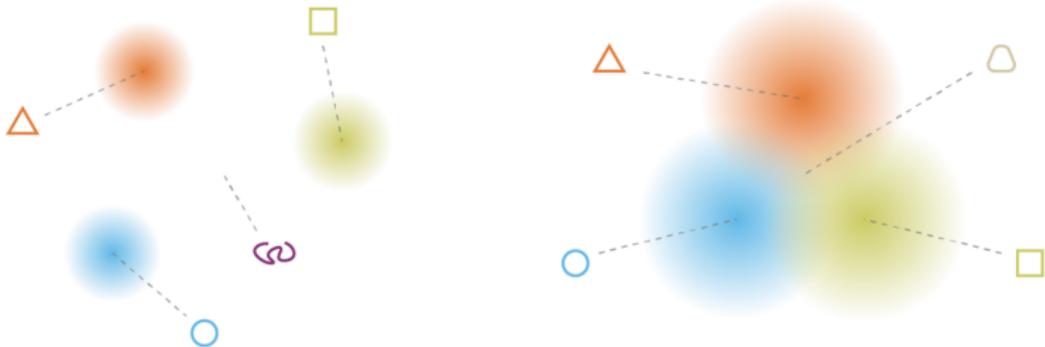
For clarity, classical AEs are called **deterministic AEs**.

**Generation:** Sample  $\mathbf{z}$  according to the learned distribution.

Note: also the **decoder** outputs a probability distribution.

If the reconstruction loss does not need a distribution, take  $\mathbf{x} := \mu$ .

# Regularizing effect

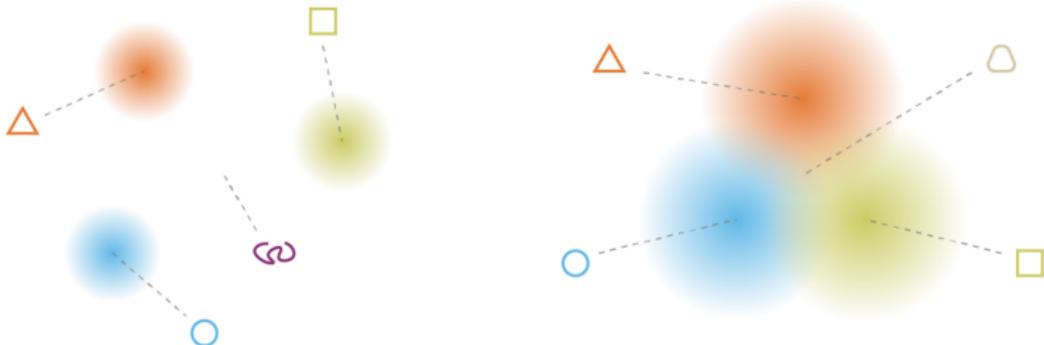


latent space of **deterministic** AE

latent space of **variational** AE

The requirement  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  leads to a compact latent space.  
Small distances  $\|\mathbf{z}_1 - \mathbf{z}_2\| \Rightarrow$  small changes in the decodings.

# Regularizing effect



latent space of **deterministic** AE

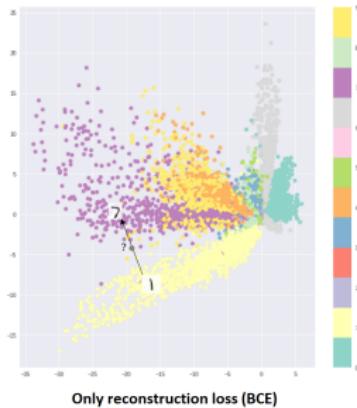
latent space of **variational** AE

The requirement  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  leads to a compact latent space.

