# Deep Learning

Lecture 6: Auto-encoders and generative models

Prof. Gilles Louppe
g.louppe@uliege.be

LIÈGE université

# Today

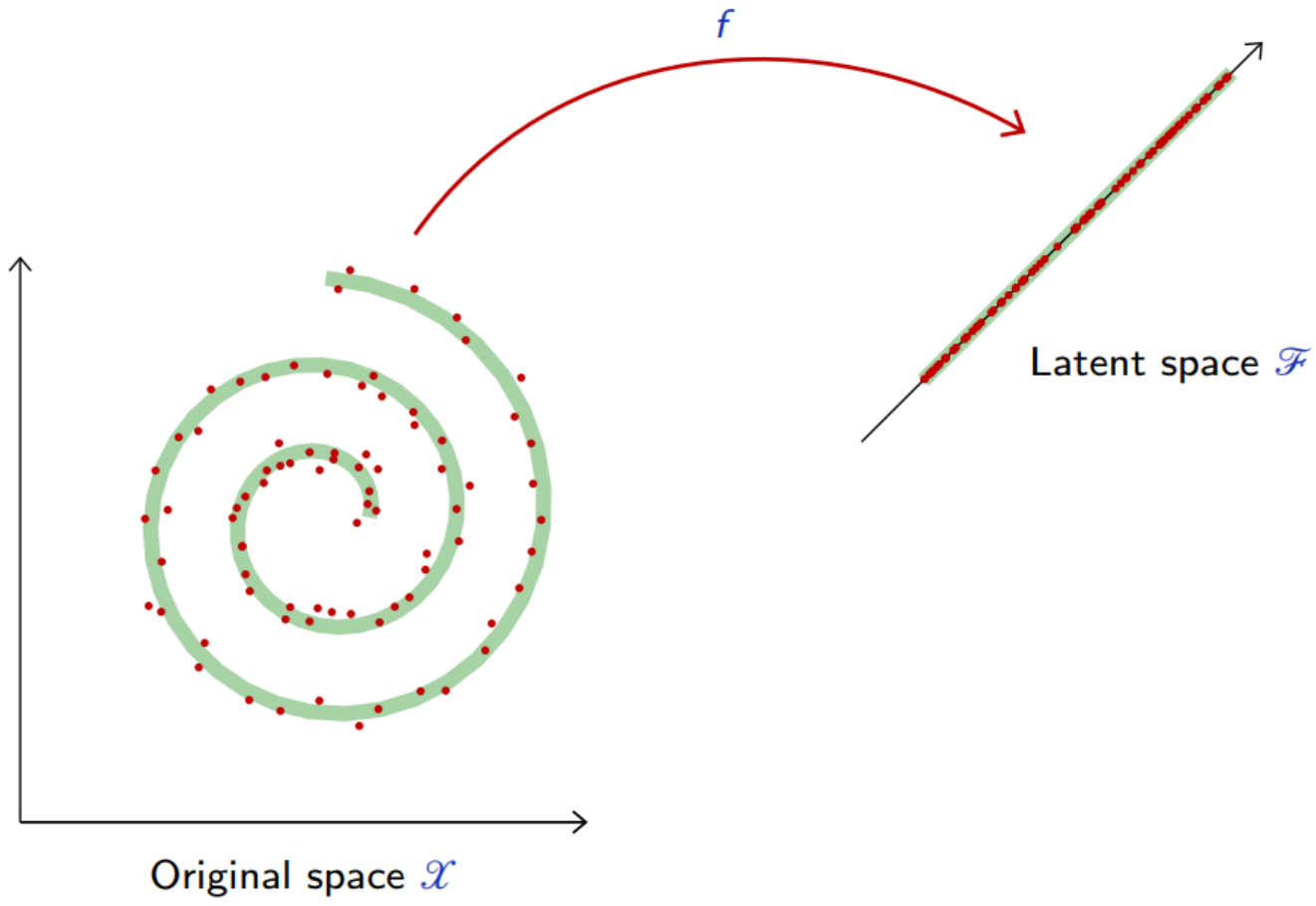
Learn a model of the data.

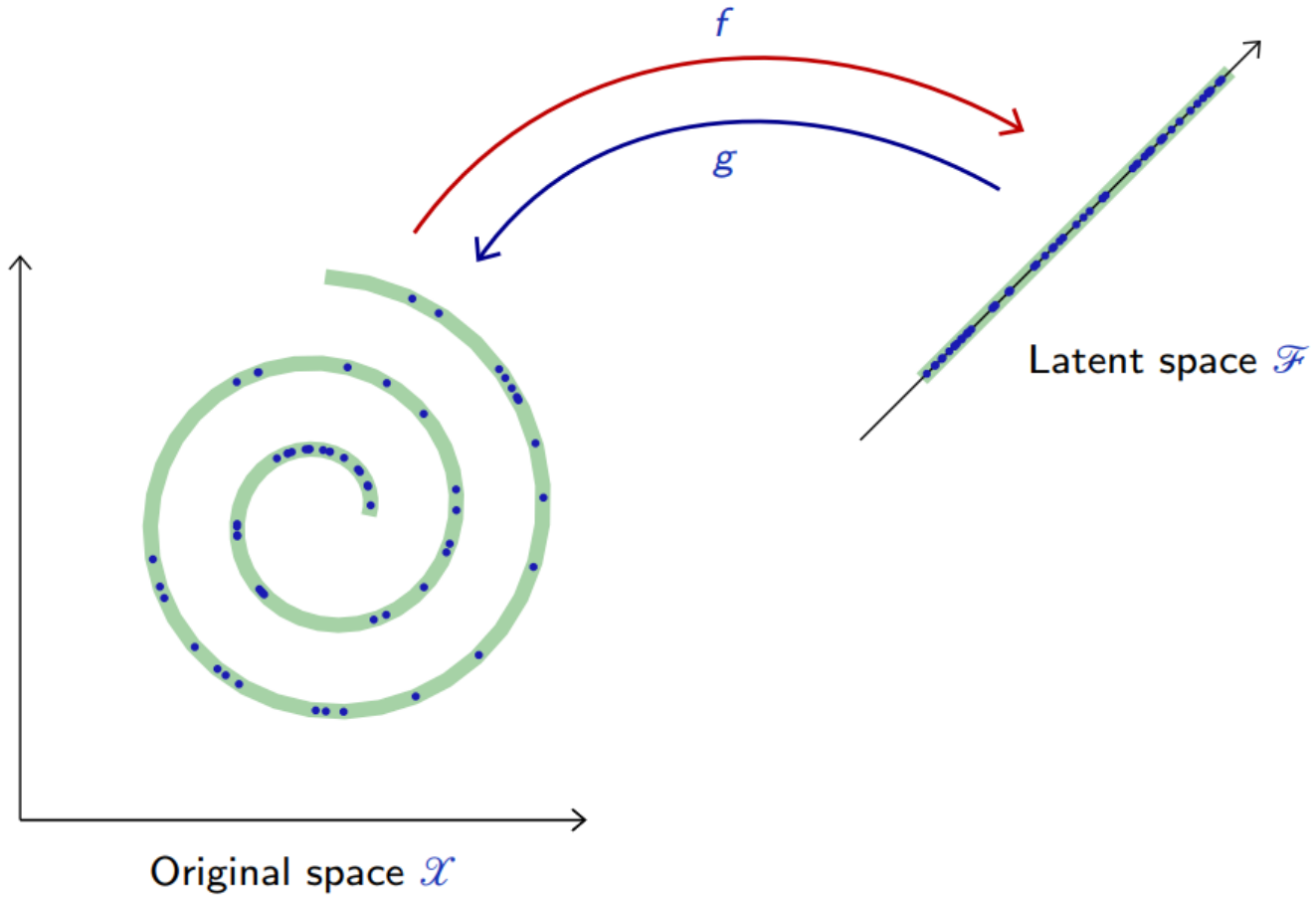
- Auto-encoders
- Generative models
- Variational inference
- Variational auto-encoders

# Auto-encoders

Many applications such as image synthesis, denoising, super-resolution, speech synthesis or compression, require to go beyond classification and regression and model explicitly a high-dimensional signal.

This modeling consists of finding *"meaningful degrees of freedom"*, or *"factors of variations"*, that describe the signal and are of lesser dimension.


$f$
Latent space $\mathcal{F}$
Original space $\mathcal{X}$

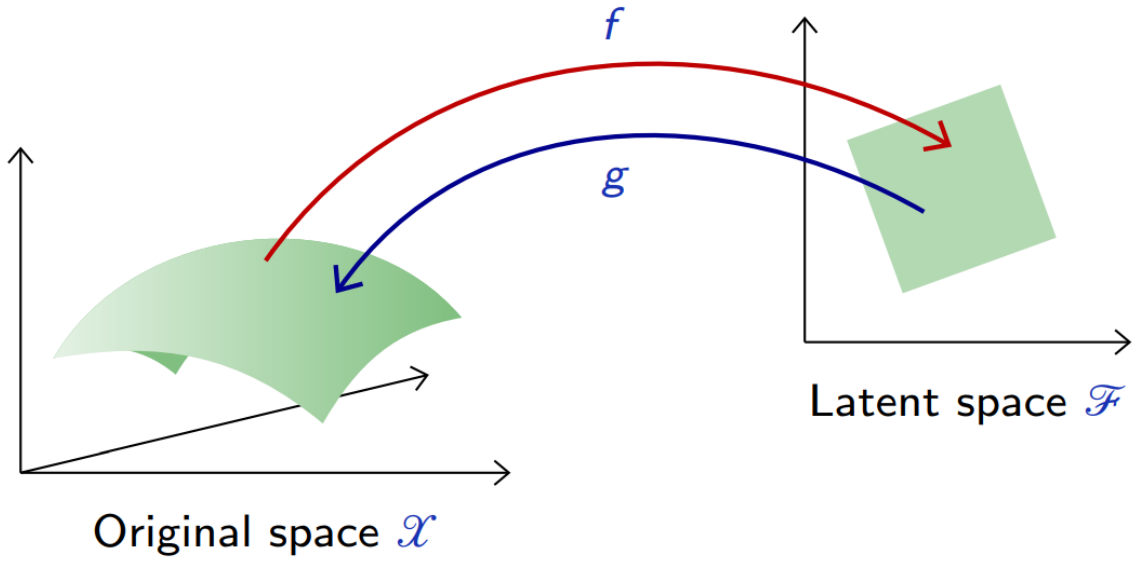

$f$
$g$
Latent space $\mathcal{F}$
Original space $\mathcal{X}$

# Auto-encoders

An auto-encoder is a composite function made of

- an encoder $f$ from the original space $\mathcal{X}$ to a latent space $\mathcal{Z}$,
- a decoder $g$ to map back to $\mathcal{X}$,

such that $g \circ f$ is close to the identity on the data.




A proper auto-encoder should capture a good parameterization of the signal, and in particular the statistical dependencies between the signal components.

Let $p(\mathbf{x})$ be the data distribution over $\mathcal{X}$. A good auto-encoder could be characterized with the reconstruction loss

$$\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[ ||\mathbf{x} - g \circ f(\mathbf{x})||^2 \right] \approx 0.$$

Given two parameterized mappings $f(\cdot; \theta_f)$ and $g(\cdot; \theta_f)$, training consists of minimizing an empirical estimate of that loss,

$$\theta = \arg\min_{\theta_f, \theta_g} \frac{1}{N} \sum_{i=1}^{N} ||\mathbf{x}_i - g(f(\mathbf{x}_i, \theta_f), \theta_g)||^2.$$

For example, when the auto-encoder is linear,

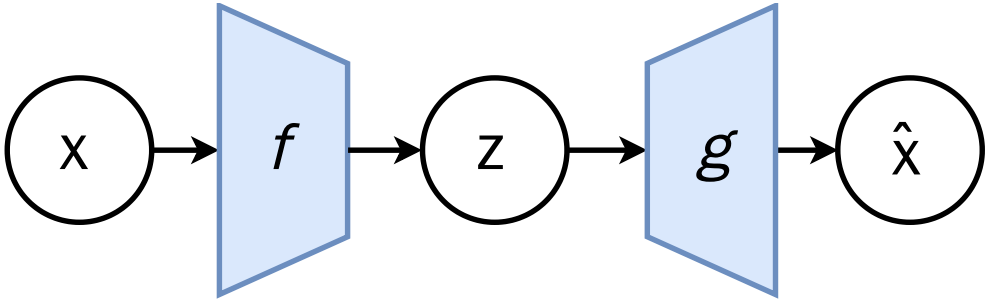
$$f : \mathbf{z} = \mathbf{U}^T \mathbf{x}$$
$$g : \hat{\mathbf{x}} = \mathbf{U}\mathbf{z},$$

with $\mathbf{U} \in \mathbb{R}^{p \times k}$, the reconstruction error reduces to

$$\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[ ||\mathbf{x} - \mathbf{U}\mathbf{U}^T \mathbf{x}||^2 \right].$$

In this case, an optimal solution is given by PCA.

