

Deep Learning & Applied AI

Deep generative models

Emanuele Rodolà
rodola@di.uniroma1.it



SAPIENZA
UNIVERSITÀ DI ROMA

Generative models

Overall idea:

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

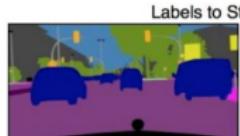
Generative models

Overall idea:

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



Generative models

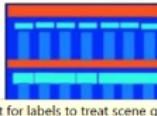


input



output

Labels to Street Scene



Labels to Facade



BW to Color



input



output

Aerial to Map



input



output

Day to Night



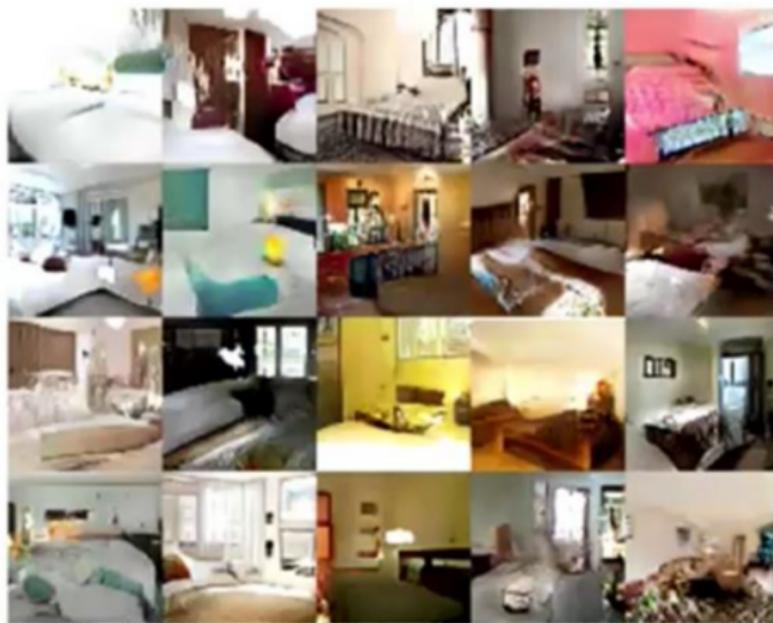
input



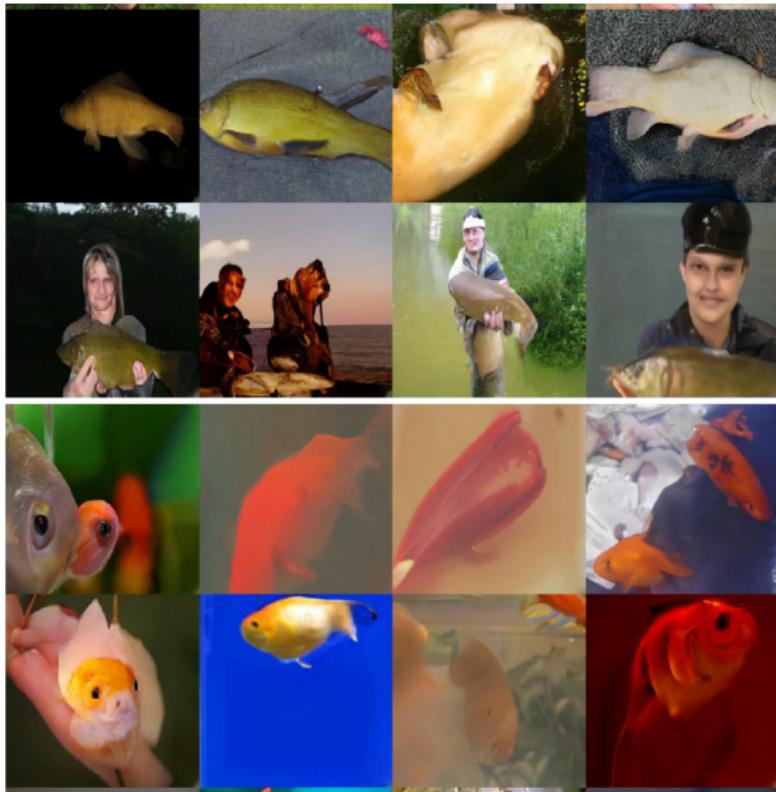
output

Edges to Photo

Generative models



Generative models



Generative models

Caption	Image
this flower has white petals and a yellow stamen	
the center is yellow surrounded by wavy dark purple petals	
this flower has lots of small round pink petals	

Dimensionality reduction

Let us be given n data points stored in matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$:

$$\mathbf{X}^\top = \begin{pmatrix} \text{---} & \mathbf{x}_1^\top & \text{---} \\ & \vdots & \\ \text{---} & \mathbf{x}_n^\top & \text{---} \end{pmatrix}$$

Dimensionality reduction

Let us be given n data points stored in matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$:

$$\mathbf{X}^\top = \begin{pmatrix} - & \mathbf{x}_1^\top & - \\ - & \vdots & - \\ - & \mathbf{x}_n^\top & - \end{pmatrix} \approx \begin{pmatrix} - & \tilde{\mathbf{x}}_1^\top & - \\ - & \vdots & - \\ - & \tilde{\mathbf{x}}_n^\top & - \end{pmatrix} = \tilde{\mathbf{X}}^\top$$

We want to replace them with a **lower-dimensional** approximation $\tilde{\mathbf{X}} \in \mathbb{R}^{k \times n}$, with $k \ll d$

Dimensionality reduction

Let us be given n data points stored in matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$:

$$\mathbf{X}^\top = \begin{pmatrix} - & \mathbf{x}_1^\top & - \\ \vdots & & \vdots \\ - & \mathbf{x}_n^\top & - \end{pmatrix} \approx \begin{pmatrix} - & \tilde{\mathbf{x}}_1^\top & - \\ \vdots & & \vdots \\ - & \tilde{\mathbf{x}}_n^\top & - \end{pmatrix} = \tilde{\mathbf{X}}^\top$$

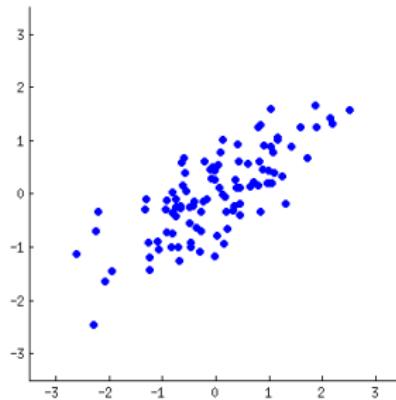
We want to replace them with a **lower-dimensional** approximation $\tilde{\mathbf{X}} \in \mathbb{R}^{k \times n}$, with $k \ll d$

This can be useful for many other tasks:

- Outlier detection
- Visualization
- Denoising
- As a **generative model**

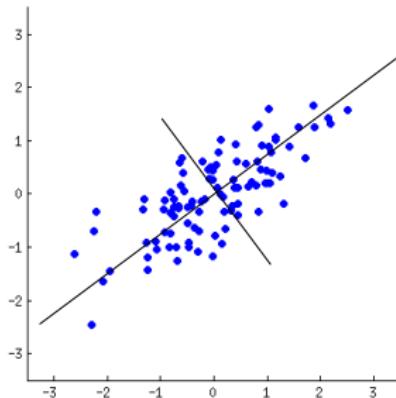
Principal component analysis (PCA)

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



Principal component analysis (PCA)

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

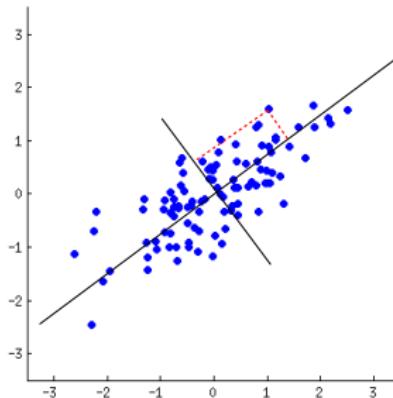


Overall idea:

- Find $k \leq d$ orthogonal directions with the most variance.
These span a k -dimensional subspace of the data.

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.
These span a k -dimensional subspace of the data.
- Project all the data points onto these directions.
This is lossy, but can be done with the smallest possible error.

Principal component analysis (PCA)

In matrix notation:

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

Principal component analysis (PCA)

In matrix notation:

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

Principal component analysis (PCA)

In matrix notation:

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

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:

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

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

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:

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

$$\mathbf{X} \approx \mathbf{WZ}$$

Principal component analysis (PCA)

In matrix notation:

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

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:

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

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

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

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

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** 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 $\mathbf{X}^\top \mathbf{w}$.

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 **projection error** to minimize is:

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

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 **projection error** to minimize is:

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

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 **projection error** to minimize is:

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

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 **projection error** to minimize is:

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

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

We want to solve the problem:

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

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 [projection error](#) to minimize is:

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

where $\mathbf{C} \in \mathbb{R}^{d \times d}$ is the symmetric [covariance matrix](#).

We want to solve the problem:

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

The solution is $\mathbf{w} =$ first [eigenvector](#) of \mathbf{C} ([Courant minmax principle](#)), and the value $\mathbf{w}^\top \mathbf{C} \mathbf{w}$ is the corresponding [eigenvalue](#).

