Deep Learning

Lecture 5: Variational auto-encoders

Gilles Louppe
g.louppe@uliege.be

Outline

Goals: Learn models of the data itself.

- Generative models
- Variational inference
- Variational auto-encoders
- Generative adversarial networks (lecture 6)

Generative models

Slides adapted from "Tutorial on Deep Generative Models"
(Shakir Mohamed and Danilo Rezende, UAI 2017).

Generative models

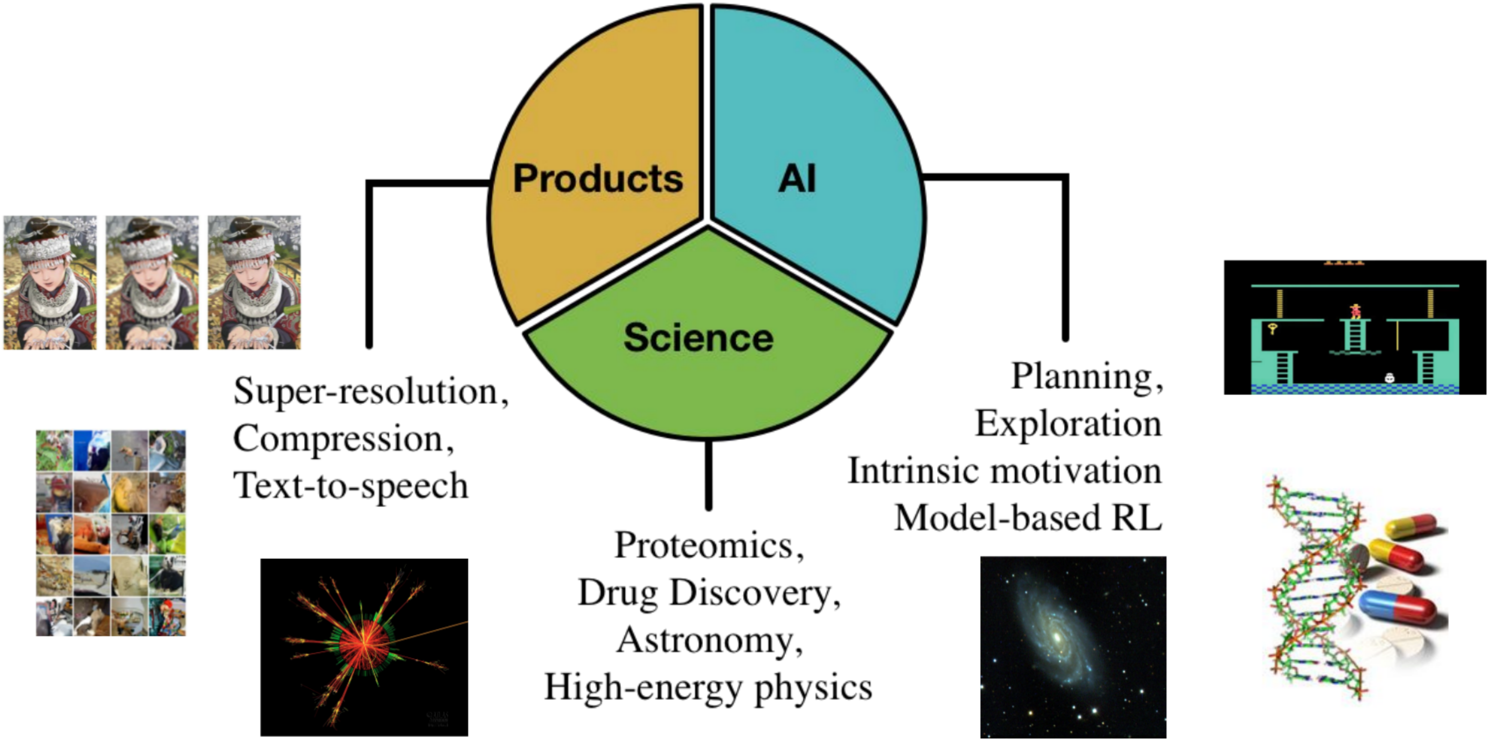
A generative model is a probabilistic model p that can be used as a simulator of the data. Its purpose is to generate synthetic but realistic high-dimension data

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

that is as close as possible from the true but unknown data distribution $p_r(\mathbf{x})$.