# Deep auto-encoders




Better results can be achieved with more sophisticated classes of mappings than linear projections, in particular by designing $f$ and $g$ as deep neural networks.

For instance,

- by combining a multi-layer perceptron encoder $f : \mathbb{R}^p \to \mathbb{R}^q$ with a multi-layer perceptron decoder $g : \mathbb{R}^q \to \mathbb{R}^p$.
- by combining a convolutional network encoder $f : \mathbb{R}^{w \times h \times c} \to \mathbb{R}^q$ with a decoder $g : \mathbb{R}^q \to \mathbb{R}^{w \times h \times c}$ composed of the reciprocal transposed convolutional layers.

Deep neural decoders require layers that increase the input dimension, i.e., that map $\mathbf{z} \in \mathbb{R}^q$ to $\hat{\mathbf{x}} = g(\mathbf{z}) \in \mathbb{R}^p$, with $p \gg q$.

- This is the opposite of what we did so far with feedforward networks, in which we reduced the dimension of the input to a few values.
- Fully connected layers could be used for that purpose but would face the same limitations as before (spatial specialization, too many parameters).
- Ideally, we would like layers that implement the inverse of convolutional and pooling layers.

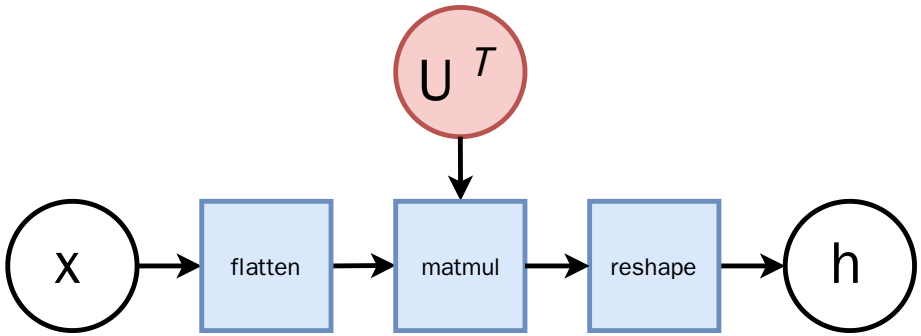
## Transposed convolutions

A transposed convolution is a convolution where the implementation of the forward and backward passes are swapped.

Given a convolutional kernel $\mathbf{u}$,

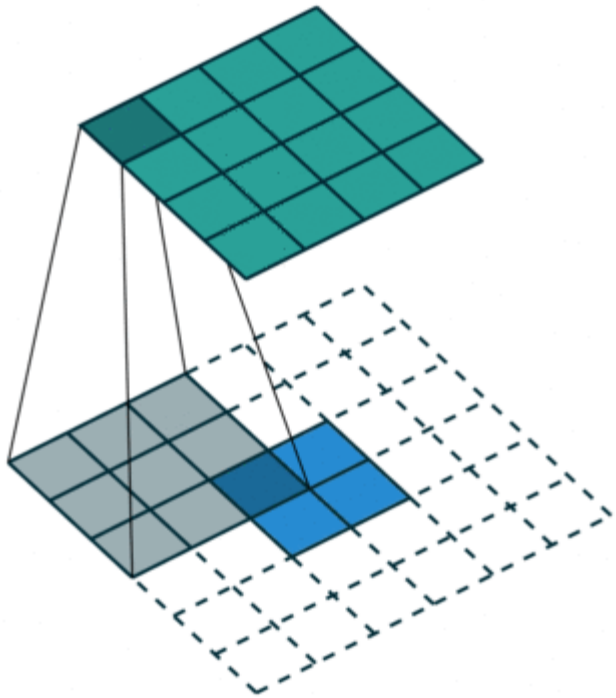
- the forward pass is implemented as $v(\mathbf{h}) = \mathbf{U}^T v(\mathbf{x})$ with appropriate reshaping, thereby effectively up-sampling an input $v(\mathbf{x})$ into a larger one;
- the backward pass is computed by multiplying the loss by $\mathbf{U}$ instead of $\mathbf{U}^T$.

Transposed convolutions are also referred to as fractionally-strided convolutions or deconvolutions (mistakenly).



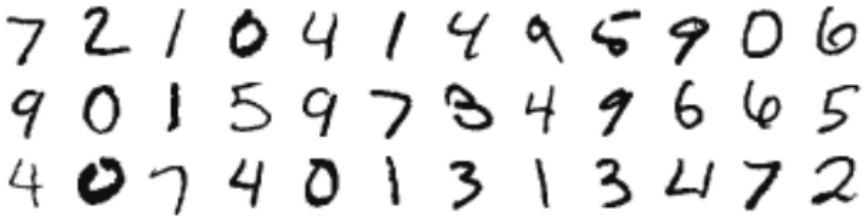
$$\mathbf{U}^T v(\mathbf{x}) = v(\mathbf{h})$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 \\ 1 & 4 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 4 & 1 & 4 & 1 \\ 3 & 4 & 1 & 4 \\ 0 & 3 & 0 & 1 \\ 3 & 0 & 1 & 0 \\ 3 & 3 & 4 & 1 \\ 1 & 3 & 3 & 4 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 3 & 3 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \\ 4 \\ 4 \end{pmatrix} = \begin{pmatrix} 2 \\ 9 \\ 6 \\ 1 \\ 6 \\ 29 \\ 30 \\ 7 \\ 10 \\ 29 \\ 33 \\ 13 \\ 12 \\ 24 \\ 16 \\ 4 \end{pmatrix}$$

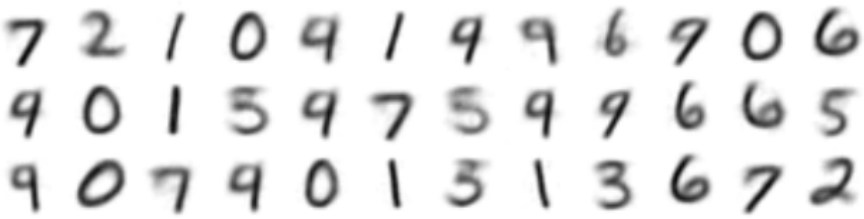


Credits: Dumoulin and Visin, A guide to convolution arithmetic for deep learning, 2016.

$X$ (original samples)

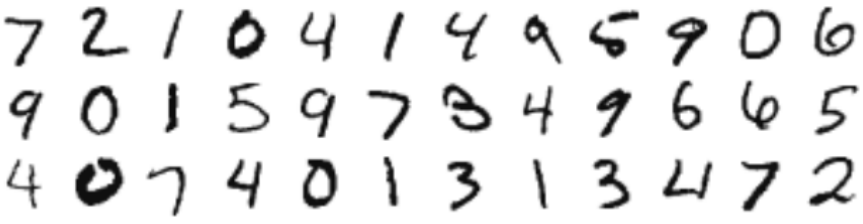


$g \circ f(X)$ (CNN, $d = 2$)

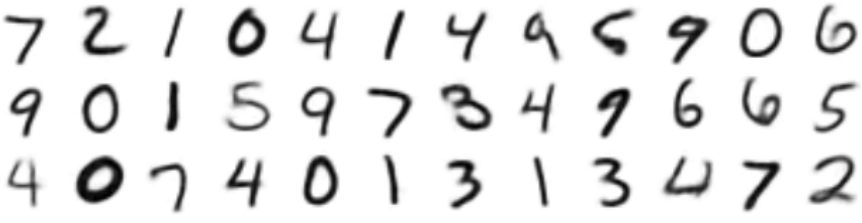


$g \circ f(X)$ (PCA, $d = 2$)

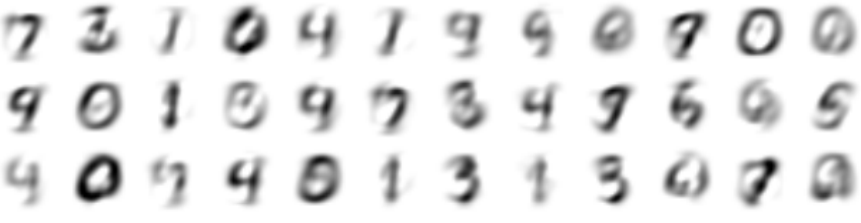
$X$ (original samples)



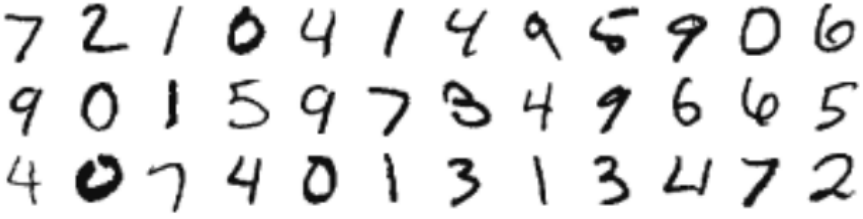
$g \circ f(X)$ (CNN, $d = 8$)



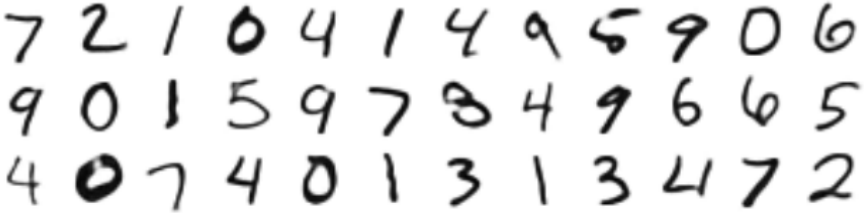
$g \circ f(X)$ (PCA, $d = 8$)



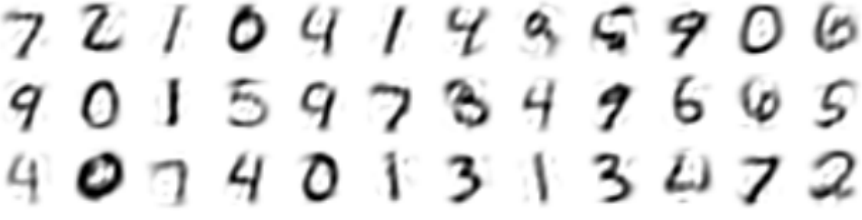
$X$ (original samples)



$g \circ f(X)$ (CNN, $d = 32$)

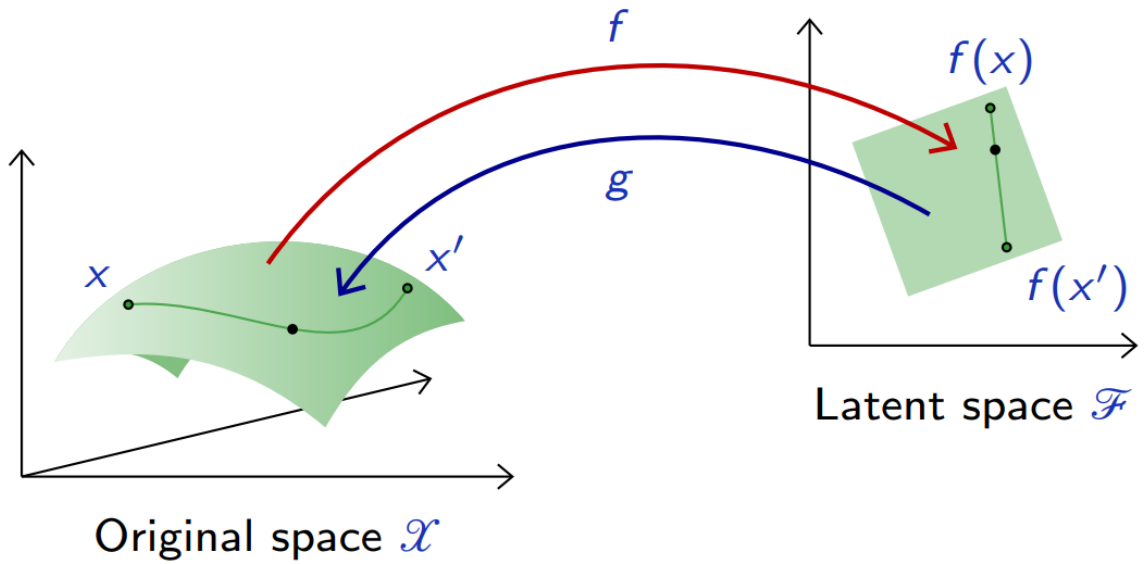


$g \circ f(X)$ (PCA, $d = 32$)



# Interpolation

To get an intuition of the learned latent representation, we can pick two samples $\mathbf{x}$ and $\mathbf{x}'$ at random and interpolate samples along the line in the latent space.