Principal component analysis (PCA)

After solving the problem:

$$\begin{aligned}\mathbf{w}_1 &= \arg \min_{\mathbf{w}} \mathbf{w}^T \mathbf{C} \mathbf{w} \\ \text{s.t. } &\|\mathbf{w}\|_2 = 1\end{aligned}$$

Principal component analysis (PCA)

After solving the problem:

$$\begin{aligned}\mathbf{w}_1 &= \arg \min_{\mathbf{w}} \mathbf{w}^T \mathbf{C} \mathbf{w} \\ \text{s.t. } &\|\mathbf{w}\|_2 = 1\end{aligned}$$

The successive orthogonal direction can be found by solving:

$$\begin{aligned}\mathbf{w}_2 &= \arg \min_{\mathbf{w}} \mathbf{w}^T \mathbf{C} \mathbf{w} \\ \text{s.t. } &\|\mathbf{w}\|_2 = 1 \\ &\mathbf{w}_1^T \mathbf{w} = 0\end{aligned}$$

Principal component analysis (PCA)

After solving the problem:

$$\begin{aligned}\mathbf{w}_1 &= \arg \min_{\mathbf{w}} \mathbf{w}^T \mathbf{C} \mathbf{w} \\ \text{s.t. } &\|\mathbf{w}\|_2 = 1\end{aligned}$$

The successive orthogonal direction can be found by solving:

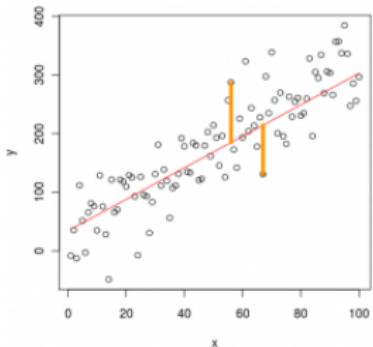
$$\begin{aligned}\mathbf{w}_2 &= \arg \min_{\mathbf{w}} \mathbf{w}^T \mathbf{C} \mathbf{w} \\ \text{s.t. } &\|\mathbf{w}\|_2 = 1 \\ &\mathbf{w}_1^T \mathbf{w} = 0\end{aligned}$$

which is the second eigenvector of \mathbf{C} , and so on for all $\mathbf{w}_{i=2\dots k}$.

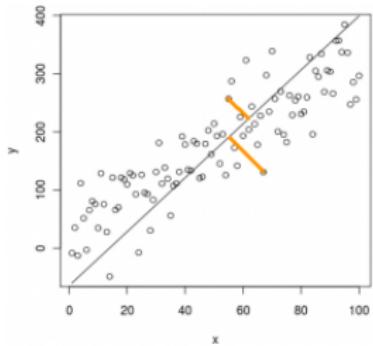
The **principal components** are thus the first $k \ll d$ eigenvectors of \mathbf{C} , sorted by increasing eigenvalue.

PCA is not linear regression

With linear regression we measure the error along each coordinate:



With PCA we measure the error along the principal direction:



PCA as a generative model

Given the \mathbf{W} satisfying, for the observations \mathbf{X} :

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

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

We can generate new data just by sampling $\mathbf{z}_{\text{new}} \in \mathbb{R}^k$ and computing:

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

PCA as a generative model

Given the \mathbf{W} satisfying, for the observations \mathbf{X} :

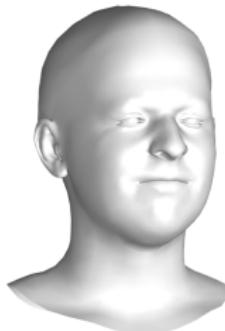
$$\mathbf{X}^\top \mathbf{W} = \mathbf{Z}^\top \quad \text{projection}$$

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

We can generate new data just by sampling $\mathbf{z}_{\text{new}} \in \mathbb{R}^k$ and computing:

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

Example:



Codes

Consider again the relations:

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

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

From a different perspective, PCA gives us a [parametric model](#).

Codes

Consider again the relations:

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

From a different perspective, PCA gives us 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.

The [encoding](#) and [decoding](#) procedures are linear.

Codes

Consider again the relations:

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

From a different perspective, PCA gives us a [parametric model](#).

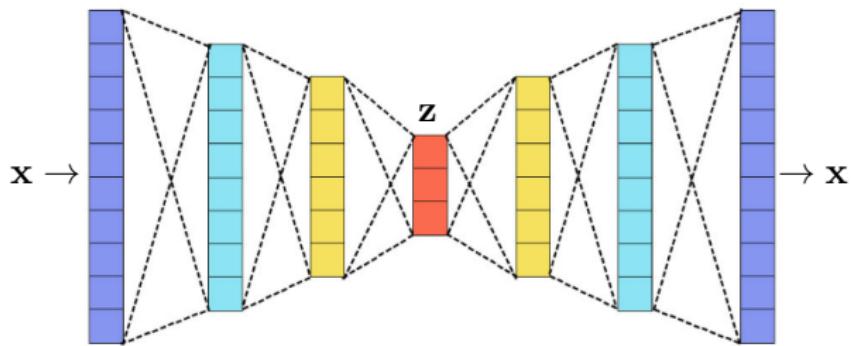
Each data point \mathbf{x} is transformed into a low-dimensional code $\mathbf{z} \in \mathbb{R}^k$, where the dimension $k < d$ is fixed.

The [encoding](#) and [decoding](#) procedures are linear.

How to generalize this idea?

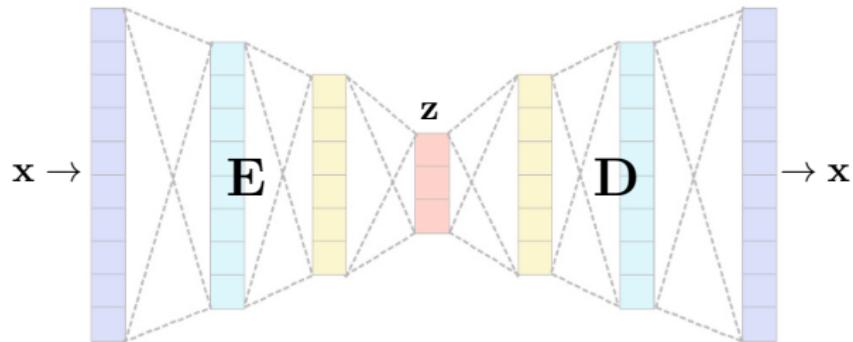
Autoencoders (AE)

We can construct powerful parametric models using deep nets.



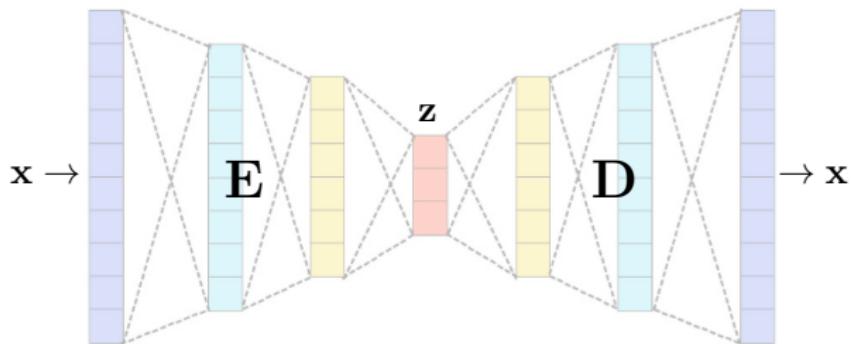
Autoencoders (AE)

We can construct powerful parametric models using deep nets.



Autoencoders (AE)

We can construct powerful parametric models using deep nets.

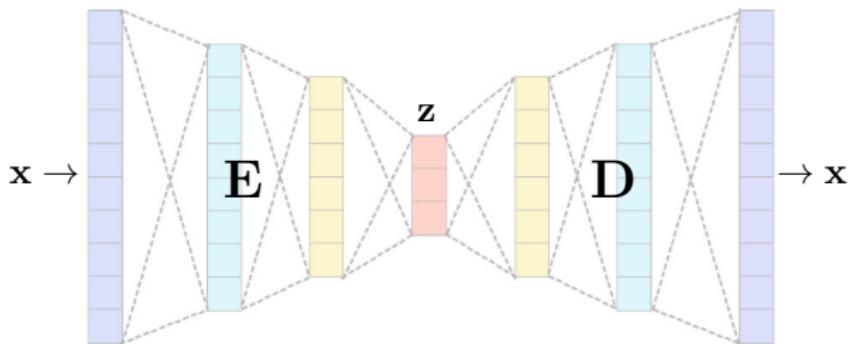


For a given dataset $\{\mathbf{x}_i\}$, we require the encoder E and decoder (or generator) D to minimize the **reconstruction loss**:

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

Autoencoders (AE)

We can construct powerful parametric models using deep nets.



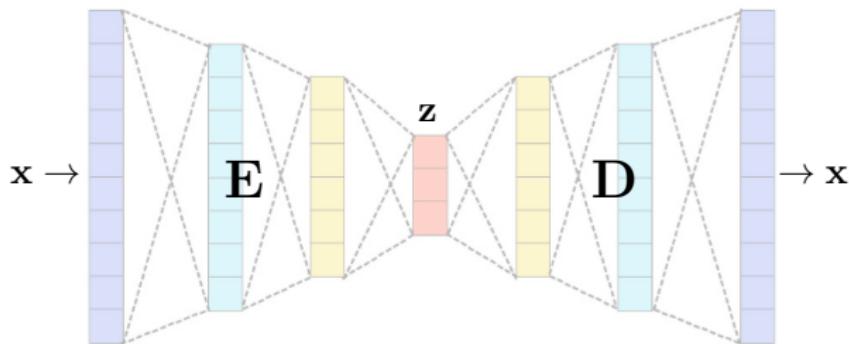
For a given dataset $\{\mathbf{x}_i\}$, we require the encoder E and decoder (or generator) D to minimize the **reconstruction loss**:

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

The choice of the metric depends on the data and on the task.