Goals:

- Learn $p(\mathbf{x}; \theta)$ (i.e., 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.



Products

AI

Science

Super-resolution,
Compression,
Text-to-speech

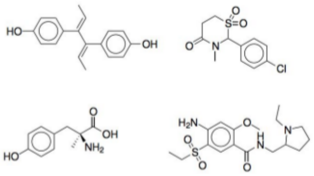
Planning,
Exploration
Intrinsic motivation
Model-based RL

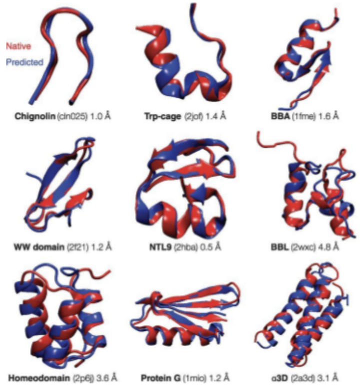
Proteomics,
Drug Discovery,
Astronomy,
High-energy physics

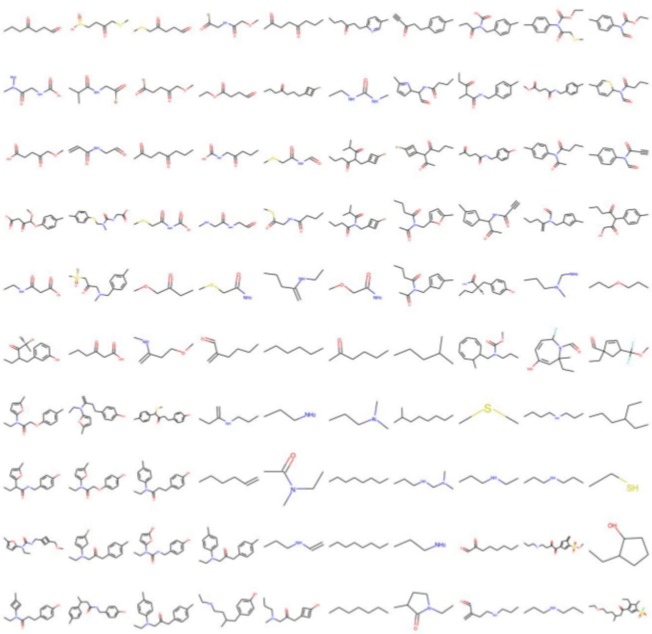
Generative models have a role in many important problems

Drug design and response prediction

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



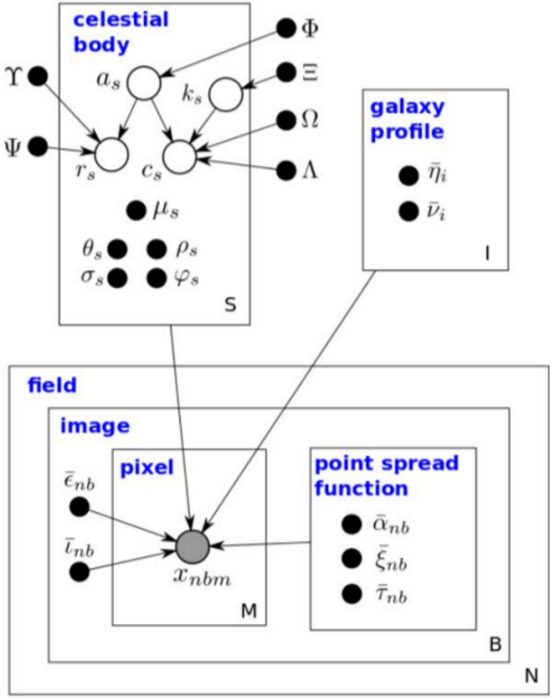




(Gomez-Bombarelli et al, 2016)