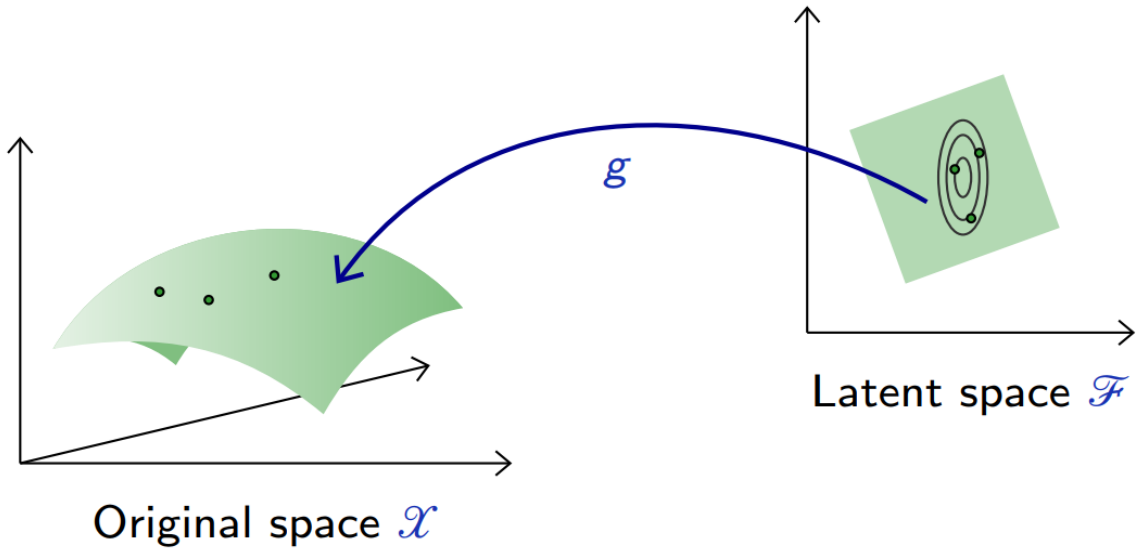


## PCA interpolation ($d = 32$)

## Autoencoder interpolation ($d = 32$)

# Sampling from latent space

The generative capability of the decoder $g$ can be assessed by introducing a (simple) density model $q$ over the latent space $\mathcal{Z}$, sample there, and map the samples into the data space $\mathcal{X}$ with $g$.

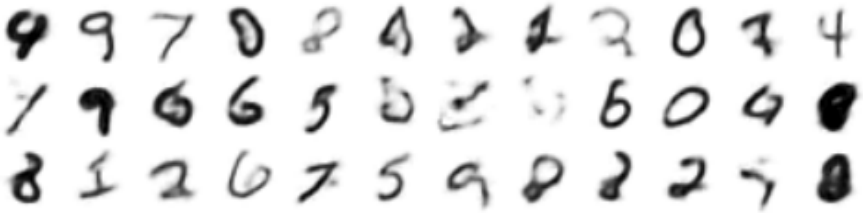


For instance, a factored Gaussian model with diagonal covariance matrix,

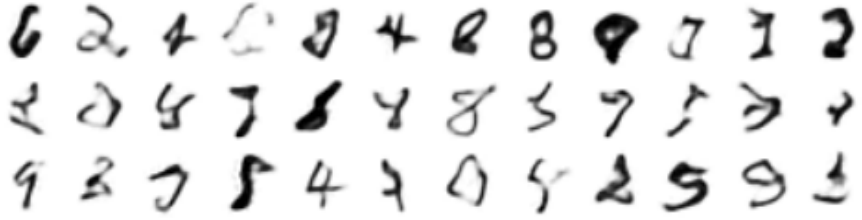
$$q(\mathbf{z}) = \mathcal{N}(\hat{\mu}, \hat{\Sigma}),$$

where both $\hat{\mu}$ and $\hat{\Sigma}$ are estimated on training data.

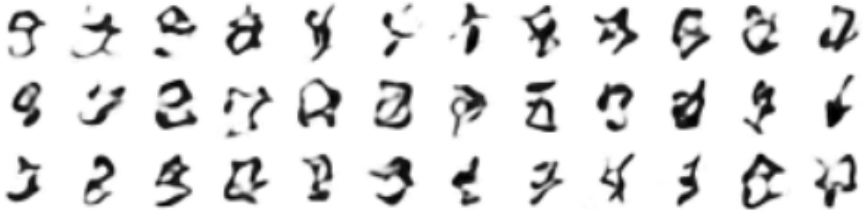
Autoencoder sampling ($d = 8$)



Autoencoder sampling ($d = 16$)



Autoencoder sampling ($d = 32$)



These results are not satisfactory because the density model on the latent space is too simple and inadequate.

Building a good model amounts to our original problem of modeling an empirical distribution, although it may now be in a lower dimension space.

# Generative models

Credits: slides adapted from *"Tutorial on Deep Generative Models"*, Shakir Mohamed and Danilo Rezende, UAI 2017.

A generative model is a probabilistic model $p$ that can be used as a simulator of the data. Its purpose is to generate synthetic but realistic high-dimensional data

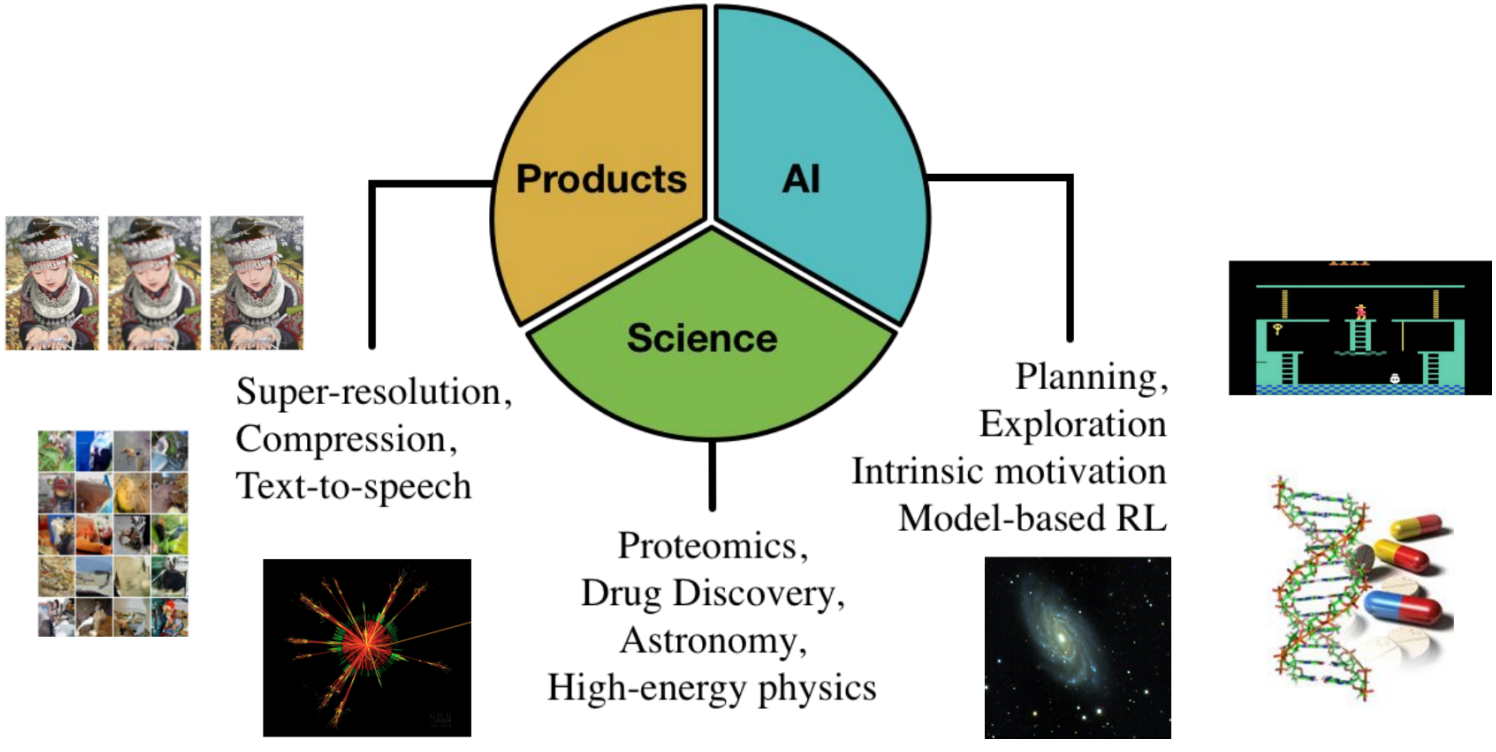
$$\mathbf{x} \sim p(\mathbf{x}; \theta),$$

that is as close as possible from the true but unknown data distribution $p(\mathbf{x})$, but for which we have empirical samples.

## Motivation

Go beyond estimating $p(y|\mathbf{x})$:

- Understand and imagine how the world evolves.
- Recognize objects in the world and their factors of variation.
- Establish concepts for reasoning and decision making.

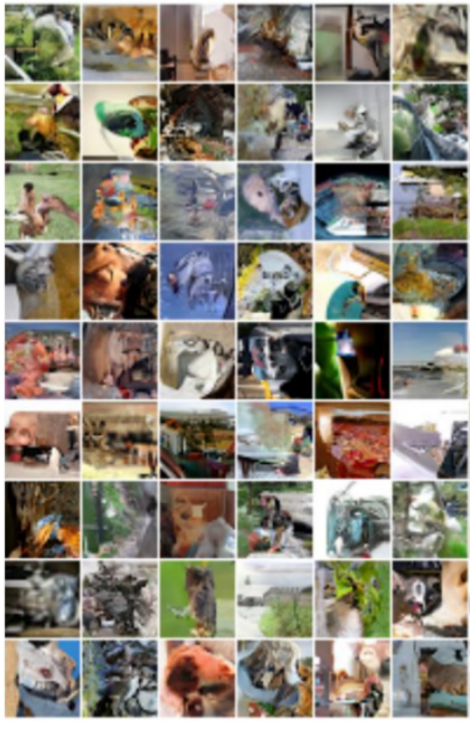



Generative models have a role in many important problems

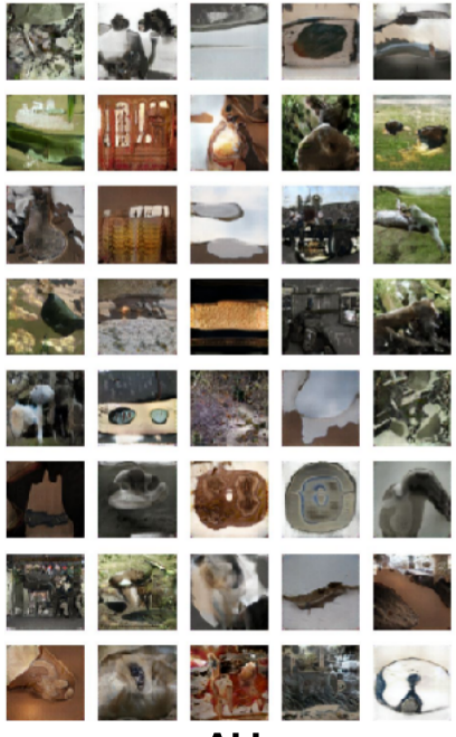
## Image and content generation

Generating images and video content.