Autoencoders (AE)

We can construct powerful parametric models using deep nets.



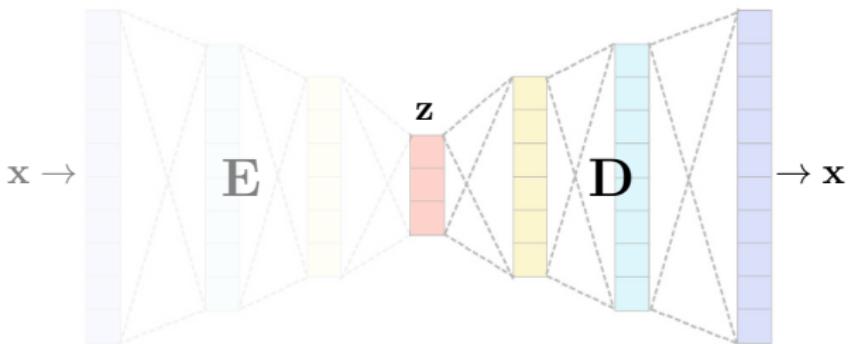
For a given dataset $\{\mathbf{x}_i\}$, we require the encoder E and decoder (or generator) D to minimize the **reconstruction loss**:

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

The choice of the metric depends on the data and on the task.

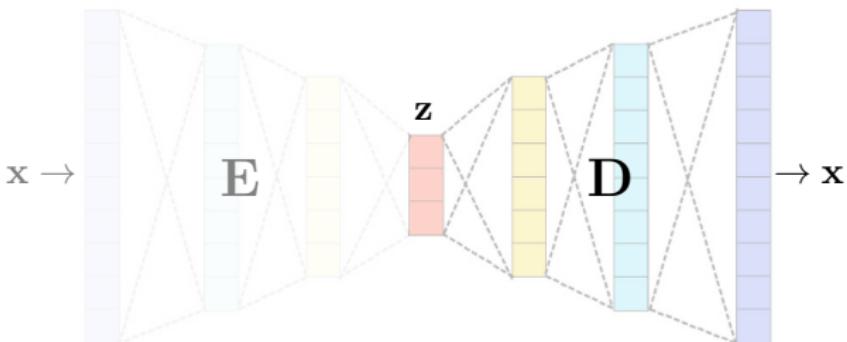
If the layers are **linear**, the codes \mathbf{z}_i span exactly the same space as PCA.

Manifold hypothesis



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

Manifold hypothesis

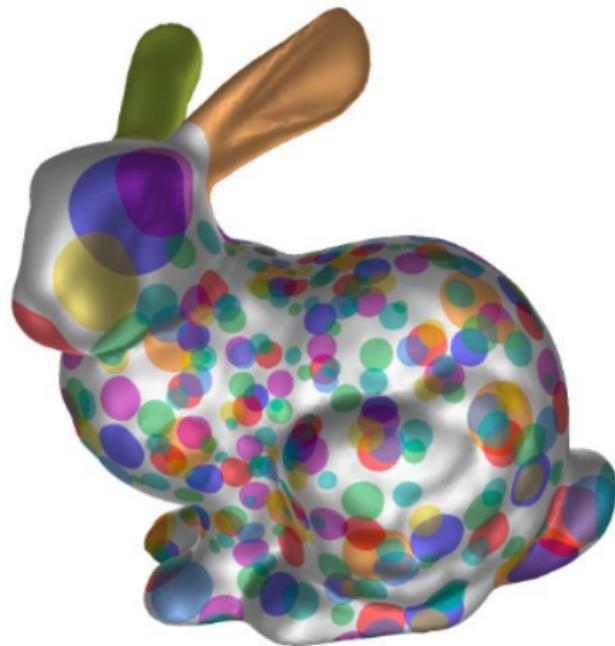


The decoder performs a mapping from a low-dimensional **latent space** to a high-dimensional **embedding** of observed data.

The latent space is Euclidean.

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

Differential geometry

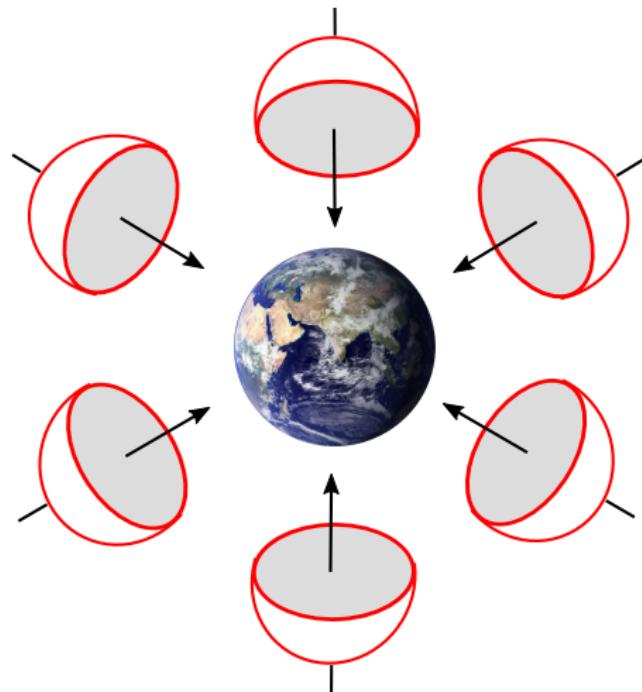


The study of local properties of curves and **surfaces**.

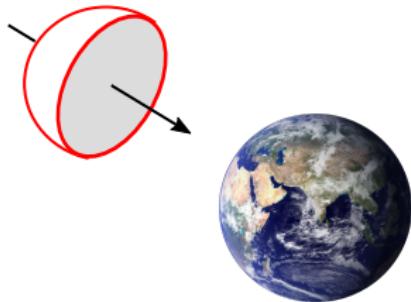
Each neighborhood has a well-behaved mapping to some subset $U \subset \mathbb{R}^2$.

Manifolds

Manifolds are unions of **charts**:

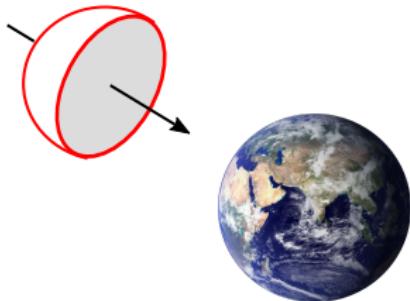


2D manifolds (surfaces)



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

2D manifolds (surfaces)

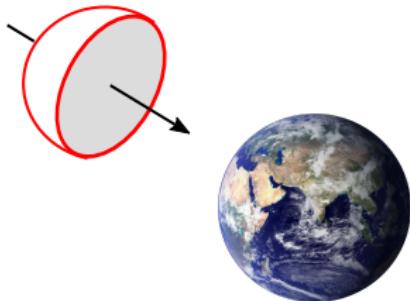


chart

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

We require ϕ to be **smooth** and **invertible** (diffeomorphism).

2D manifolds (surfaces)



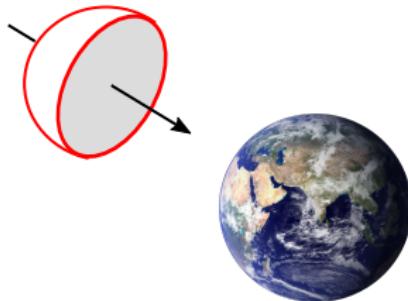
chart

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

We require ϕ to be **smooth** and **invertible** (diffeomorphism).

- The domain of ϕ is the **parametric space** and is Euclidean.

2D manifolds (surfaces)



chart

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

We require ϕ to be **smooth** and **invertible** (diffeomorphism).

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

2D manifolds (surfaces)

- Cut pieces of a plane.

2D manifolds (surfaces)

- Cut pieces of a plane.
- Deform these pieces.

2D manifolds (surfaces)

- Cut pieces of a plane.
- Deform these pieces.
- Glue them together.

2D manifolds (surfaces)

- Cut pieces of a plane.
- Deform these pieces.
- Glue them together.



Manifolds

Manifolds can be k -dimensional, meaning that we have charts:

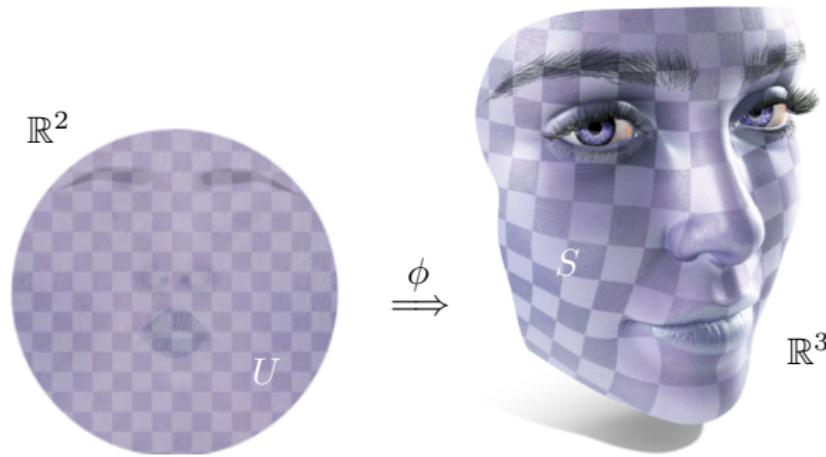
$$\phi : \mathbb{R}^k \rightarrow \mathcal{M} \subset \mathbb{R}^d \quad \text{with } k < d$$