Locating celestial bodies

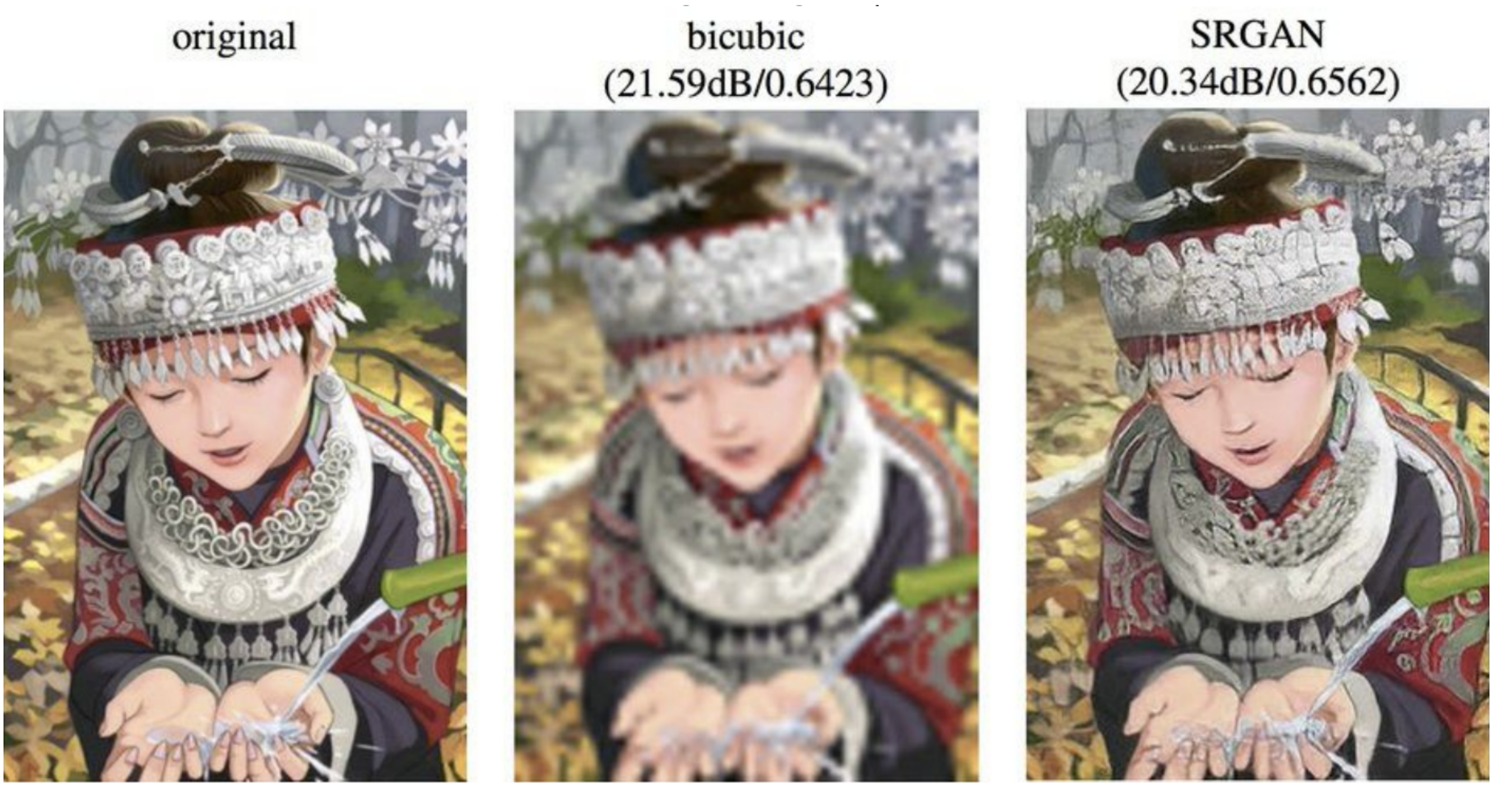
Generative models for applications in astronomy and high-energy physics.



(Regier et al, 2015)

Image super-resolution

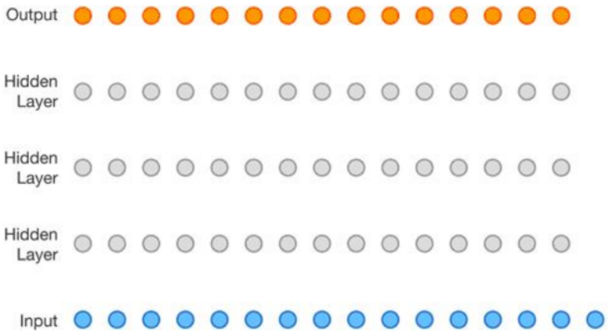
Photo-realistic single image super-resolution.