DRAW



Pixel RNN

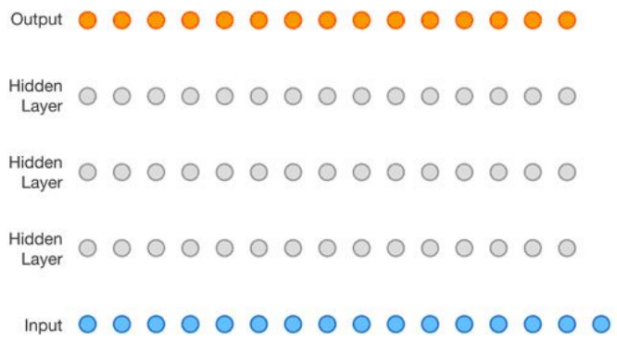


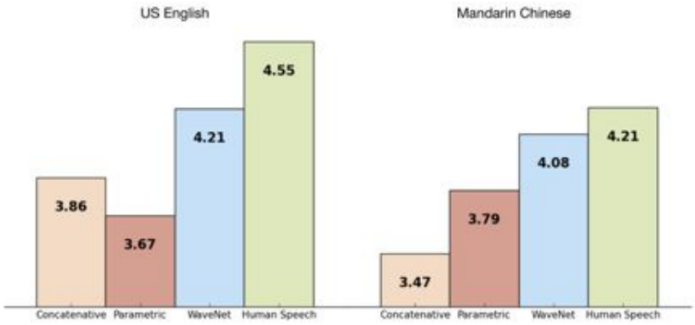
ALI

(Gregor et al, 2015; Oord et al, 2016; Dumoulin et al, 2016)

## Text-to-speech synthesis

Generating audio conditioned on text.




(Oord et al, 2016)

## Communication and compression

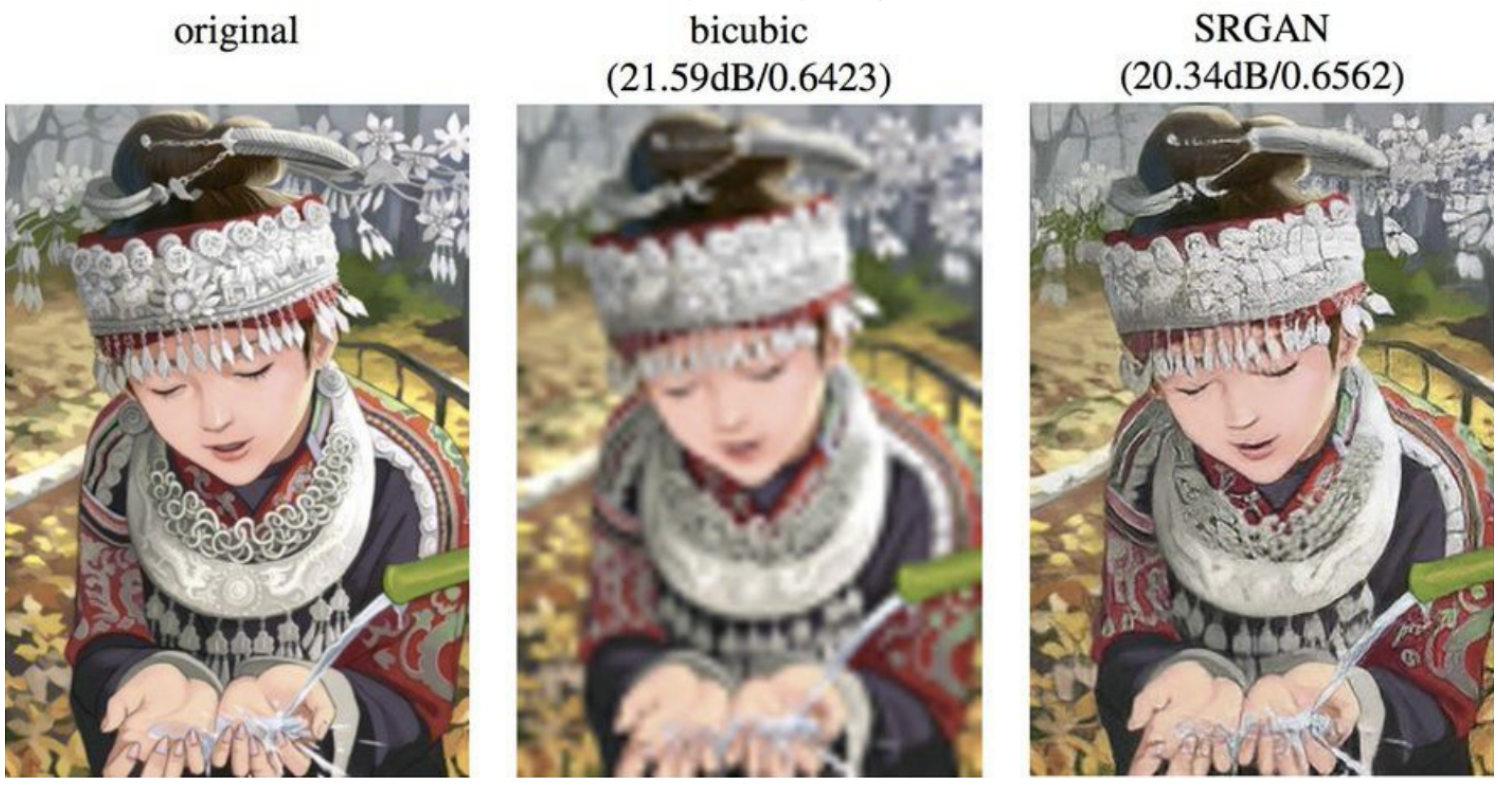
Hierarchical compression of images and other data.



(Gregor et al, 2016)

## Image super-resolution

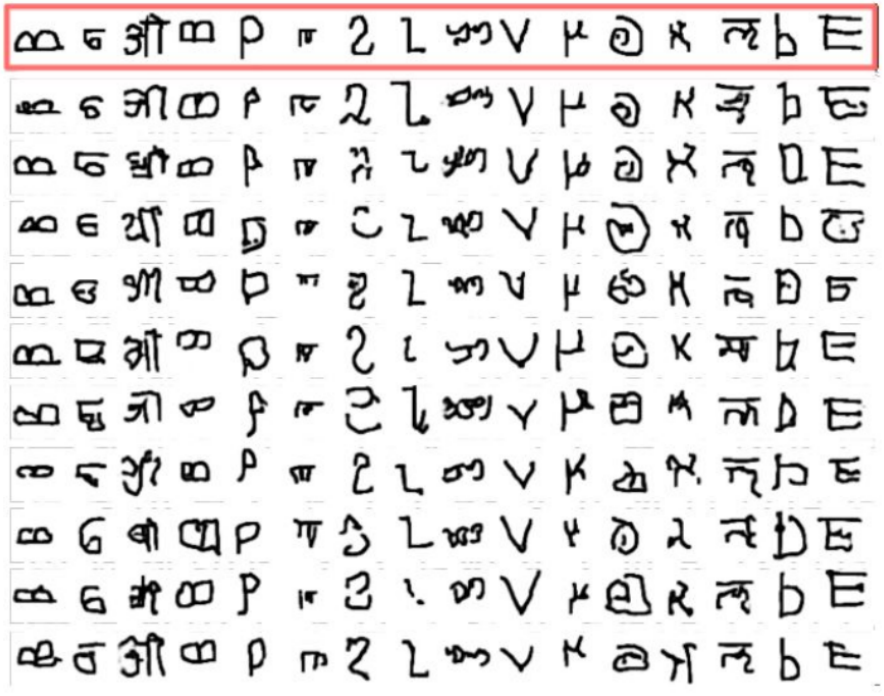
Photo-realistic single image super-resolution.

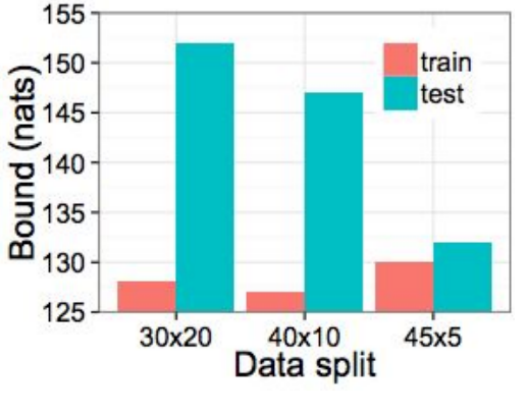



(Ledig et al, 2016)

## One-shot generalization

Rapid generalization of novel concepts.

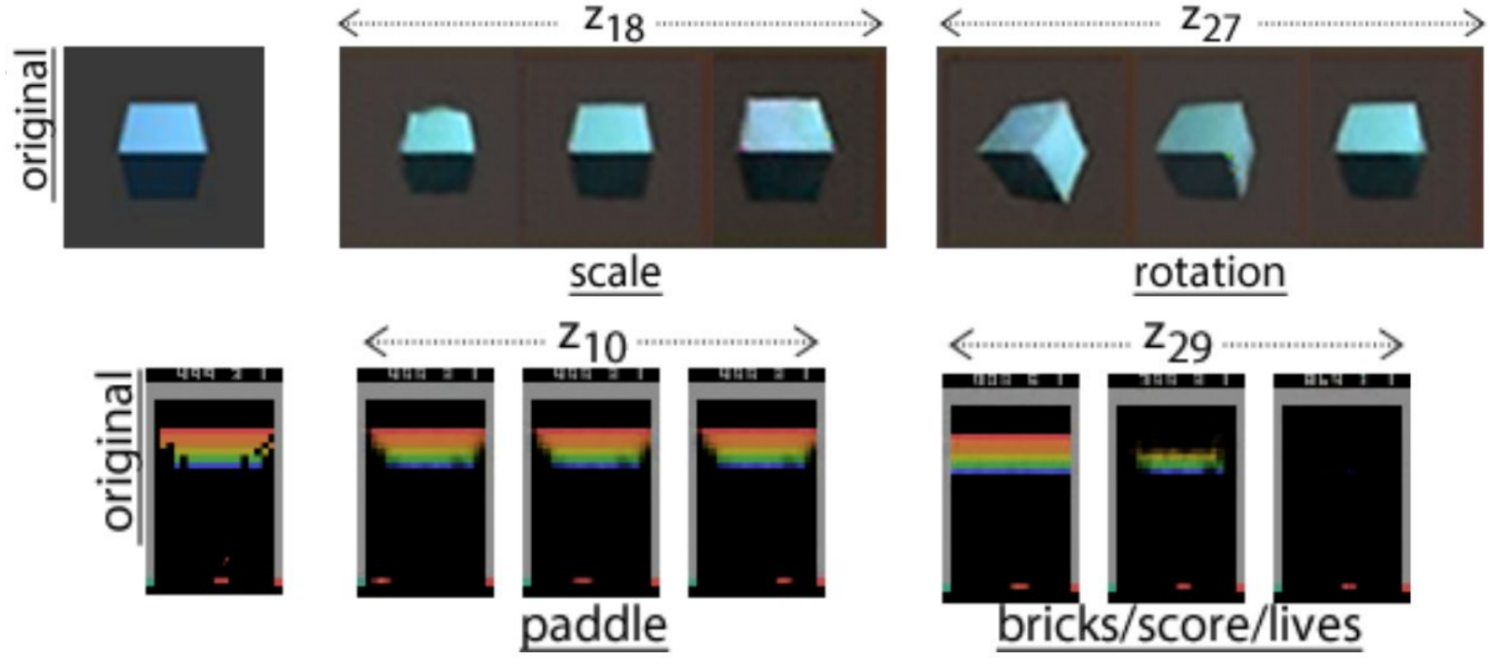





(Gregor et al, 2016)