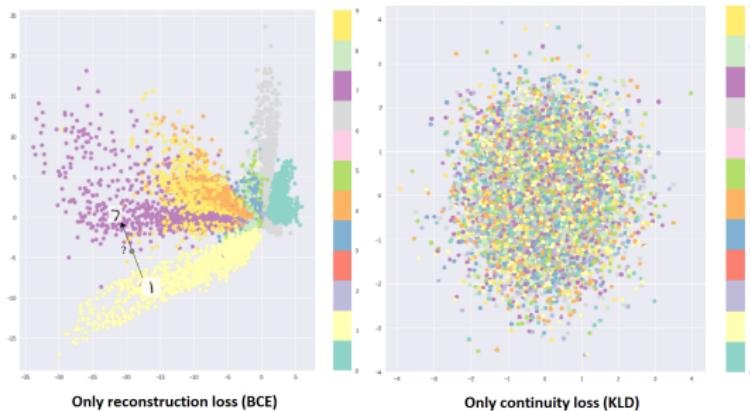
Small distances  $\|\mathbf{z}_1 - \mathbf{z}_2\| \Rightarrow$  small changes in the decodings.

A deterministic AE might generate arbitrary noise.

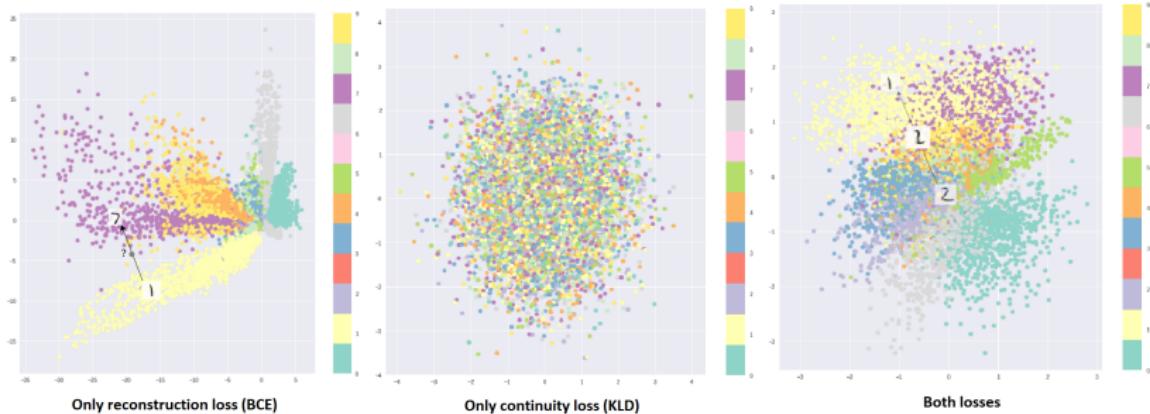
# Regularizing effect



# Regularizing effect



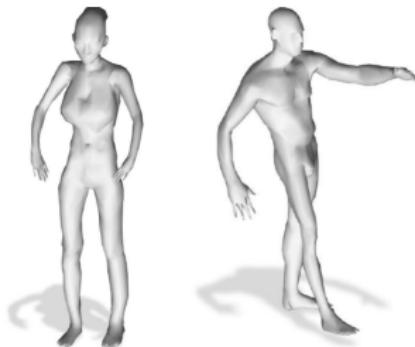
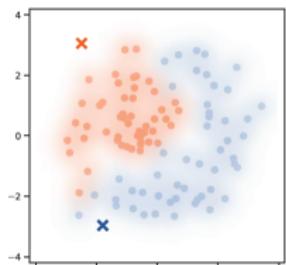
# Regularizing effect



The distributional regularizer brings **smoothness** in the learned space.