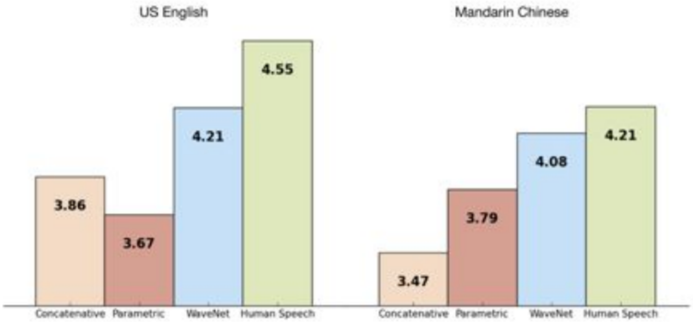


(Ledig et al, 2016)

Text-to-speech synthesis

Generating audio conditioned on text.



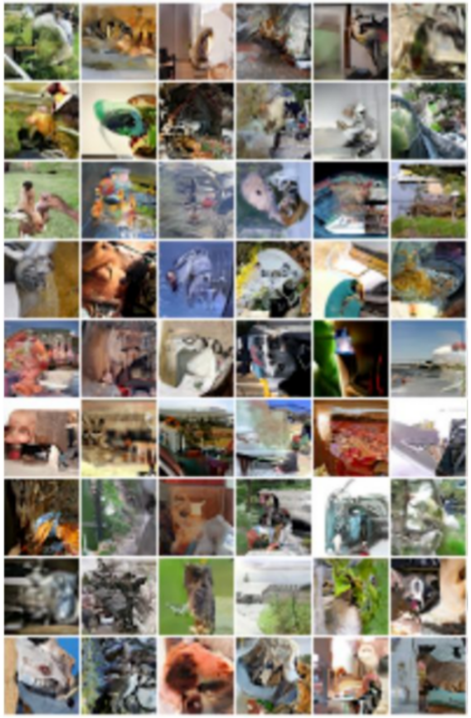


(Oord et al, 2016)

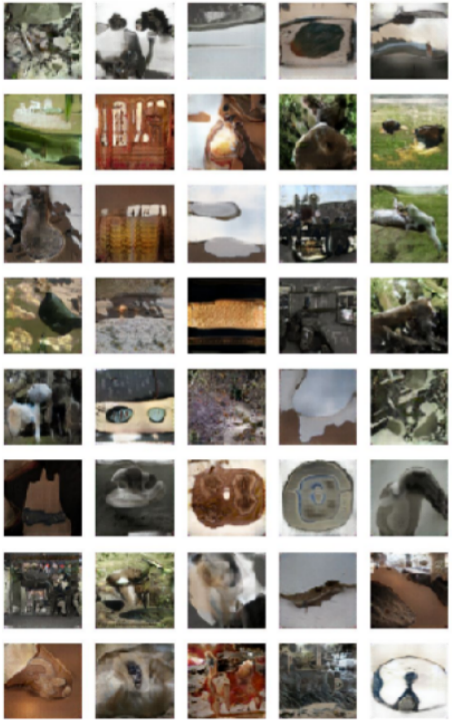
Image and content generation

Generating images and video content.