Manifolds

Manifolds can be k -dimensional, meaning that we have charts:

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

The parametrization is **not unique**:

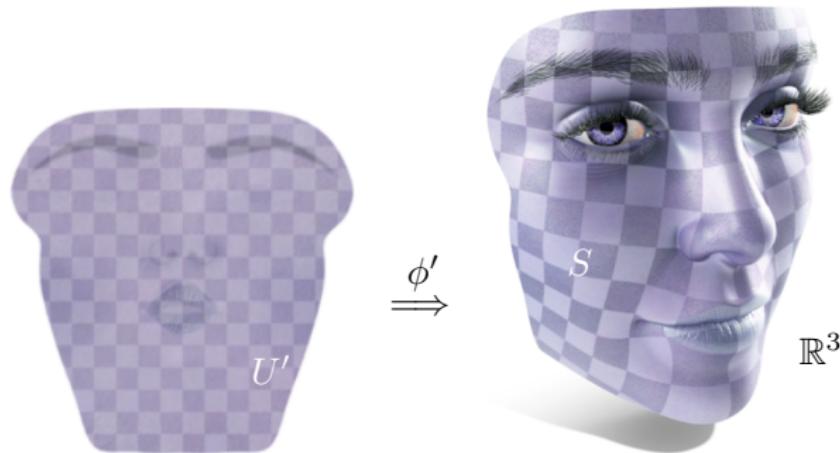


Manifolds

Manifolds can be k -dimensional, meaning that we have charts:

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

The parametrization is **not unique**:

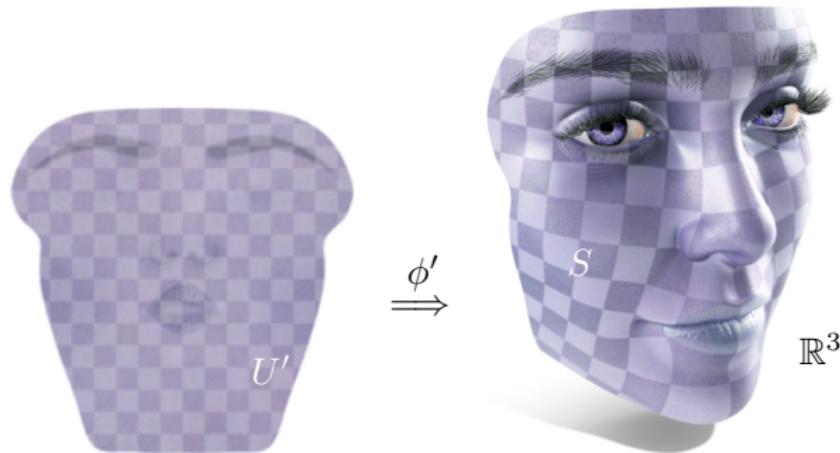


Manifolds

Manifolds can be k -dimensional, meaning that we have charts:

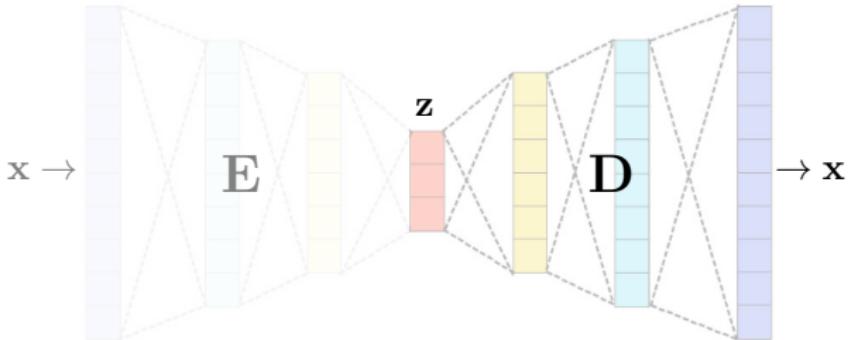
$$\phi : \mathbb{R}^k \rightarrow \mathcal{M} \subset \mathbb{R}^d \quad \text{with } k < d$$

The parametrization is **not unique**:



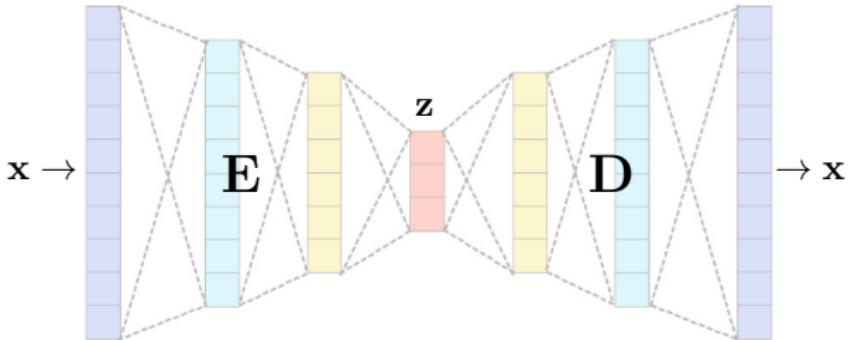
However, all encode exactly **the same** geometric information.

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 z to the data space of the inputs x .

Manifolds and generative models

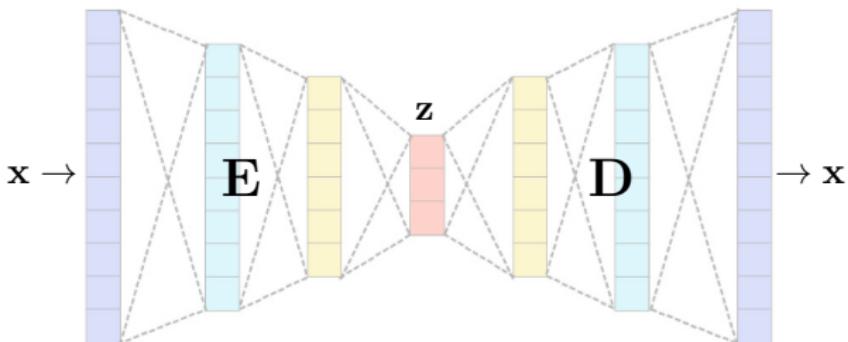


The decoder $D : \mathbb{R}^k \rightarrow \mathbb{R}^d$ is a chart from the latent space spanned by the codes \mathbf{z} to the data space of the inputs \mathbf{x} .

It is differentiable.

It is invertible via the encoder E .

Manifolds and generative models



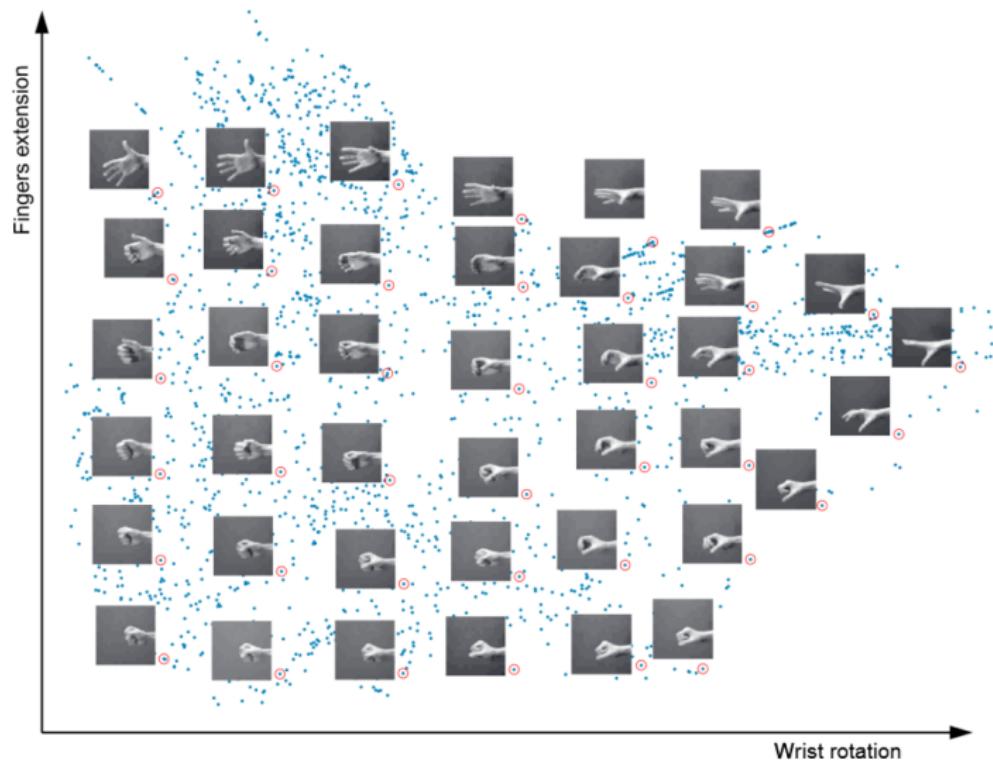
The decoder $D : \mathbb{R}^k \rightarrow \mathbb{R}^d$ is a chart from the latent space spanned by the codes \mathbf{z} to the data space of the inputs \mathbf{x} .

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



The intrinsic/latent dimension k of the manifold is not known a priori.

AE Variants

Many possible variations are possible, acting as extra [regularization](#).

Examples:

- [Denoising AE](#)

Set random values of the input to zero, require exact reconstruction.

The AE is forced to capture correlations between inputs.

Vincent et al, “Extracting and composing robust features with denoising autoencoders”, ICML 2008

AE Variants

Many possible variations are possible, acting as extra **regularization**.

Examples:

- Denoising AE

Set random values of the input to zero, require exact reconstruction.