## Visual concept learning

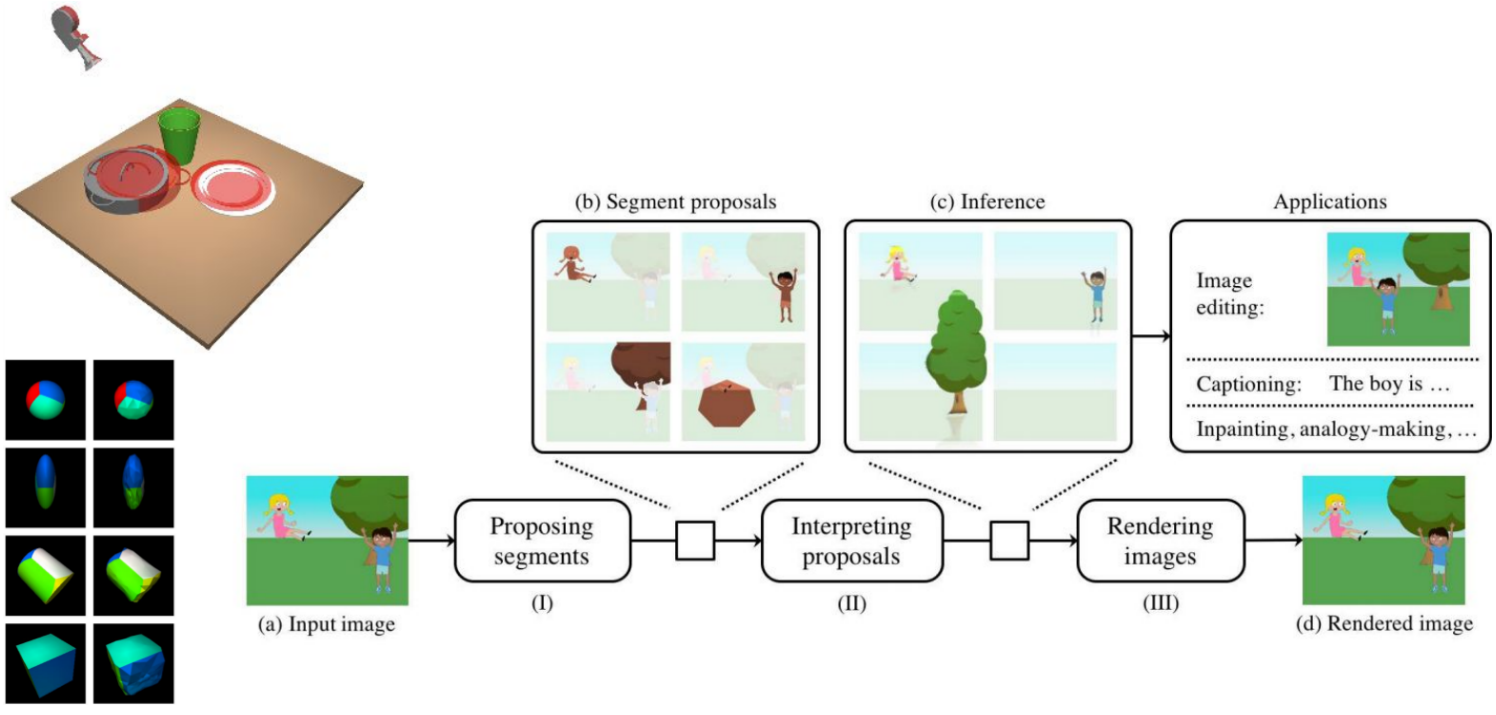
Understanding the factors of variation and invariances.




(Higgins et al, 2017)

## Scene understanding

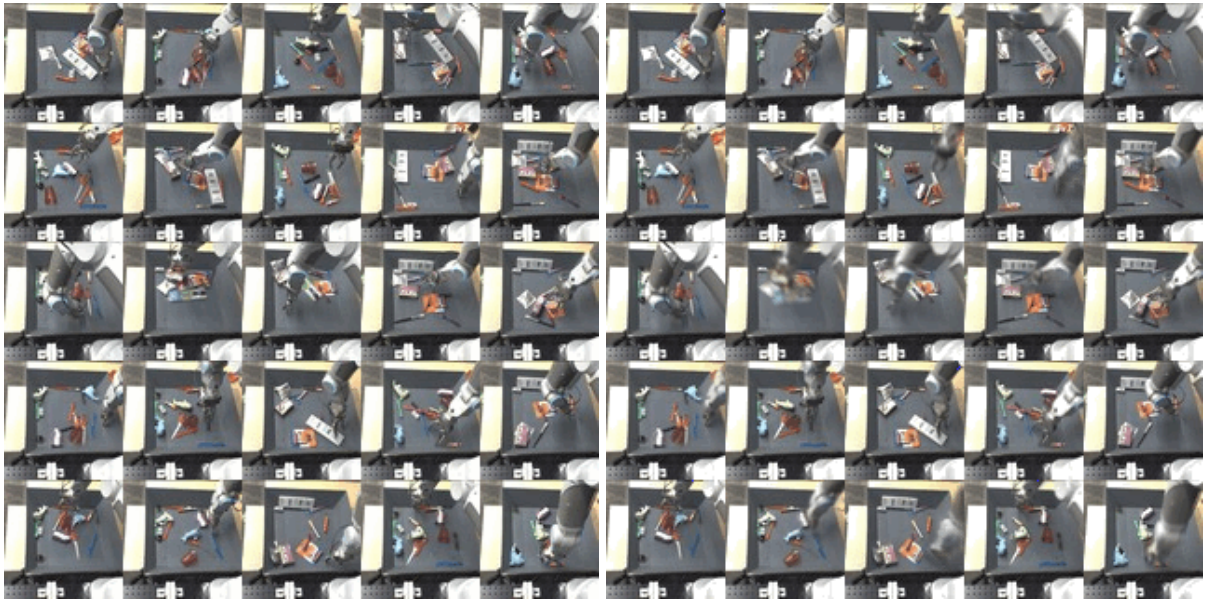
Understanding the components of scenes and their interactions.




(Wu et al, 2017)

## Future simulation

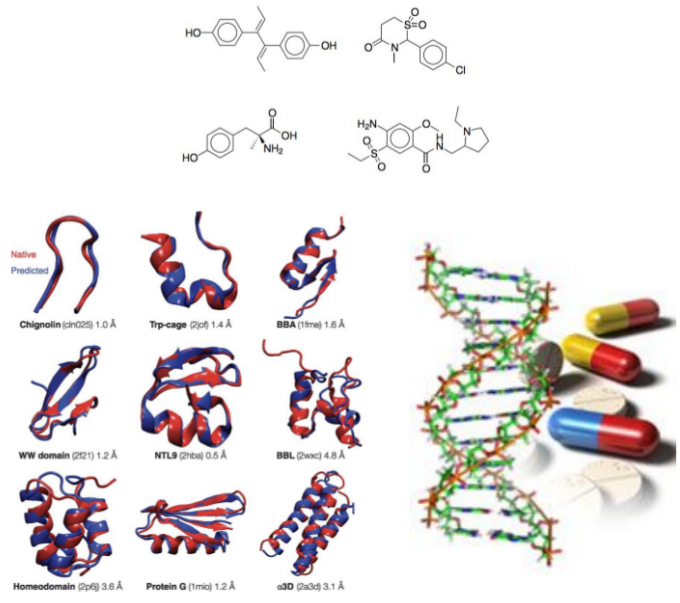
Simulate future trajectories of environments based on actions for planning.

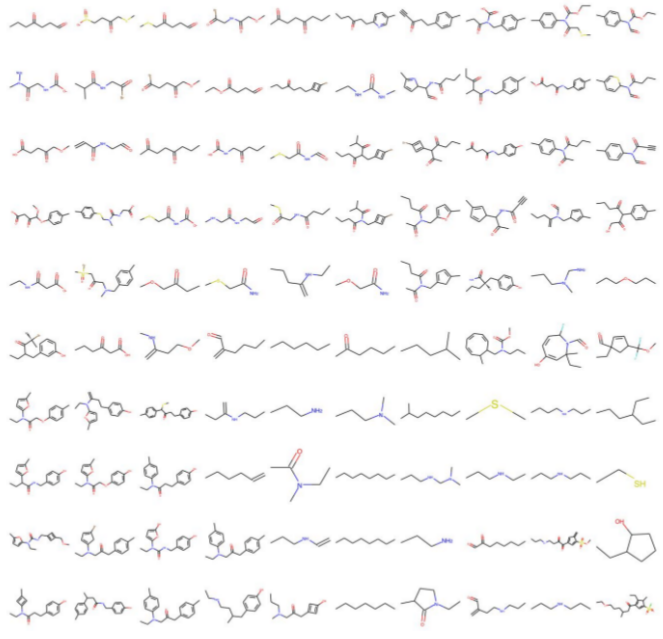


(Finn et al, 2016)

## Drug design and response prediction

Generative models for proposing candidate molecules and for improving prediction through semi-supervised learning.

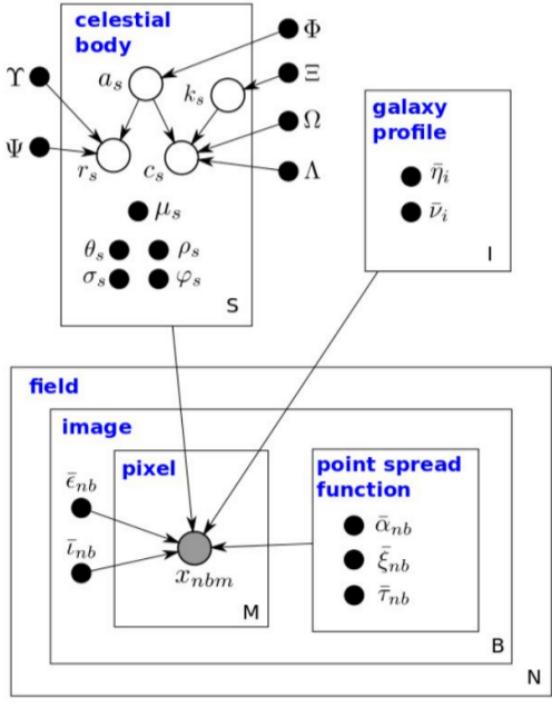


(Gomez-Bombarelli et al, 2016)

## Locating celestial bodies

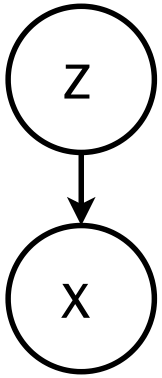
Generative models for applications in astronomy and high-energy physics.




(Regier et al, 2015)

# Variational inference

## Latent variable model




Consider for now a prescribed latent variable model that relates a set of observable variables $\mathbf{x} \in \mathcal{X}$ to a set of unobserved variables $\mathbf{z} \in \mathcal{Z}$.

The probabilistic model is given and motivated by domain knowledge assumptions.

Examples include:

- Linear discriminant analysis
- Bayesian networks
- Hidden Markov models
- Probabilistic programs

The probabilistic model defines a joint probability distribution $p(\mathbf{x}, \mathbf{z})$, which decomposes as

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z}).$$

If we interpret $\mathbf{z}$ as causal factors for the high-dimension representations $\mathbf{x}$, then sampling from $p(\mathbf{x}|\mathbf{z})$ can be interpreted as a stochastic generating process from $\mathcal{Z}$ to $\mathcal{X}$.

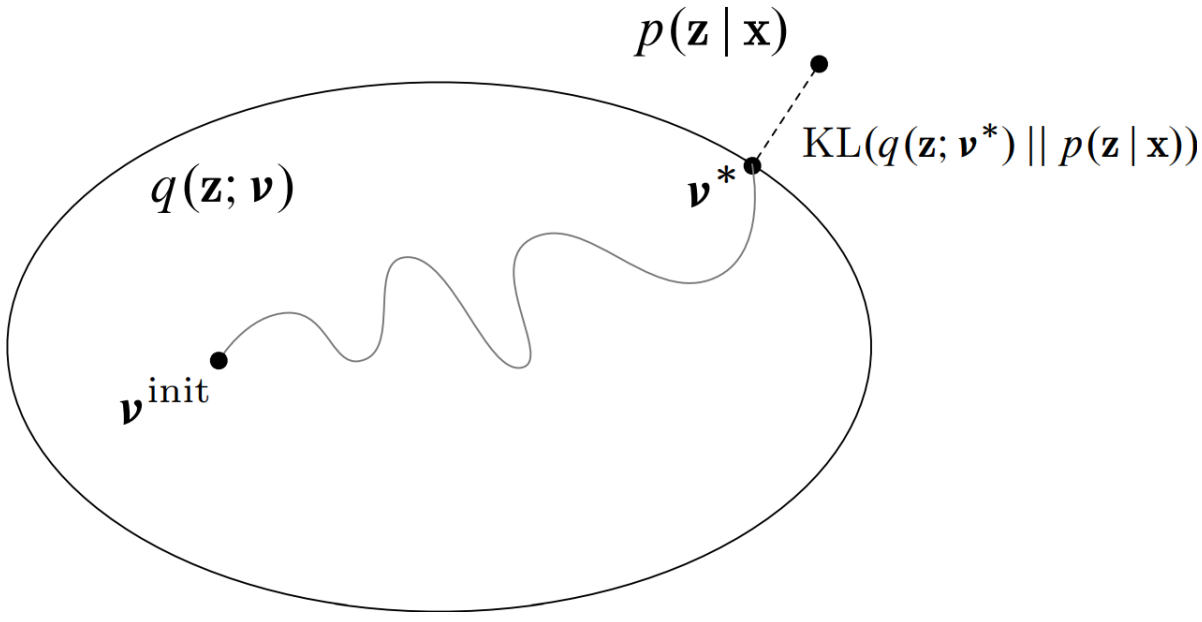
For a given model $p(\mathbf{x}, \mathbf{z})$, inference consists in computing the posterior

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}.$$

For most interesting cases, this is usually intractable since it requires evaluating the evidence

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

# Variational inference




Variational inference turns posterior inference into an optimization problem.