DRAW



Pixel RNN

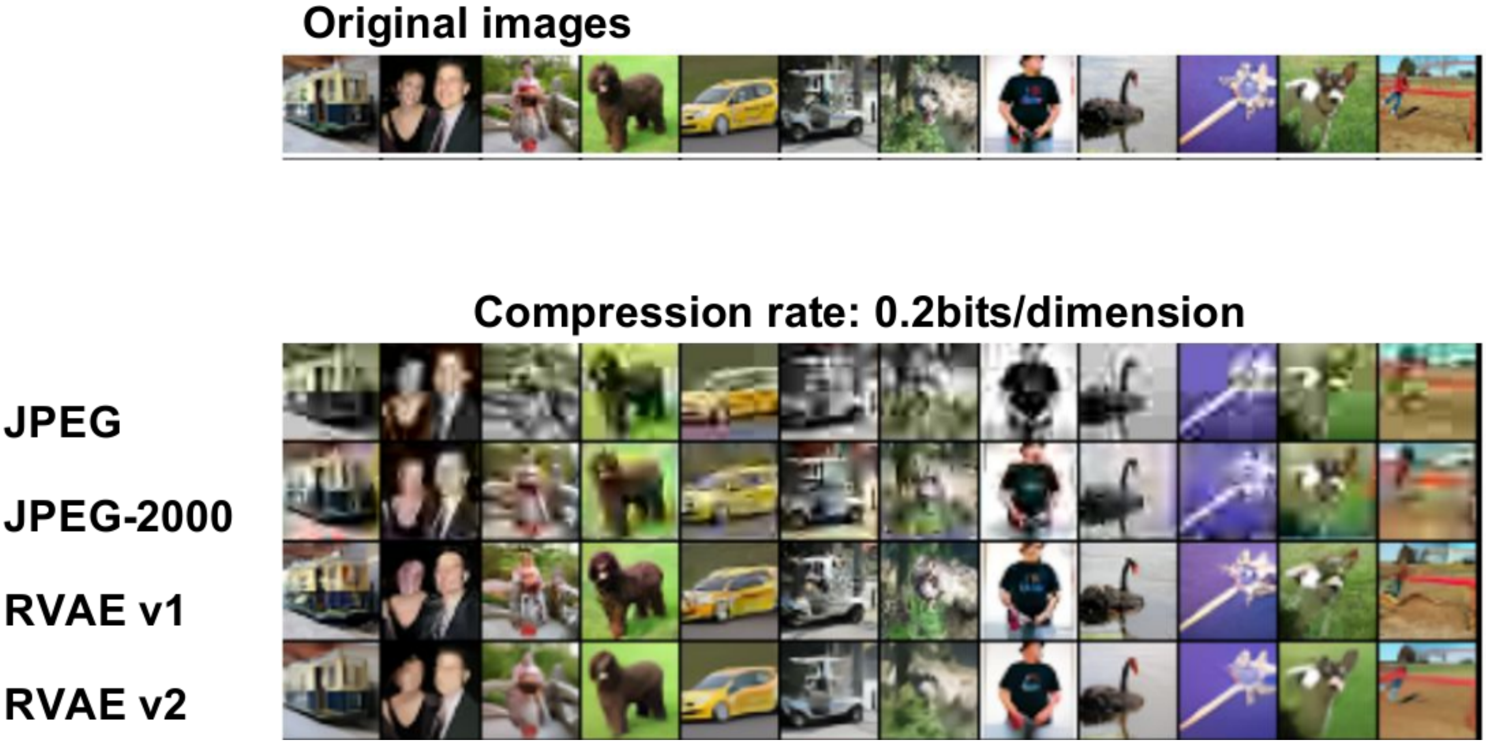


ALI

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

Communication and compression

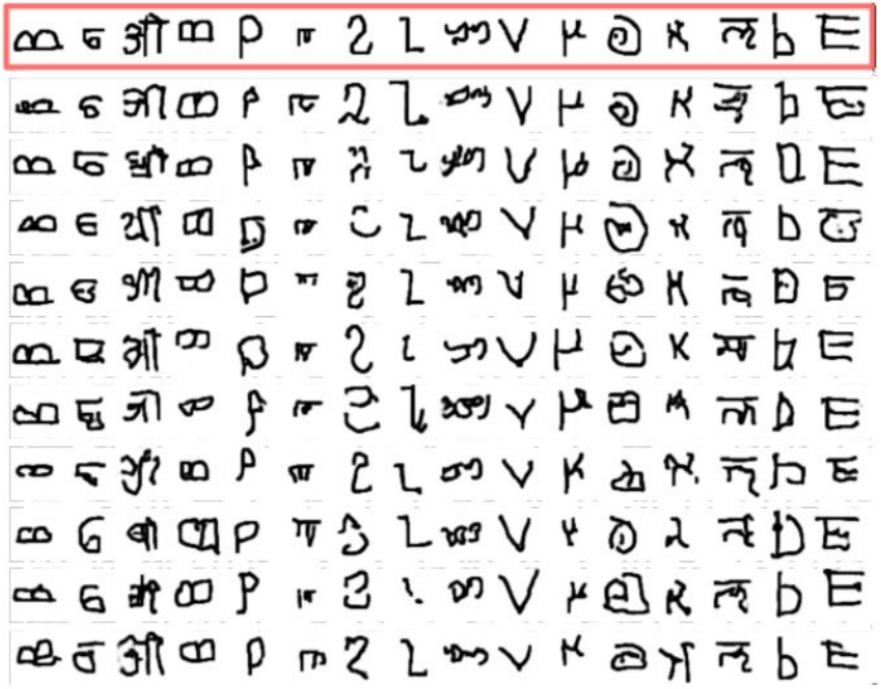
Hierarchical compression of images and other data.