The AE is forced to capture correlations between inputs.

- Contractive AE

Penalize the gradient of the latent code w.r.t the input.

The latent code becomes robust to small variations in the input.

Vincent et al, "Extracting and composing robust features with denoising autoencoders", ICML 2008; Rifai et al, "Contractive auto-encoders: Explicit invariance during feature extraction", 2011

AE Variants

Many possible variations are possible, acting as extra [regularization](#).

Examples:

- [Denoising AE](#)

Set random values of the input to zero, require exact reconstruction.

The AE is forced to capture correlations between inputs.

- [Contractive AE](#)

Penalize the gradient of the latent code w.r.t the input.

The latent code becomes robust to small variations in the input.

Adding constraints on the latent codes (es. [sparsity](#)), optimizing for the dimension, etc.

Vincent et al, "Extracting and composing robust features with denoising autoencoders", ICML 2008; Rifai et al, "Contractive auto-encoders: Explicit invariance during feature extraction", 2011

Variational autoencoders (VAE)

Autoencoders may possibly learn arbitrary functions, where similar inputs are mapped to arbitrarily distant regions of the latent space.

There is no **coupling**.

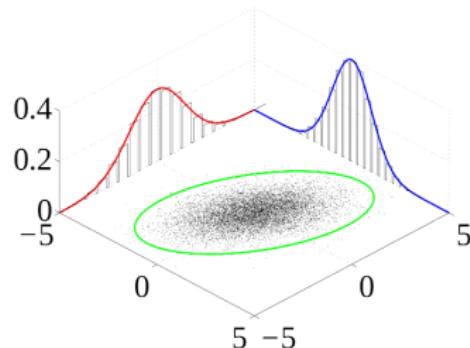
Variational autoencoders (VAE)

Autoencoders may possibly learn arbitrary functions, where similar inputs are mapped to arbitrarily distant regions of the latent space.

There is no **coupling**.

A **variational** autoencoder constructs the parameters of a probability distribution on the latent space.

- The data is seen as a **sampling** of the learned distribution.
- The form of the distribution is fixed and decided **a priori**.



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

Variational inference

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

The **encoder** is modeled as the parametric **posterior** distribution:

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

Variational inference

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

The **encoder** is modeled as the parametric **posterior** distribution:

$$p_{\theta}(\textcolor{blue}{z}|\textcolor{red}{x}) = \frac{p_{\theta}(\textcolor{red}{x}|\textcolor{blue}{z})p_{\theta}(\textcolor{blue}{z})}{p_{\theta}(\textcolor{red}{x})}$$

Variational inference

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

The **encoder** is modeled as the parametric **posterior** distribution:

$$p_{\theta}(\textcolor{blue}{z}|\textcolor{red}{x}) = \frac{p_{\theta}(\textcolor{red}{x}|z)p_{\theta}(z)}{p_{\theta}(\textcolor{red}{x})} = \frac{p_{\theta}(x, z)}{p_{\theta}(\textcolor{red}{x})}$$

Variational inference

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

The **encoder** is modeled as the parametric **posterior** distribution:

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

However, computing:

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

is a high-dimensional **intractable** integral over all possible \mathbf{z} .

Variational inference

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

The encoder is modeled as the parametric posterior distribution:

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

Instead, compute an approximation:

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

where $q_{\phi}(\mathbf{z}|\mathbf{x})$ has a fixed parametric form w.r.t. ϕ .

Variational inference

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

The encoder is modeled as the parametric posterior distribution:

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

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 $q_{\phi}(\mathbf{z}|\mathbf{x})$ has a fixed parametric form w.r.t. ϕ .

Variational inference

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

The encoder is modeled as the parametric posterior distribution:

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

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 $q_{\phi}(\mathbf{z}|\mathbf{x})$ has a fixed parametric form w.r.t. ϕ .

Variational inference

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

The encoder is modeled as the parametric posterior distribution:

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

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 $q_{\phi}(\mathbf{z}|\mathbf{x})$ has a fixed parametric form w.r.t. ϕ .

Variational inference

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

The encoder is modeled as the parametric posterior distribution:

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

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 $q_{\phi}(\mathbf{z}|\mathbf{x})$ has a fixed parametric form w.r.t. ϕ .

Variational inference

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

The encoder is modeled as the parametric posterior distribution:

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

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 $q_{\phi}(\mathbf{z}|\mathbf{x})$ has a fixed parametric form w.r.t. ϕ .

Variational inference

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

The encoder is modeled as the parametric posterior distribution:

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

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 $q_{\phi}(\mathbf{z}|\mathbf{x})$ has a fixed parametric form w.r.t. ϕ .

Variational inference

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

The encoder is modeled as the parametric posterior distribution:

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

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 $q_{\phi}(\mathbf{z}|\mathbf{x})$ has a fixed parametric form w.r.t. ϕ .

Variational inference

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

The encoder is modeled as the parametric posterior distribution:

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

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 $q_{\phi}(\mathbf{z}|\mathbf{x})$ has a fixed parametric form w.r.t. ϕ .

Variational inference

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

The encoder is modeled as the parametric posterior distribution:

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

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 $q_{\phi}(\mathbf{z}|\mathbf{x})$ has a fixed parametric form w.r.t. ϕ .

Variational inference

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

The encoder is modeled as the parametric posterior distribution:

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

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 $q_{\phi}(\mathbf{z}|\mathbf{x})$ has a fixed parametric form w.r.t. ϕ .

Variational inference

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

The encoder is modeled as the parametric posterior distribution:

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

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 $q_{\phi}(\mathbf{z}|\mathbf{x})$ has a fixed parametric form w.r.t. ϕ .

Variational inference

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

The encoder is modeled as the parametric posterior distribution:

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

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

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

where $q_{\phi}(\mathbf{z}|\mathbf{x})$ has a fixed parametric form w.r.t. ϕ .

Variational inference

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

The encoder is modeled as the parametric posterior distribution:

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

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

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

where $q_{\phi}(\mathbf{z}|\mathbf{x})$ has a fixed parametric form w.r.t. ϕ 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)

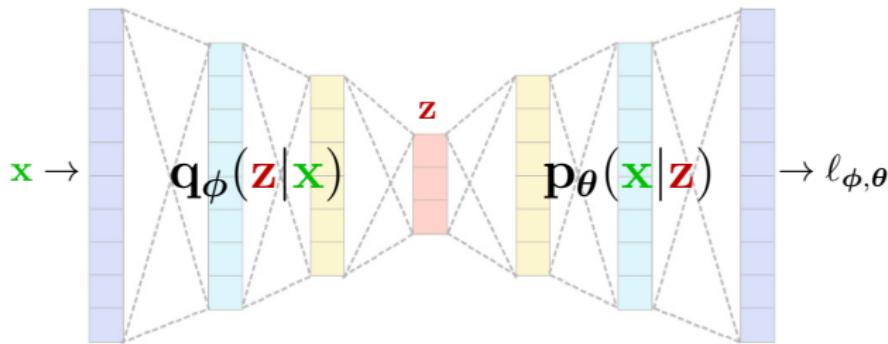
$$\max_{\phi, \theta} \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} - KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

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)

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

The prior over the latent variables is Gaussian and has no free parameters:

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

Variational autoencoder (VAE)

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

The prior over the latent variables is Gaussian and has **no free parameters**:

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

The **probabilistic encoder** also generates a Gaussian distribution:

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

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

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

Variational autoencoder (VAE)

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

The prior over the latent variables is Gaussian and has **no free parameters**:

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

The **probabilistic encoder** also generates a Gaussian distribution:

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

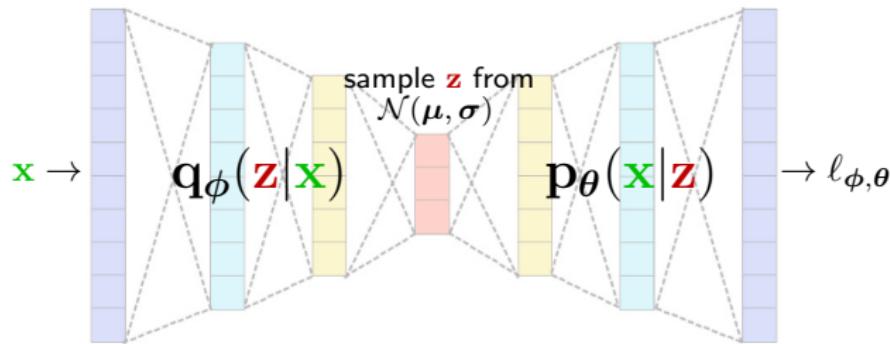
where $\boldsymbol{\mu}, \boldsymbol{\sigma}$ are functions of the input \mathbf{x} and the network parameters ϕ .

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

Using Gaussians, the KL term has a closed form.