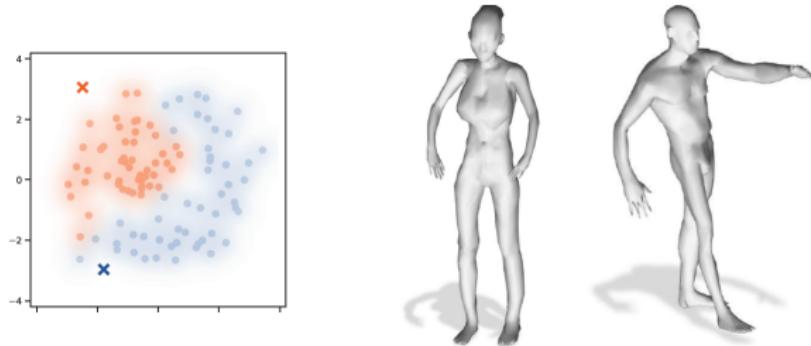
# Unlikely samples

Sampling on the **boundaries** of the learned distribution leads to the generation of **unlikely** samples:



# Unlikely samples

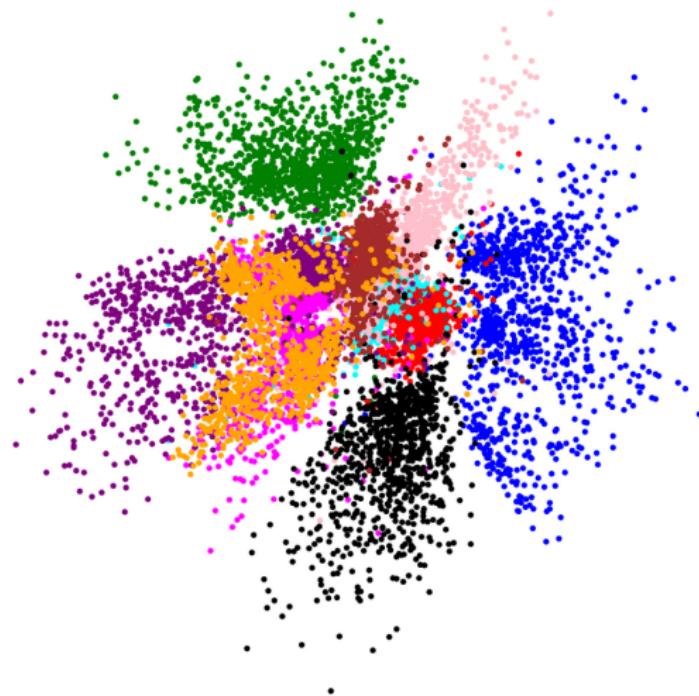
Sampling on the **boundaries** of the learned distribution leads to the generation of **unlikely** samples:



**Posterior collapse:** the latents are ignored if the decoder is powerful enough to model the data perfectly!

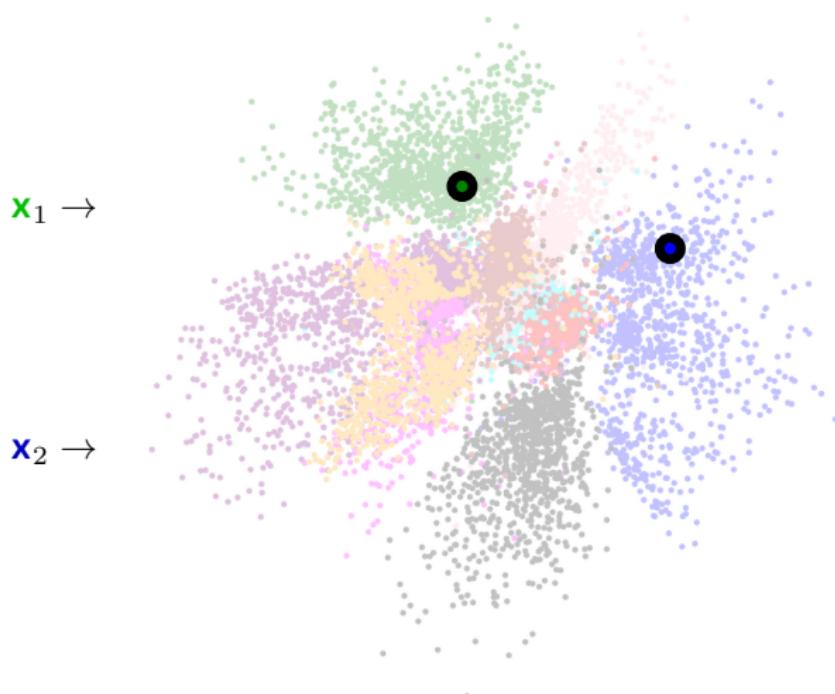
## VAE interpolation

Assume we have a trained VAE, colors correspond to different classes:



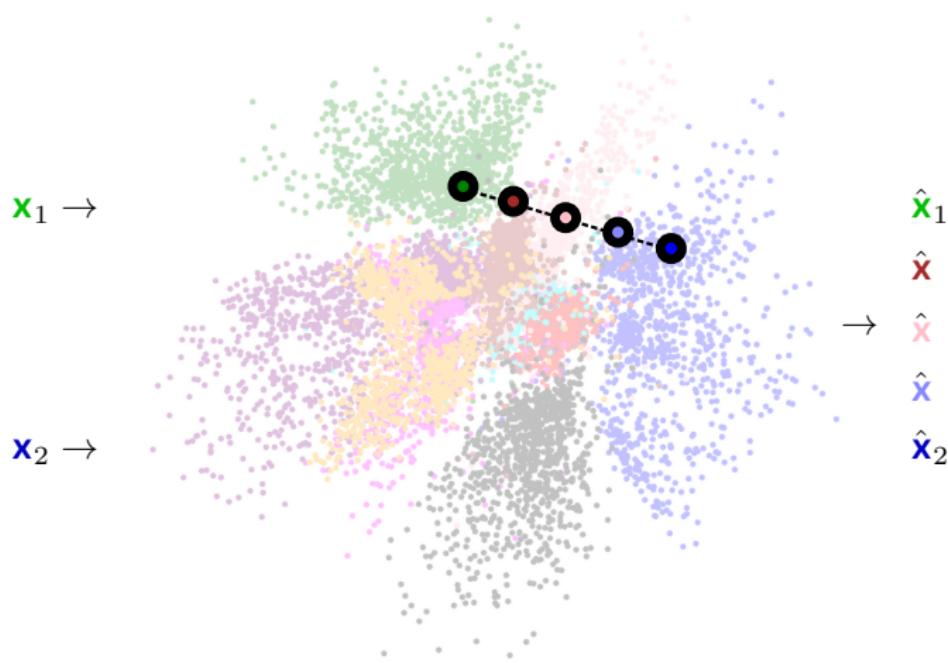
## VAE interpolation

Assume we have a trained VAE, colors correspond to different classes:



# VAE interpolation

Assume we have a trained VAE, colors correspond to different classes:



The interpolated samples are **not** part of the training set.