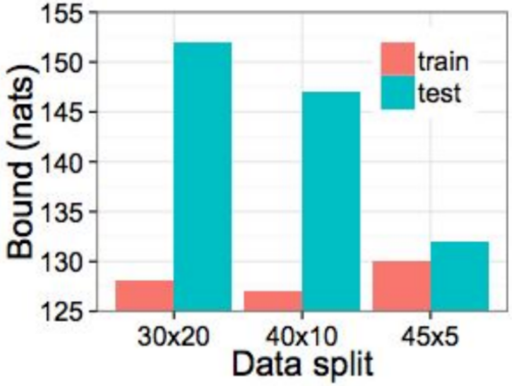


(Gregor et al, 2016)

One-shot generalization

Rapid generalization of novel concepts.

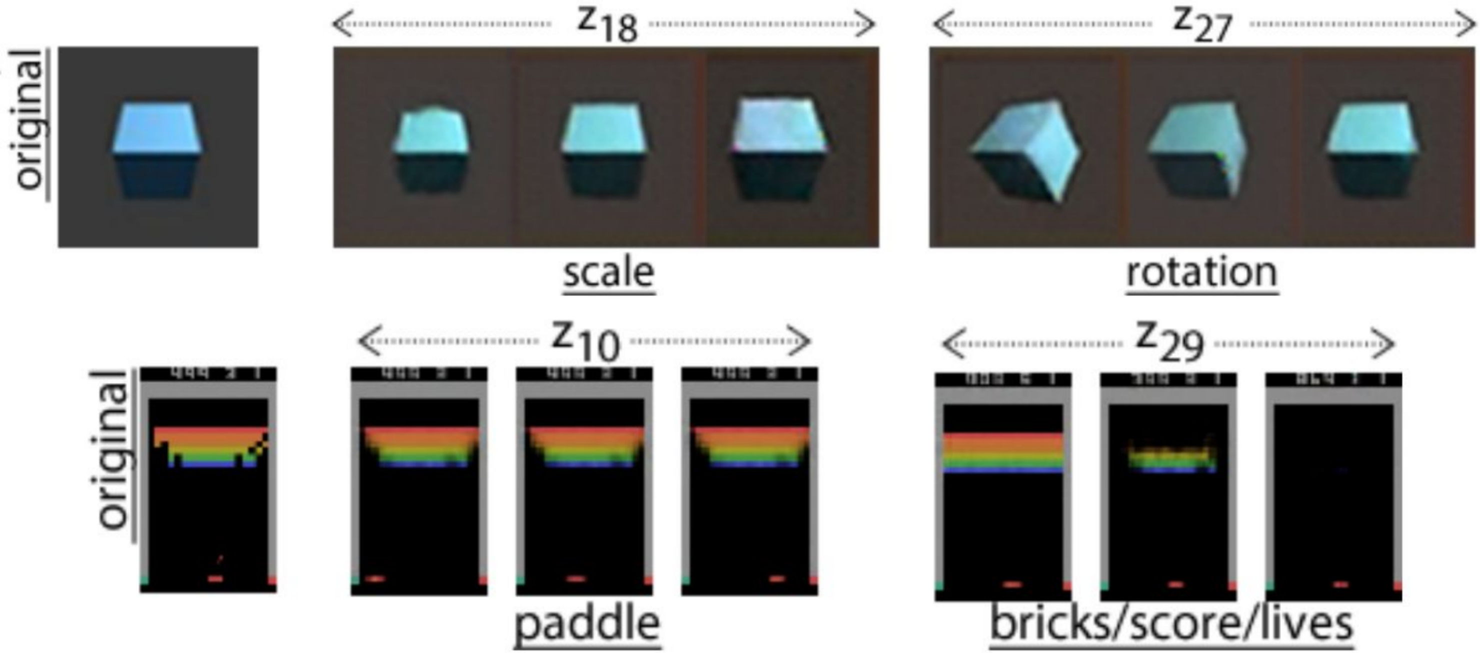




(Gregor et al, 2016)