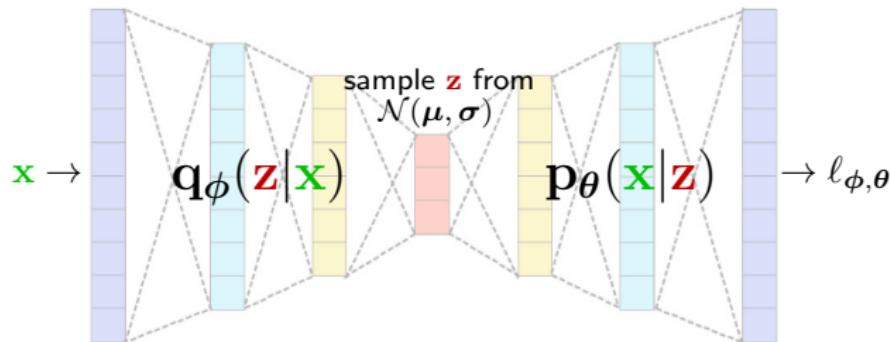
Variational autoencoder (VAE)

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



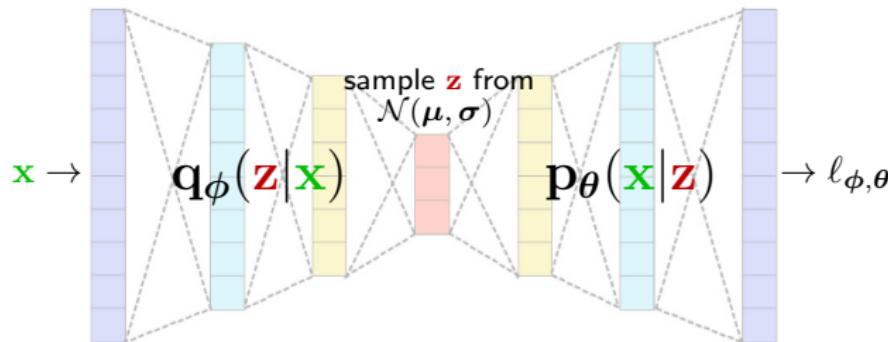
Variational autoencoder (VAE)

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



Variational autoencoder (VAE)

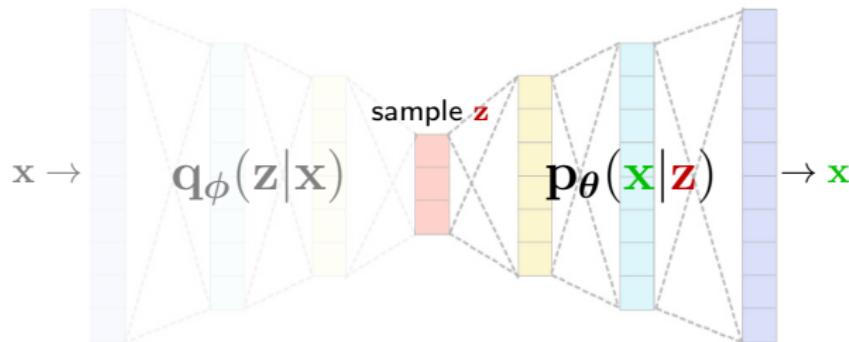
$$\ell_{\phi, \theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{reconstruction loss} \atop (\text{choose your own})} + \underbrace{KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))}_{\text{regularizer}}$$



To distinguish from VAE, classical AE are also called [deterministic AE](#).

Variational autoencoder (VAE)

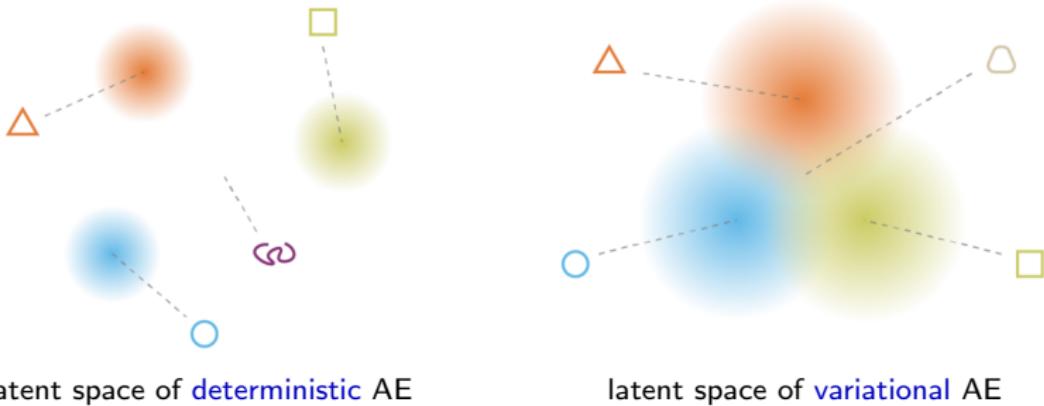
$$\ell_{\phi, \theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{reconstruction loss} \\ (\text{choose your own})} + \underbrace{KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))}_{\text{regularizer}}$$



To distinguish from VAE, classical AE are also called **deterministic AE**.

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

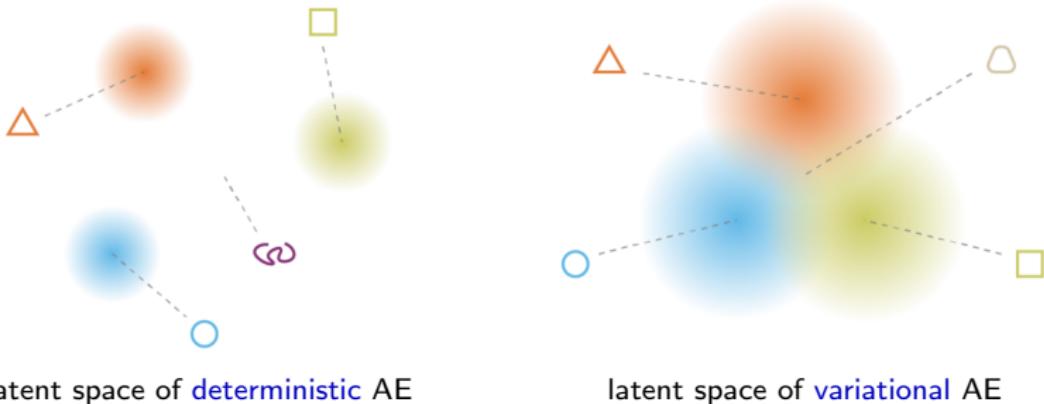
Regularizing effect



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

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

Regularizing effect

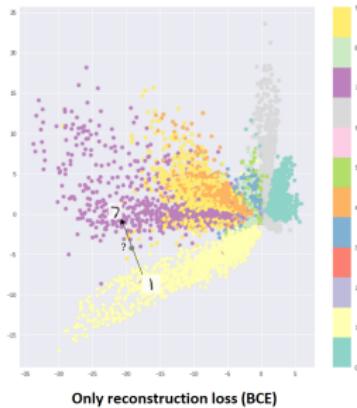


The requirement $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ leads to a compact latent space.

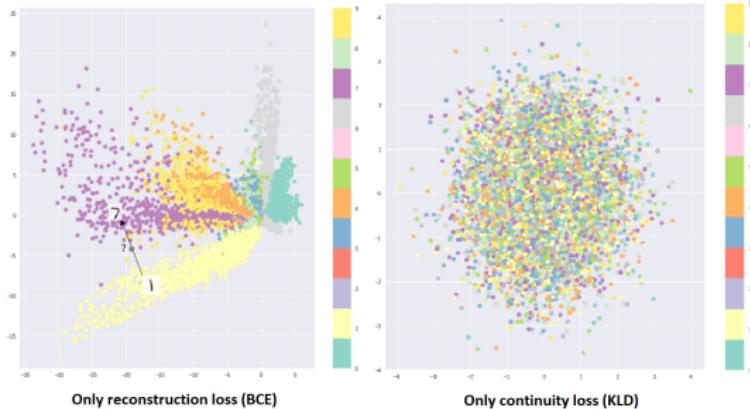
Small distances $\|\mathbf{z}_1 - \mathbf{z}_2\|_2 \Rightarrow$ small changes in the decoded output.

A deterministic AE might generate arbitrary noise.

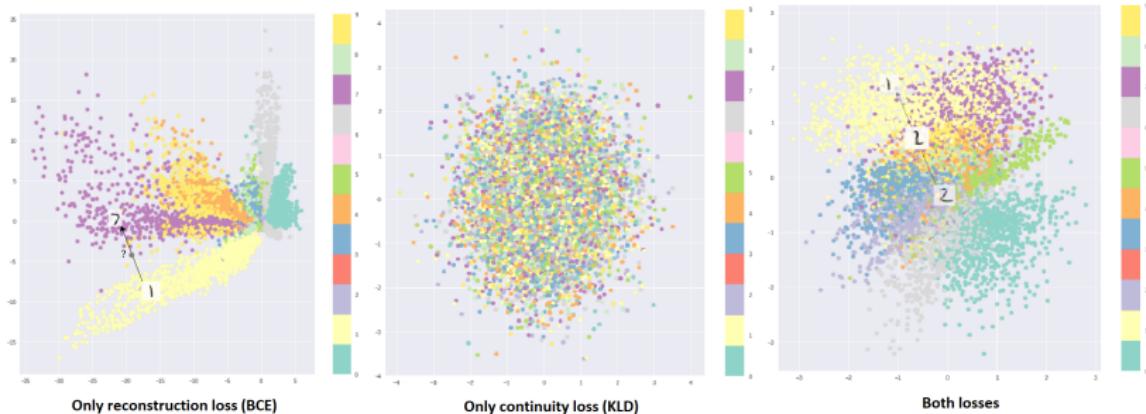
Regularizing effect



Regularizing effect



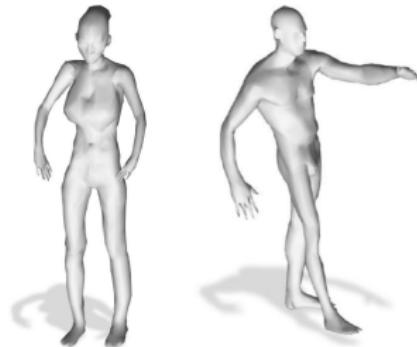
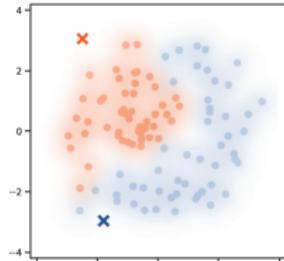
Regularizing effect