- Consider a family of distributions $q(\mathbf{z}|\mathbf{x}; \nu)$ that approximate the posterior $p(\mathbf{z}|\mathbf{x})$, where the variational parameters $\nu$ index the family of distributions.
- The parameters $\nu$ are fit to minimize the KL divergence between $p(\mathbf{z}|\mathbf{x})$ and the approximation $q(\mathbf{z}|\mathbf{x}; \nu)$.

Formally, we want to minimize

$$\begin{aligned} KL(q(\mathbf{z}|\mathbf{x};\nu)||p(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\nu)}\left[\log \frac{q(\mathbf{z}|\mathbf{x};\nu)}{p(\mathbf{z}|\mathbf{x})}\right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\nu)}\left[\log q(\mathbf{z}|\mathbf{x};\nu) - \log p(\mathbf{x},\mathbf{z})\right] + \log p(\mathbf{x}). \end{aligned}$$

For the same reason as before, the KL divergence cannot be directly minimized because of the $\log p(\mathbf{x})$ term.

However, we can write

$$KL(q(\mathbf{z}|\mathbf{x};\nu)||p(\mathbf{z}|\mathbf{x})) = \log p(\mathbf{x}) - \underbrace{\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\nu)}\left[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x};\nu)\right]}_{\mathrm{ELBO}(\mathbf{x};\nu)}$$

where $\mathrm{ELBO}(\mathbf{x};\nu)$ is called the evidence lower bound objective.

- Since $\log p(\mathbf{x})$ does not depend on $\nu$, it can be considered as a constant, and minimizing the KL divergence is equivalent to maximizing the evidence lower bound, while being computationally tractable.
- Given a dataset $\mathbf{d} = \{\mathbf{x}_i | i = 1, ..., N\}$, the final objective is the sum $\sum_{\{\mathbf{x}_i \in \mathbf{d}\}} \mathrm{ELBO}(\mathbf{x}_i;\nu)$.

Remark that

$$\begin{aligned}\text{ELBO}(\mathbf{x};\nu) &= \mathbb{E}_{q(\mathbf{z};|\mathbf{x}\nu)}\left[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x};\nu)\right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\nu)}\left[\log p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x};\nu)\right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\nu)}\left[\log p(\mathbf{x}|\mathbf{z})\right] - KL(q(\mathbf{z}|\mathbf{x};\nu)||p(\mathbf{z}))\end{aligned}$$

Therefore, maximizing the ELBO:

- encourages distributions to place their mass on configurations of latent variables that explain the observed data (first term);
- encourages distributions close to the prior (second term).

## Optimization

We want

$$\begin{aligned}\nu^* &= \arg\max_{\nu} \mathrm{ELBO}(\mathbf{x};\nu) \\ &= \arg\max_{\nu} \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\nu)}\left[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x};\nu)\right].\end{aligned}$$

We can proceed by gradient ascent, provided we can evaluate $\nabla_\nu \mathrm{ELBO}(\mathbf{x};\nu)$.

In general, this gradient is difficult to compute because the expectation is unknown and the parameters $\nu$ are parameters of the distribution $q(\mathbf{z}|\mathbf{x};\nu)$ we integrate over.

# Variational auto-encoders

So far we assumed a prescribed probabilistic model motivated by domain knowledge. We will now directly learn a stochastic generating process with a neural network.

# Variational auto-encoders

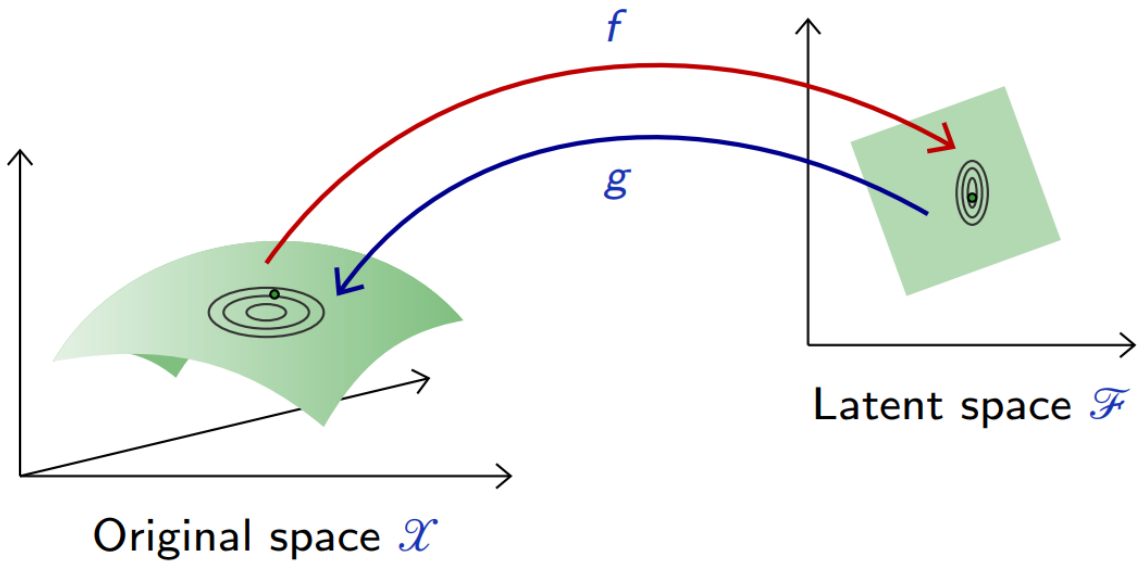
A variational auto-encoder is a deep latent variable model where:

- The likelihood $p(\mathbf{x}|\mathbf{z};\theta)$ is parameterized with a generative network $\text{NN}_\theta$ (or decoder) that takes as input $\mathbf{z}$ and outputs parameters $\phi = \text{NN}_\theta(\mathbf{z})$ to the data distribution. E.g.,

$$\mu, \sigma = \text{NN}_\theta(\mathbf{z})$$
$$p(\mathbf{x}|\mathbf{z};\theta) = \mathcal{N}(\mathbf{x};\mu,\sigma^2\mathbf{I})$$

- The approximate posterior $q(\mathbf{z}|\mathbf{x};\varphi)$ is parameterized with an inference network $\text{NN}_\varphi$ (or encoder) that takes as input $\mathbf{x}$ and outputs parameters $\nu = \text{NN}_\varphi(\mathbf{x})$ to the approximate posterior. E.g.,

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


f
g
Latent space $\mathcal{F}$
Original space $\mathcal{X}$

As before, we can use variational inference, but to jointly optimize the generative and the inference networks parameters $\theta$ and $\varphi$.

We want

$$\begin{aligned}\theta^*, \varphi^* &= \arg\max_{\theta,\varphi} \text{ELBO}(\mathbf{x}; \theta, \varphi) \\ &= \arg\max_{\theta,\varphi} \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)} \left[\log p(\mathbf{x}, \mathbf{z}; \theta) - \log q(\mathbf{z}|\mathbf{x}; \varphi)\right] \\ &= \arg\max_{\theta,\varphi} \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)} \left[\log p(\mathbf{x}|\mathbf{z}; \theta)\right] - KL(q(\mathbf{z}|\mathbf{x}; \varphi)||p(\mathbf{z})).\end{aligned}$$

- Given some generative network $\theta$, we want to put the mass of the latent variables, by adjusting $\varphi$, such that they explain the observed data, while remaining close to the prior.
- Given some inference network $\varphi$, we want to put the mass of the observed variables, by adjusting $\theta$, such that they are well explained by the latent variables.

Unbiased gradients of the ELBO with respect to the generative model parameters $\theta$ are simple to obtain:

$$\begin{aligned}\nabla_\theta \text{ELBO}(\mathbf{x};\theta,\varphi) &= \nabla_\theta \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[\log p(\mathbf{x},\mathbf{z};\theta) - \log q(\mathbf{z}|\mathbf{x};\varphi)\right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[\nabla_\theta(\log p(\mathbf{x},\mathbf{z};\theta) - \log q(\mathbf{z}|\mathbf{x};\varphi))\right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[\nabla_\theta \log p(\mathbf{x},\mathbf{z};\theta)\right],\end{aligned}$$

which can be estimated with Monte Carlo integration.

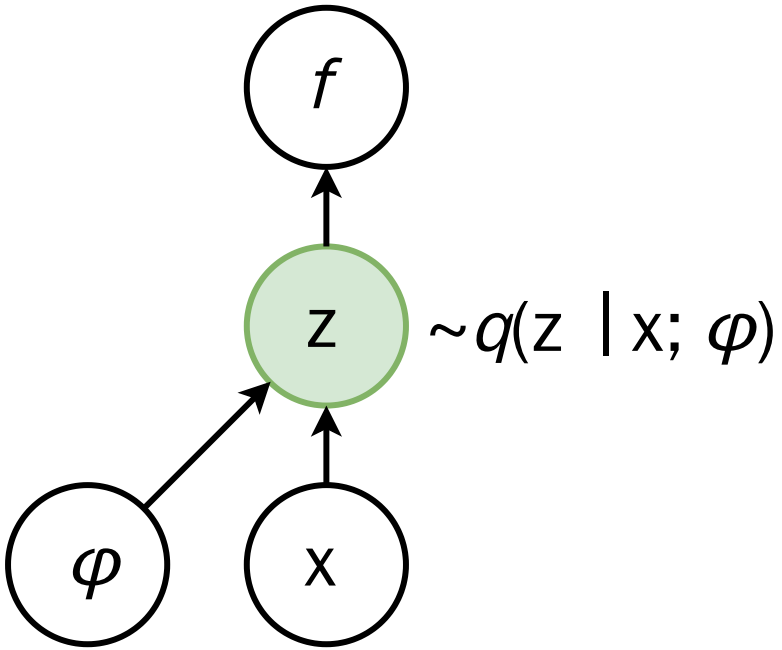
However, gradients with respect to the inference model parameters $\varphi$ are more difficult to obtain:

$$\begin{aligned}\nabla_\varphi \text{ELBO}(\mathbf{x};\theta,\varphi) &= \nabla_\varphi \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[\log p(\mathbf{x},\mathbf{z};\theta) - \log q(\mathbf{z}|\mathbf{x};\varphi)\right] \\ &\neq \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[\nabla_\varphi(\log p(\mathbf{x},\mathbf{z};\theta) - \log q(\mathbf{z}|\mathbf{x};\varphi))\right]\end{aligned}$$

Let us abbreviate

$$\begin{aligned}\mathrm{ELBO}(\mathbf{x};\theta,\varphi) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[\log p(\mathbf{x},\mathbf{z};\theta) - \log q(\mathbf{z}|\mathbf{x};\varphi)\right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[f(\mathbf{x},\mathbf{z};\varphi)\right].\end{aligned}$$

We have




We cannot backpropagate through the stochastic node $\mathbf{z}$ to compute $\nabla_\varphi f$!

# Reparameterization trick

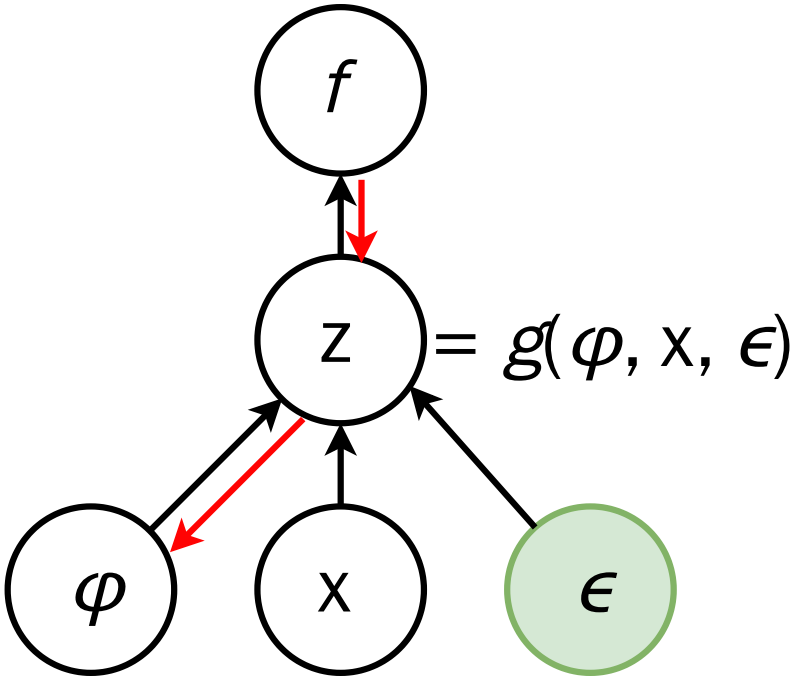
The reparameterization trick consists in re-expressing the variable

$$\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}; \varphi)$$

as some differentiable and invertible transformation of another random variable $\epsilon$ given $\mathbf{x}$ and $\varphi$,

$$\mathbf{z} = g(\varphi, \mathbf{x}, \epsilon),$$

such that the distribution of $\epsilon$ is independent of $\mathbf{x}$ or $\varphi$.