Visual concept learning

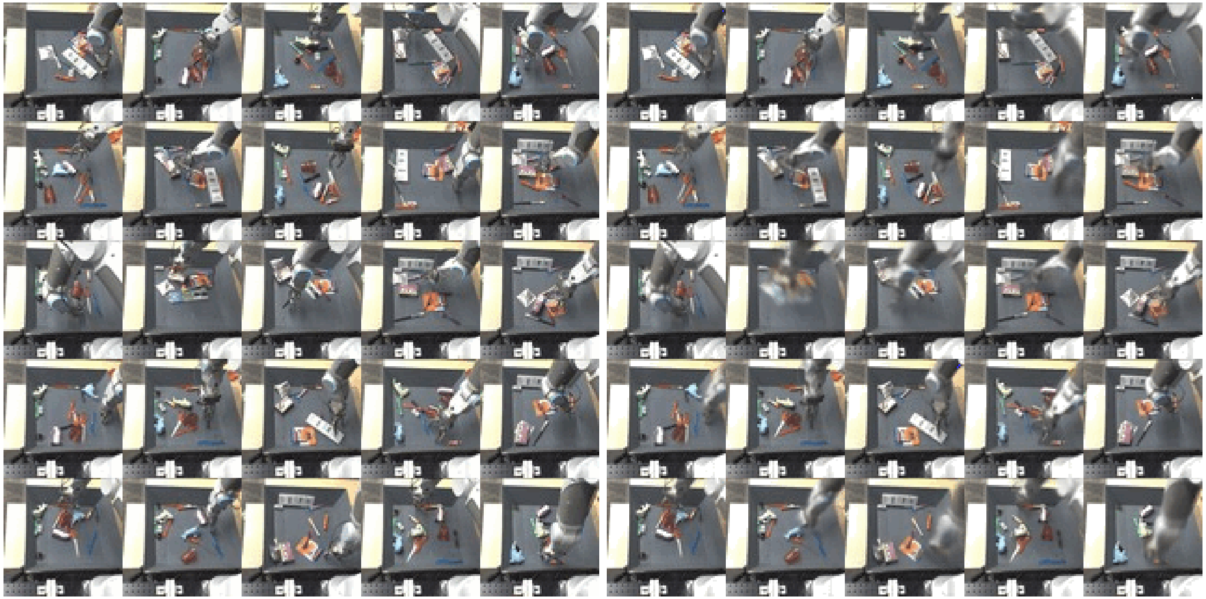
Understanding the factors of variation and invariances.



(Higgins et al, 2017)

Future simulation

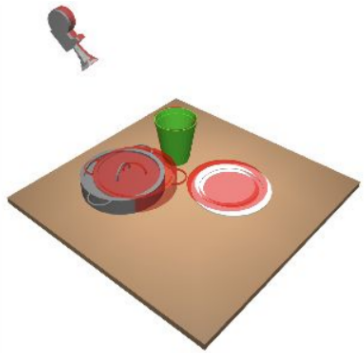
Simulate future trajectories of environments based on actions for planning.

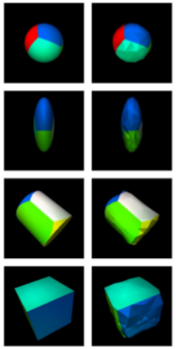


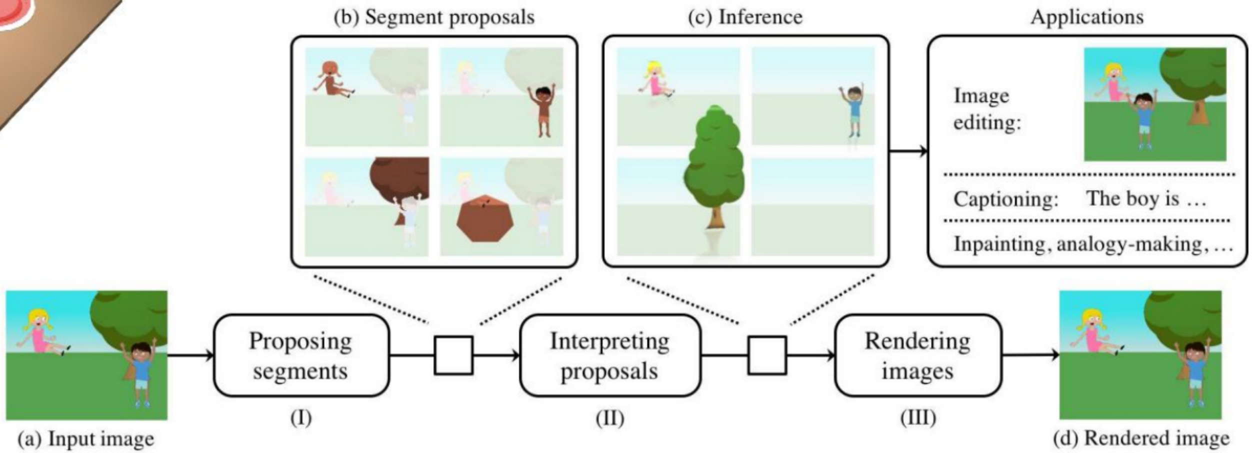
(Finn et al, 2016)