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

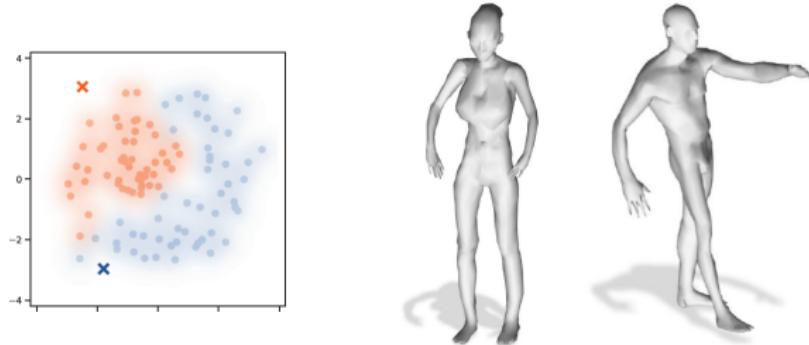
Unlikely samples

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



Unlikely samples

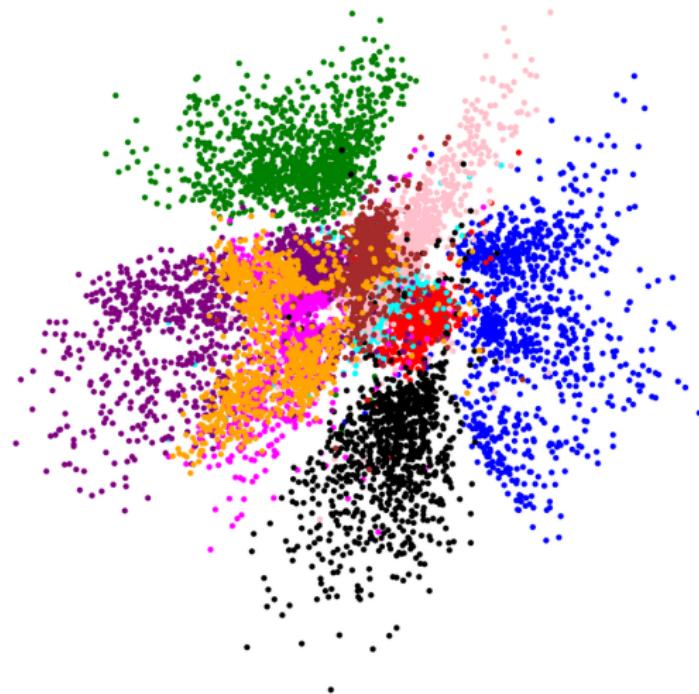
Sampling on the **boundaries** of the learned distribution leads to the generation of **unlikely** samples:



Another relevant phenomenon is **posterior collapse**, where the latents are ignored if the decoder is powerful enough to model the data perfectly.

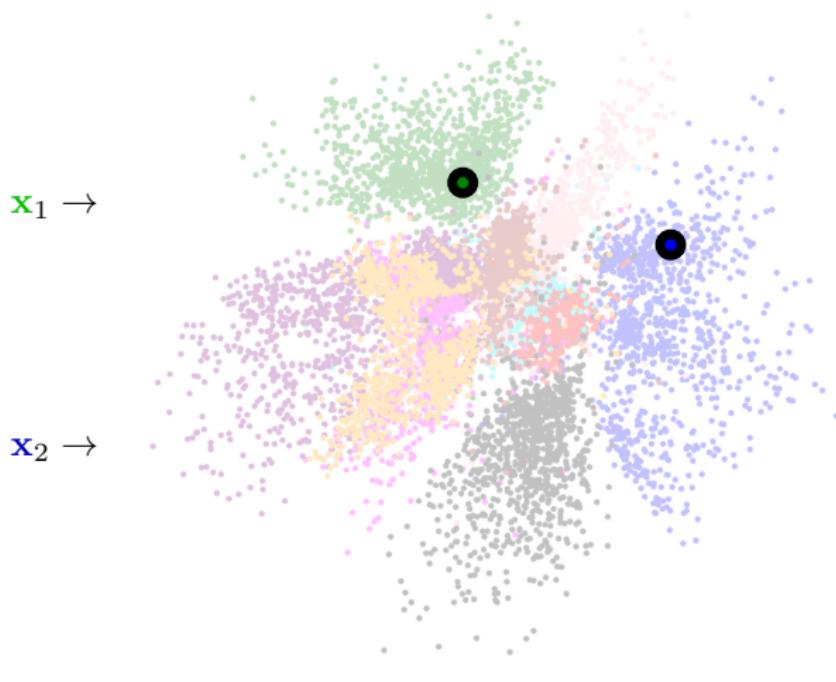
VAE interpolation

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



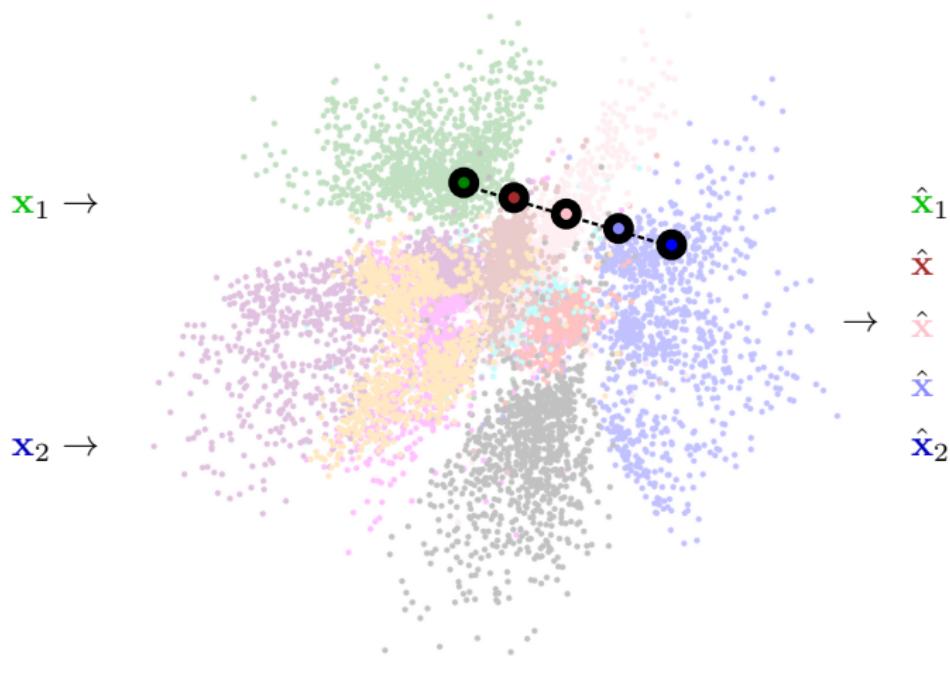
VAE interpolation

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

?

!

VAE interpolation

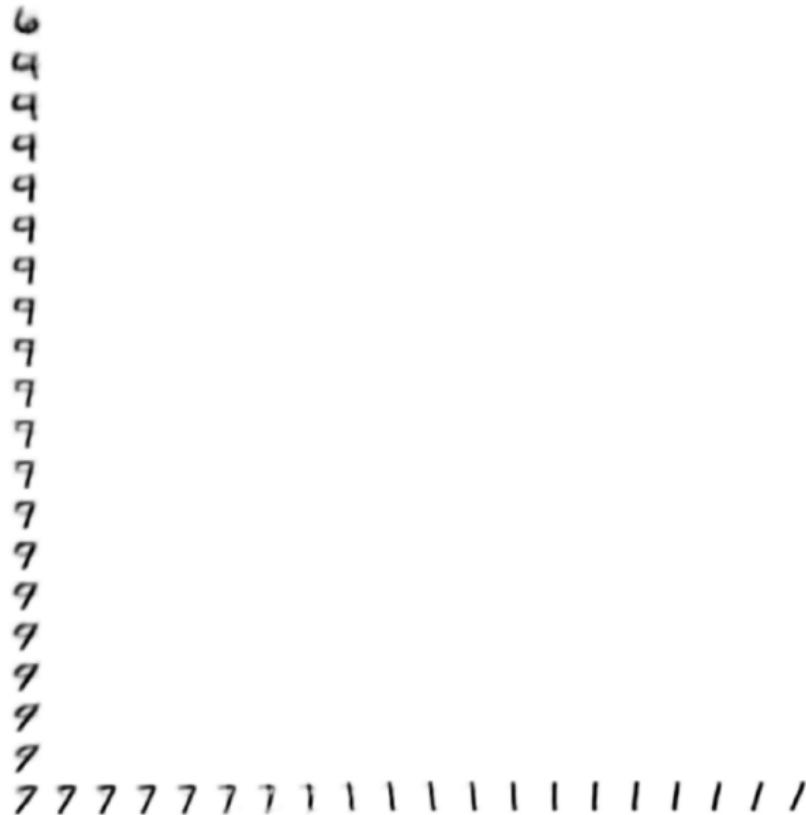
7 7 7 7 7 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