For example, if $q(\mathbf{z}|\mathbf{x};\varphi) = \mathcal{N}(\mathbf{z};\mu(\mathbf{x};\varphi),\sigma^2(\mathbf{x};\varphi))$, where $\mu(\mathbf{x};\varphi)$ and $\sigma^2(\mathbf{x};\varphi)$ are the outputs of the inference network $NN_\varphi$, then a common reparameterization is:

$$p(\epsilon) = \mathcal{N}(\epsilon;\mathbf{0},\mathbf{I})$$
$$\mathbf{z} = \mu(\mathbf{x};\varphi) + \sigma(\mathbf{x};\varphi) \odot \epsilon$$

Given such a change of variable, the ELBO can be rewritten as:

$$\begin{aligned}\mathrm{ELBO}(\mathbf{x};\theta,\varphi) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[f(\mathbf{x},\mathbf{z};\varphi)\right] \\ &= \mathbb{E}_{p(\epsilon)}\left[f(\mathbf{x},g(\varphi,\mathbf{x},\epsilon);\varphi)\right]\end{aligned}$$

Therefore,

$$\begin{aligned}\nabla_\varphi \mathrm{ELBO}(\mathbf{x};\theta,\varphi) &= \nabla_\varphi \mathbb{E}_{p(\epsilon)}\left[f(\mathbf{x},g(\varphi,\mathbf{x},\epsilon);\varphi)\right] \\ &= \mathbb{E}_{p(\epsilon)}\left[\nabla_\varphi f(\mathbf{x},g(\varphi,\mathbf{x},\epsilon);\varphi)\right],\end{aligned}$$

which we can now estimate with Monte Carlo integration.

The last required ingredient is the evaluation of the likelihood $q(\mathbf{z}|\mathbf{x};\varphi)$ given the change of variable $g$. As long as $g$ is invertible, we have:

$$\log q(\mathbf{z}|\mathbf{x};\varphi) = \log p(\epsilon) - \log \left|\det\left(\frac{\partial \mathbf{z}}{\partial \epsilon}\right)\right|.$$

# Example

Consider the following setup:

- Generative model:

$$
\begin{aligned}
\mathbf{z} &\in \mathbb{R}^J \\
p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}) \\
p(\mathbf{x}|\mathbf{z}; \theta) &= \mathcal{N}(\mathbf{x}; \mu(\mathbf{z}; \theta), \sigma^2(\mathbf{z}; \theta)\mathbf{I}) \\
\mu(\mathbf{z}; \theta) &= \mathbf{W}_2^T \mathbf{h} + \mathbf{b}_2 \\
\log \sigma^2(\mathbf{z}; \theta) &= \mathbf{W}_3^T \mathbf{h} + \mathbf{b}_3 \\
\mathbf{h} &= \mathrm{ReLU}(\mathbf{W}_1^T \mathbf{z} + \mathbf{b}_1) \\
\theta &= \{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \mathbf{W}_3, \mathbf{b}_3\}
\end{aligned}
$$

- Inference model:

$$
\begin{aligned}
q(\mathbf{z}|\mathbf{x};\varphi) &= \mathcal{N}(\mathbf{z};\mu(\mathbf{x};\varphi),\sigma^2(\mathbf{x};\varphi)\mathbf{I}) \\
p(\epsilon) &= \mathcal{N}(\epsilon;\mathbf{0},\mathbf{I}) \\
\mathbf{z} &= \mu(\mathbf{x};\varphi) + \sigma(\mathbf{x};\varphi) \odot \epsilon \\
\mu(\mathbf{x};\varphi) &= \mathbf{W}_5^T\mathbf{h} + \mathbf{b}_5 \\
\log\sigma^2(\mathbf{x};\varphi) &= \mathbf{W}_6^T\mathbf{h} + \mathbf{b}_6 \\
\mathbf{h} &= \mathrm{ReLU}(\mathbf{W}_4^T\mathbf{x} + \mathbf{b}_4) \\
\varphi &= \{\mathbf{W}_4, \mathbf{b}_4, \mathbf{W}_5, \mathbf{b}_5, \mathbf{W}_6, \mathbf{b}_6\}
\end{aligned}
$$

Note that there is no restriction on the generative and inference network architectures. They could as well be arbitrarily complex convolutional networks.

Plugging everything together, the objective can be expressed as:

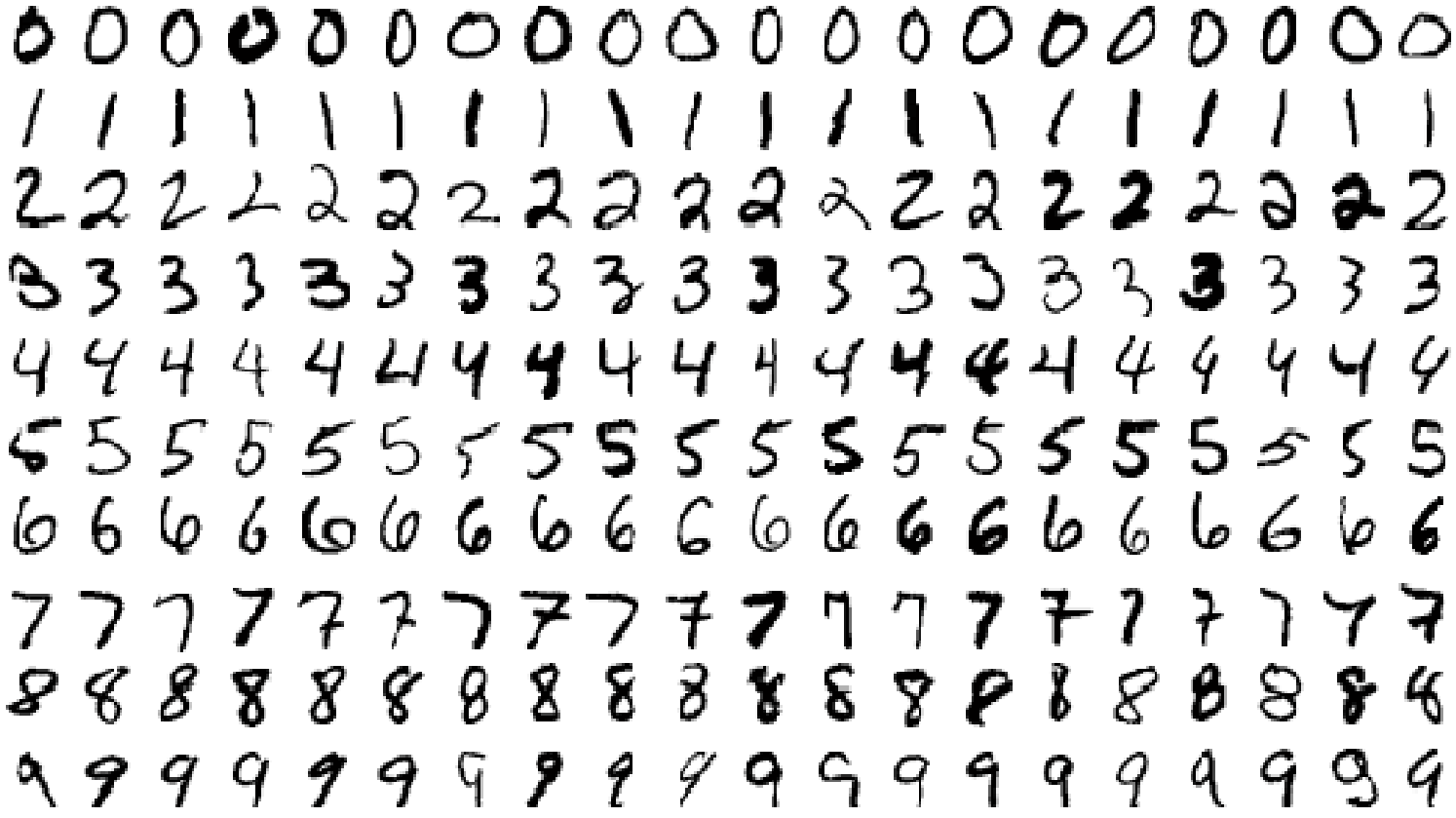
$$\begin{aligned}\mathrm{ELBO}(\mathbf{x};\theta,\varphi) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[\log p(\mathbf{x},\mathbf{z};\theta) - \log q(\mathbf{z}|\mathbf{x};\varphi)\right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[\log p(\mathbf{x}|\mathbf{z};\theta)\right] - KL(q(\mathbf{z}|\mathbf{x};\varphi)||p(\mathbf{z})) \\ &= \mathbb{E}_{p(\epsilon)}\left[\log p(\mathbf{x}|\mathbf{z}=g(\varphi,\mathbf{x},\epsilon);\theta)\right] - KL(q(\mathbf{z}|\mathbf{x};\varphi)||p(\mathbf{z}))\end{aligned}$$

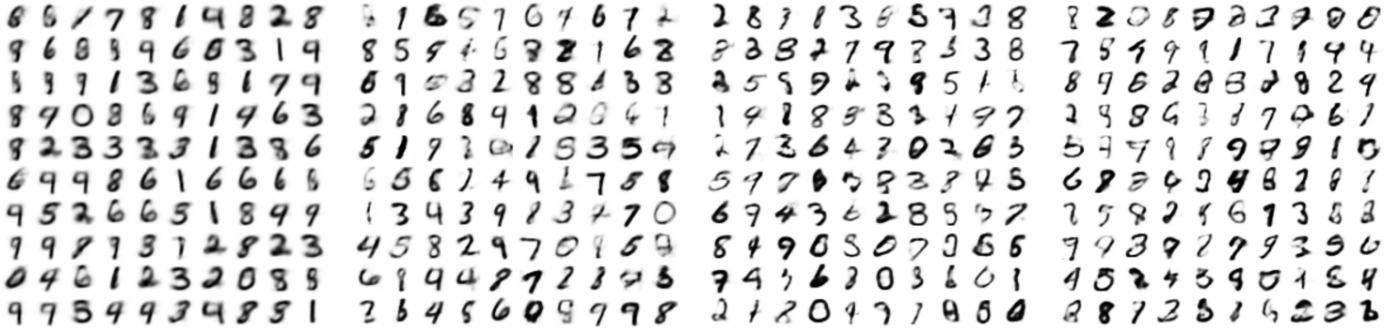
where the KL divergence can be expressed analytically as

$$KL(q(\mathbf{z}|\mathbf{x};\varphi)||p(\mathbf{z})) = \frac{1}{2}\sum_{j=1}^{J}\left(1+\log(\sigma_j^2(\mathbf{x};\varphi)) - \mu_j^2(\mathbf{x};\varphi) - \sigma_j^2(\mathbf{x};\varphi)\right),$$

which allows to evaluate its derivative without approximation.