# VAE interpolation

7

1

# VAE interpolation

7 7 7 7 7 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

# VAE interpolation

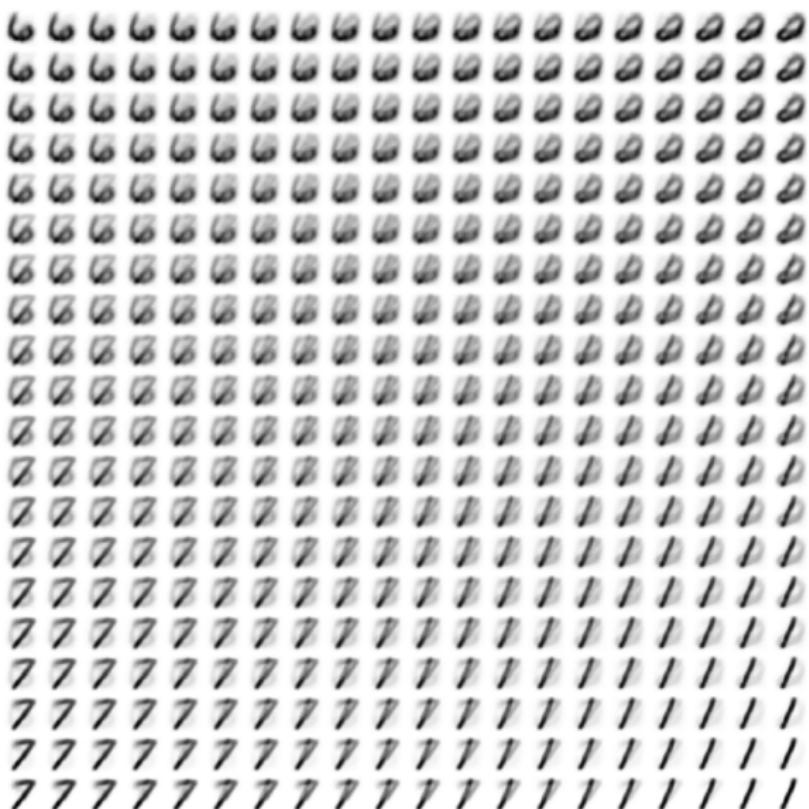
6

7 7 7 7 7 7 1 1 1 1 1 1 1 1 1 1 1 1

## VAE interpolation

## VAE interpolation

# Euclidean interpolation



# Example: Deformable 3D shapes



Cosmo, Norelli, Halimi, Kimmel, Rodolà, "LIMP: Learning Latent Shape Representations with Metric Preservation Priors", 2020

# Example: Deformable 3D shapes



Cosmo, Norelli, Halimi, Kimmel, Rodolà, "LIMP: Learning Latent Shape Representations with Metric Preservation Priors", 2020

## Example: Deformable 3D shapes



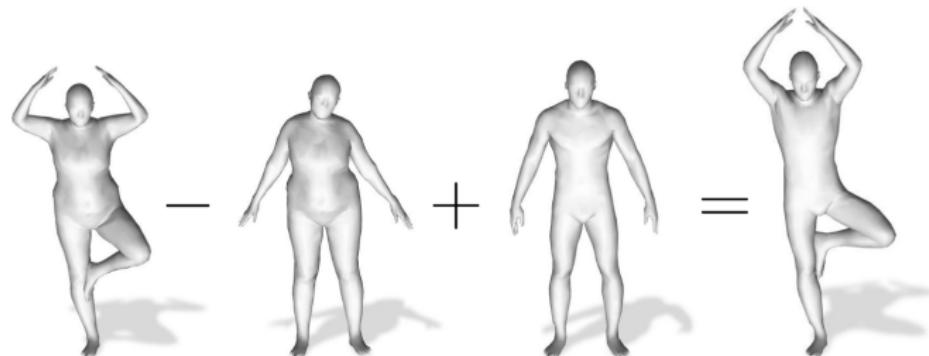
The latent space can be explored along different dimensions ([disentanglement](#)):



Cosmo, Norelli, Halimi, Kimmel, Rodolà, "LIMP: Learning Latent Shape Representations with Metric Preservation Priors", 2020