VAE interpolation

6

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

VAE interpolation



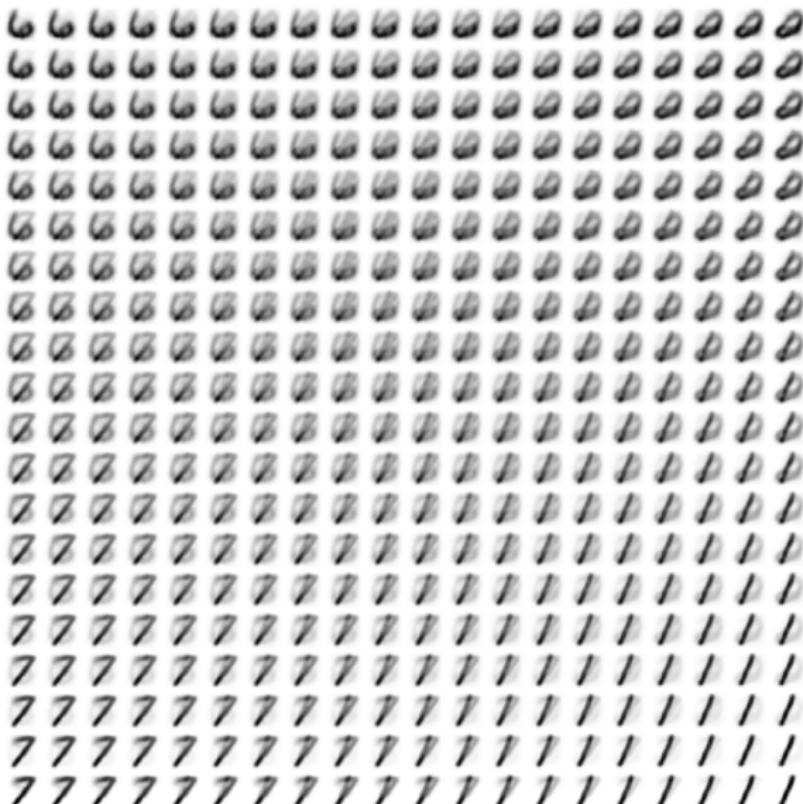
VAE interpolation

A 10x10 grid of handwritten digits, likely generated by a Variational Autoencoder (VAE) during an interpolation process. The digits transition smoothly from a '6' at the top-left to a '7' at the bottom-right, illustrating the latent space manifold.

The digits are arranged in a grid:

- Row 1: 6 6 6 6 6 6 6 6 6 6
- Row 2: 9 4 4 2 2 2 2 2 2 2
- Row 3: 9 2 2 2 2 2 2 2 2 2
- Row 4: 9 9 2 2 2 2 2 2 2 2
- Row 5: 9 9 4 2 2 2 2 2 2 3
- Row 6: 9 9 4 2 2 2 2 2 3 3
- Row 7: 9 9 9 9 2 2 3 3 3 3
- Row 8: 9 9 9 9 9 3 3 3 3 3
- Row 9: 9 9 9 9 9 8 3 3 3 3
- Row 10: 7 9 9 9 9 9 8 8 8 8
- Row 11: 7 9 9 9 9 9 8 8 8 8
- Row 12: 7 9 9 9 9 9 8 8 8 8
- Row 13: 7 9 9 9 9 9 8 8 8 8
- Row 14: 7 9 9 9 9 9 9 8 8 8
- Row 15: 7 9 9 9 9 9 9 9 8 8
- Row 16: 7 9 9 9 9 9 9 9 9 8
- Row 17: 7 9 9 9 9 9 9 9 9 9
- Row 18: 7 9 9 9 9 9 9 9 9 9
- Row 19: 7 9 9 9 9 9 9 9 9 9
- Row 20: 7 7 7 7 7 7 7 7 7 7

Euclidean interpolation

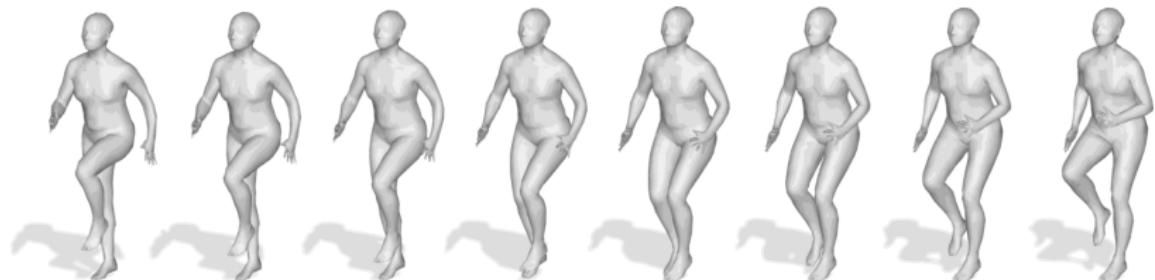


Example: Deformable 3D shapes



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



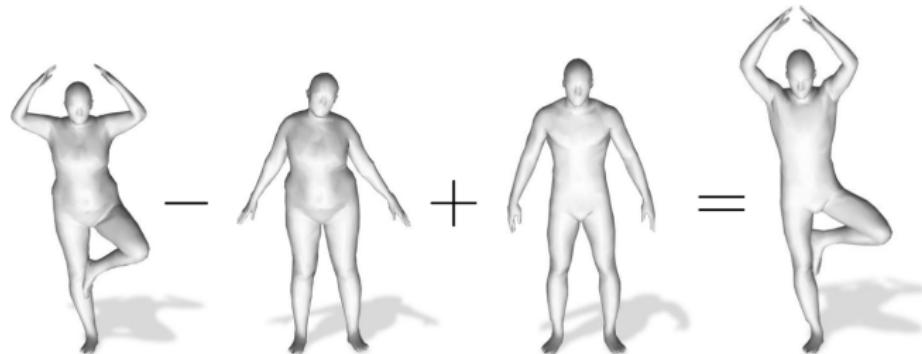
The latent space can be explored along different dimensions
(disentanglement):



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

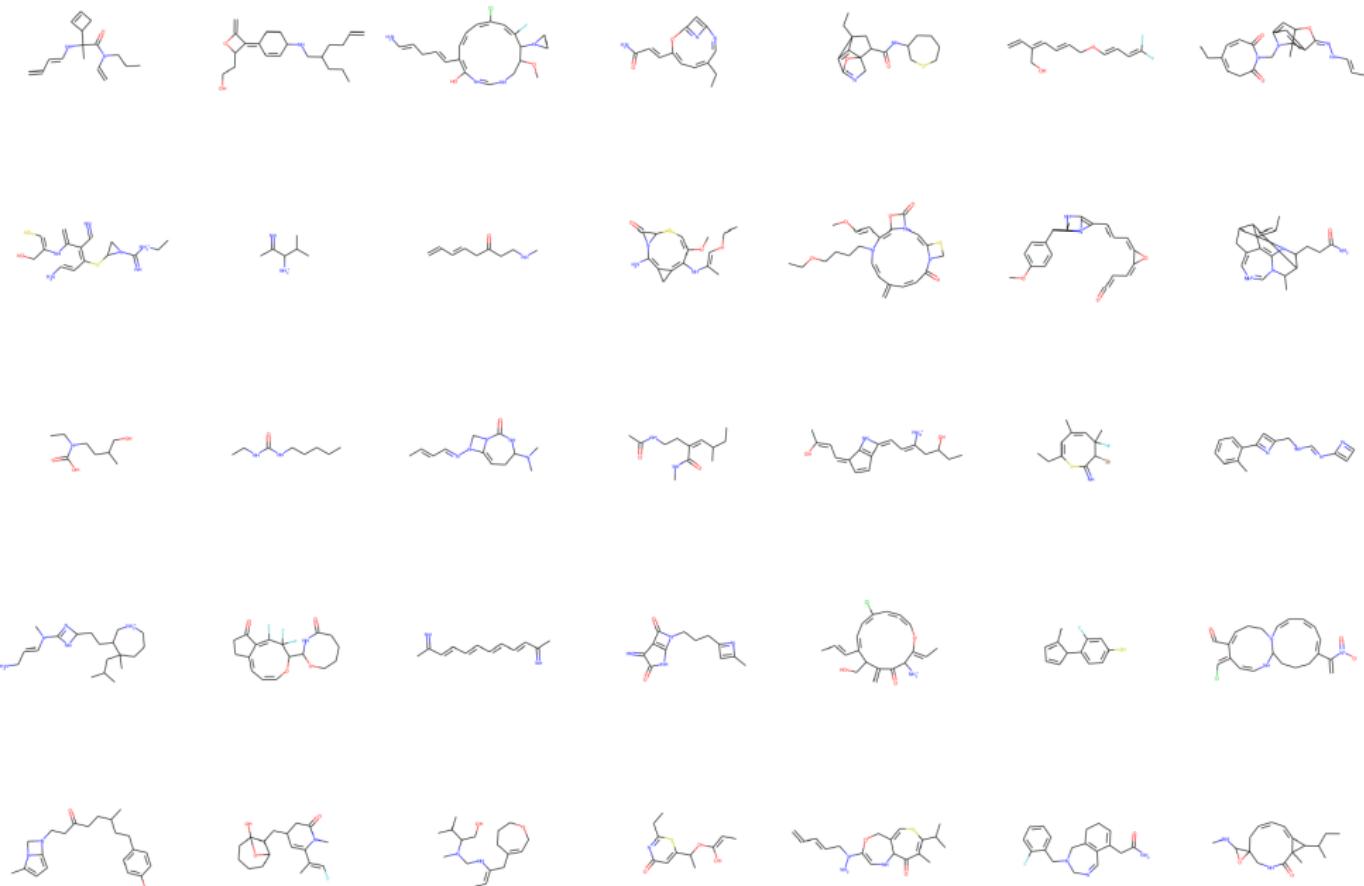
Analogies

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

Suggested reading

- All the references given throughout the slides.