Consider as data $\mathbf{d}$ the MNIST digit dataset:



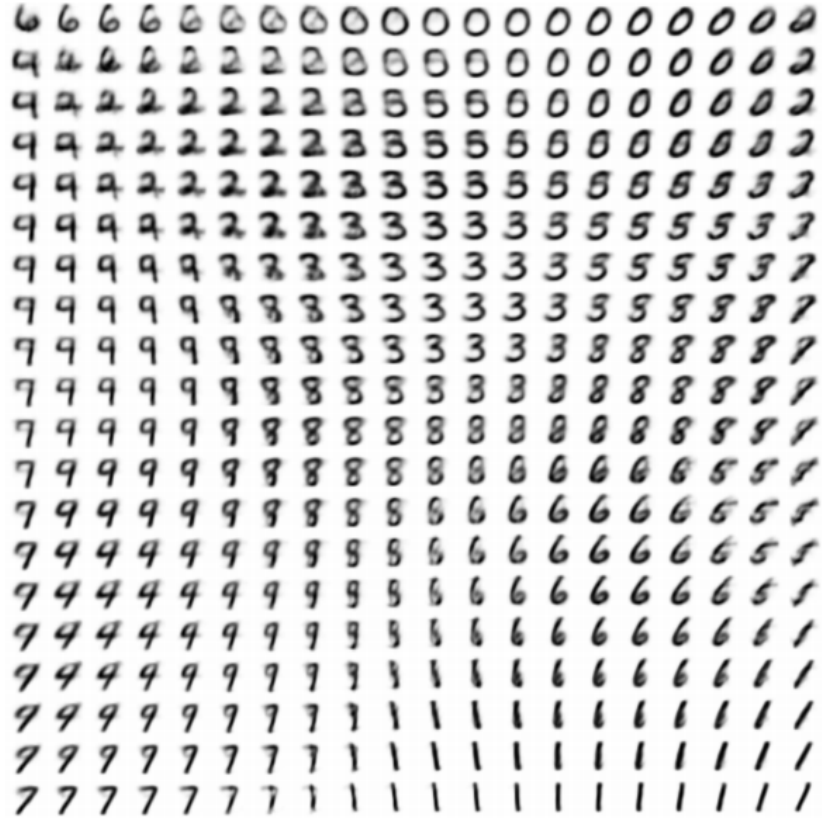


(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

(Kingma and Welling, 2013)

(a) Learned Frey Face manifold



(b) Learned MNIST manifold

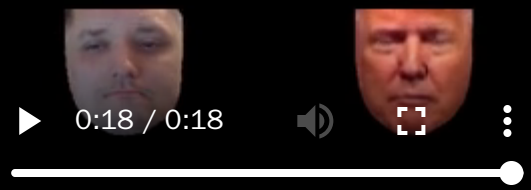
Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

(Kingma and Welling, 2013)

# Applications of (variational) AEs

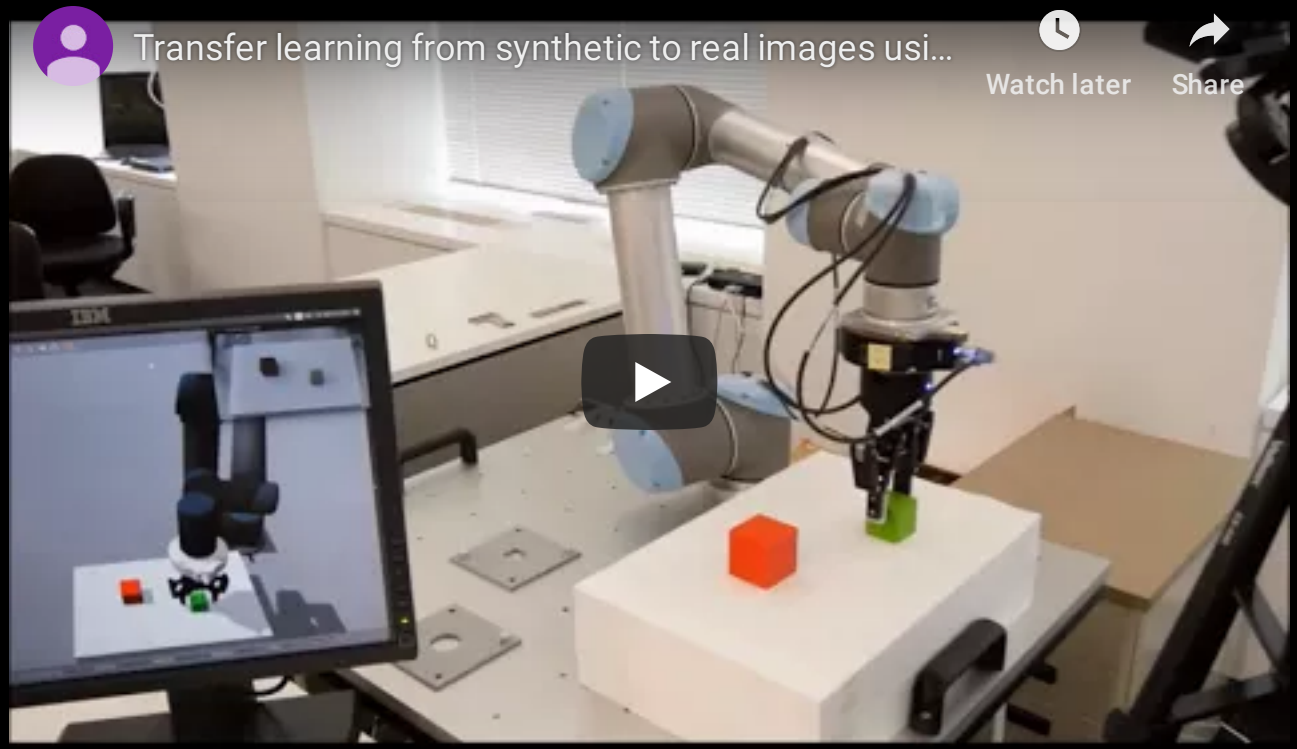

Random walks in latent space.

(Alex Radford, 2015)




Impersonation by encoding-decoding an unknown face.

(Kamil Czarnogórski, 2016)




(Inoue et al, 2017)

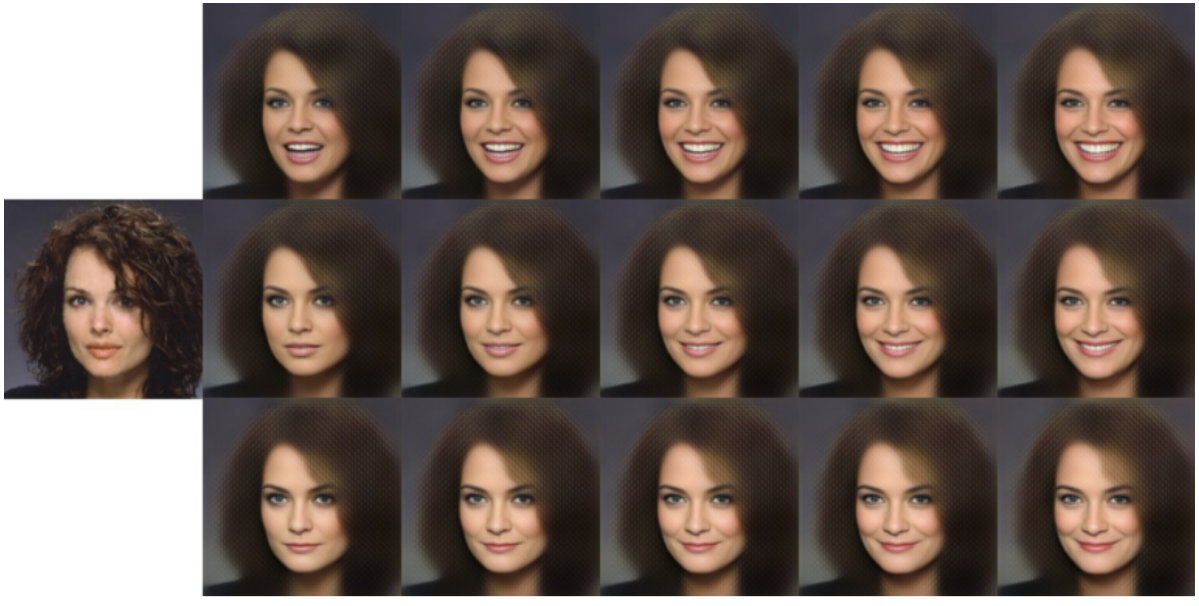


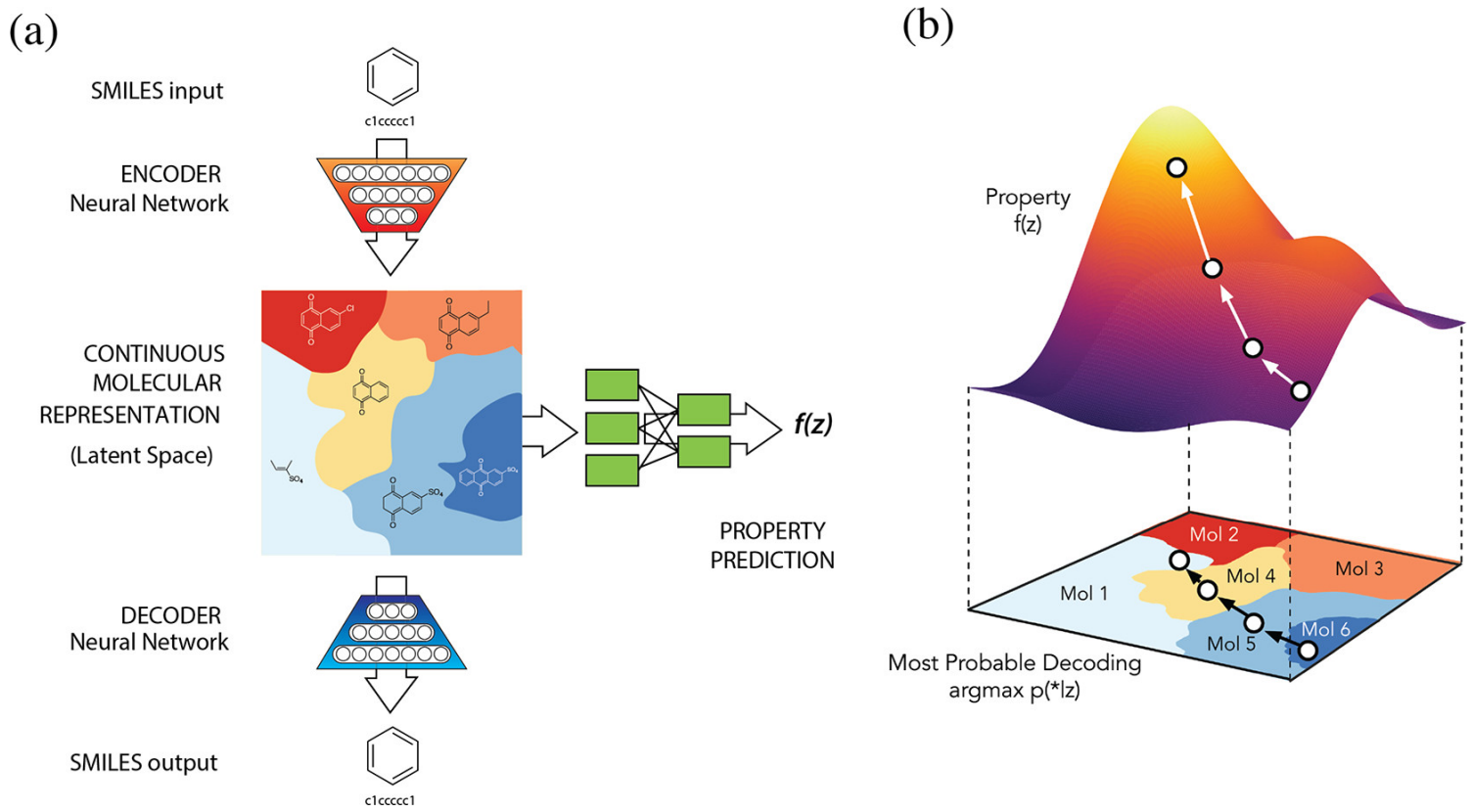
**Figure 7: Decoupling attribute vectors for smiling (x-axis) and mouth open (y-axis) allows for more flexible latent space transformations. Input shown at left with reconstruction adjacent. (model: VAE from Lamb 16 on CelebA)**

(Tom White, 2016)

| |
|---|
| **“ i want to talk to you . ”** |
| *“i want to be with you . ”* |
| *“i do n’t want to be with you . ”* |
| *i do n’t want to be with you .* |
| **she did n’t want to be with him .** |
| **he was silent for a long moment .** |
| *he was silent for a moment .* |
| *it was quiet for a moment .* |
| *it was dark and cold .* |
| *there was a pause .* |
| **it was my turn .** |

Table 8: Paths between pairs of random points in VAE space: Note that intermediate sentences are grammatical, and that topic and syntactic structure are usually locally consistent.

(Bowman et al, 2015)




Design of new molecules with desired chemical properties.
(Gomez-Bombarelli et al, 2016)

The end.

# References


- Mohamed and Rezende, "Tutorial on Deep Generative Models", UAI 2017.
- Blei et al, "Variational inference: Foundations and modern methods", 2016.
- Kingma and Welling, "Auto-Encoding Variational Bayes", 2013.