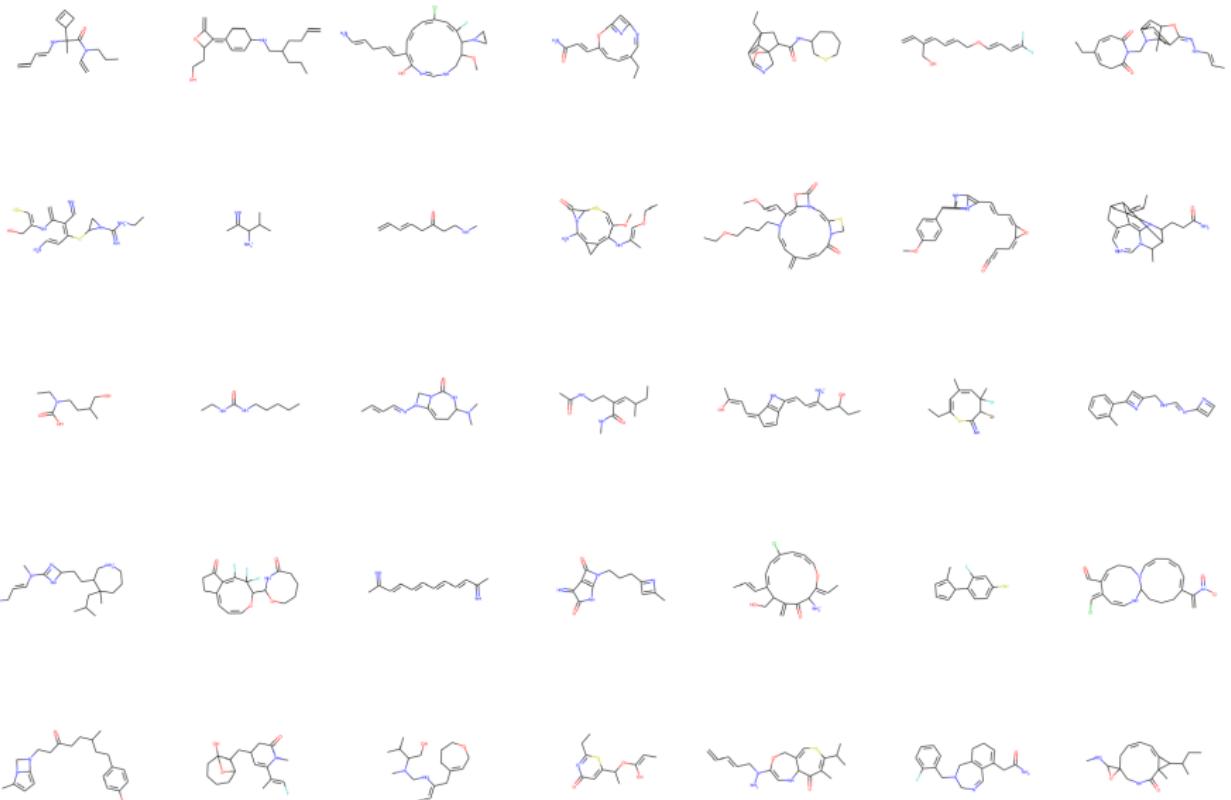
# Analogies

Algebraic manipulation of codes leads to the following:

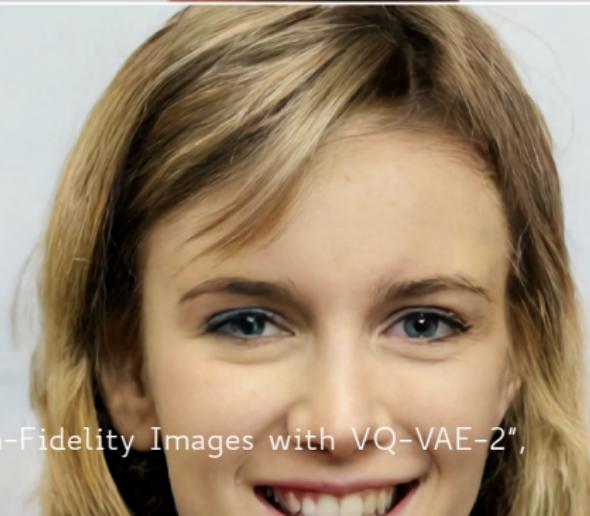


In NLP this is done by manipulating [word embeddings](#):

$$\text{King} - \text{Man} + \text{Woman} = \text{Queen}$$



Bresson, "A Two-Step Graph Convolutional Decoder for Molecule Generation", 2019



Razavi et al, "Generating Diverse High-Fidelity Images with VQ-VAE-2",  
2019