Scene understanding

Understanding the components of scenes and their interactions.







(b) Segment proposals

(c) Inference

Applications

Image editing:

Captioning: The boy is …

Inpainting, analogy-making, …

Proposing segments

(I)

Interpreting proposals

(II)

Rendering images

(III)

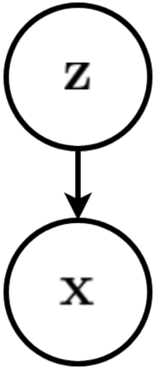
(a) Input image

(d) Rendered image

(Wu et al, 2017)

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 $\mathbf{z} \in \mathcal{Z}$.

This model is given and motivated by domain knowledge assumptions.

Examples:

- Linear discriminant analysis (see previous lecture)
- Bayesian networks
- Hidden Markov models
- Probabilistic programs

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

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

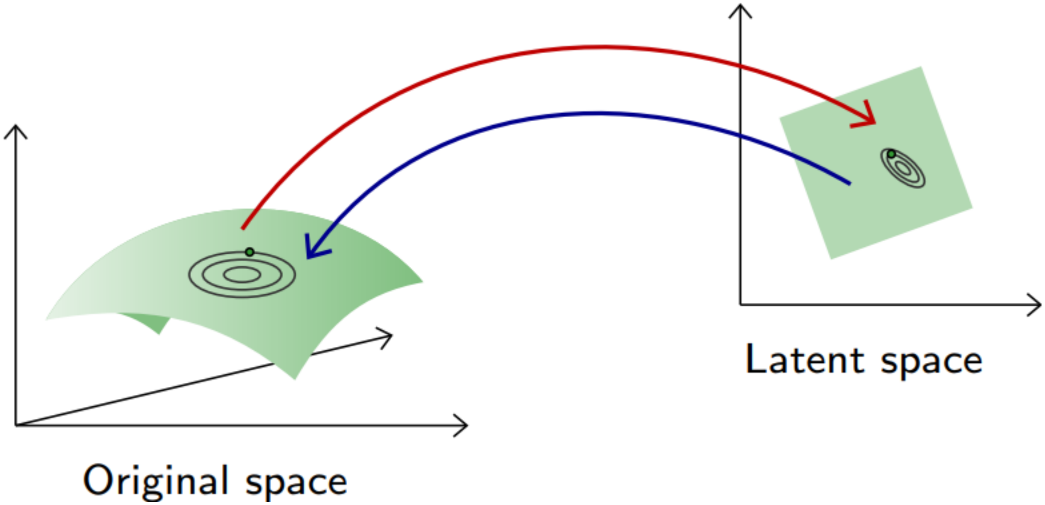
If we interpret $\mathbf{z}$ as causal factors for the high-dimension representations $\mathbf{x}$, then sampling from $p(\mathbf{x}|\mathbf{z})$ can be interpreted as a stochastic generating process from $\mathcal{Z}$ to $\mathcal{X}$.

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

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



Credits: Francois Fleuret, EE559 Deep Learning, EPFL, 2018.

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 ν index the family of distributions.

The parameters ν are fit to minimize the KL divergence between $p(\mathbf{z}|\mathbf{x})$ and the approximation $q(\mathbf{z}|\mathbf{x};\nu)$:

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

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

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

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

Since $\log p(\mathbf{x})$ does not depend on ν, it can be considered as a constant, and minimizing the KL divergence is equivalent to maximizing the evidence lower bound, while being computationally tractable.

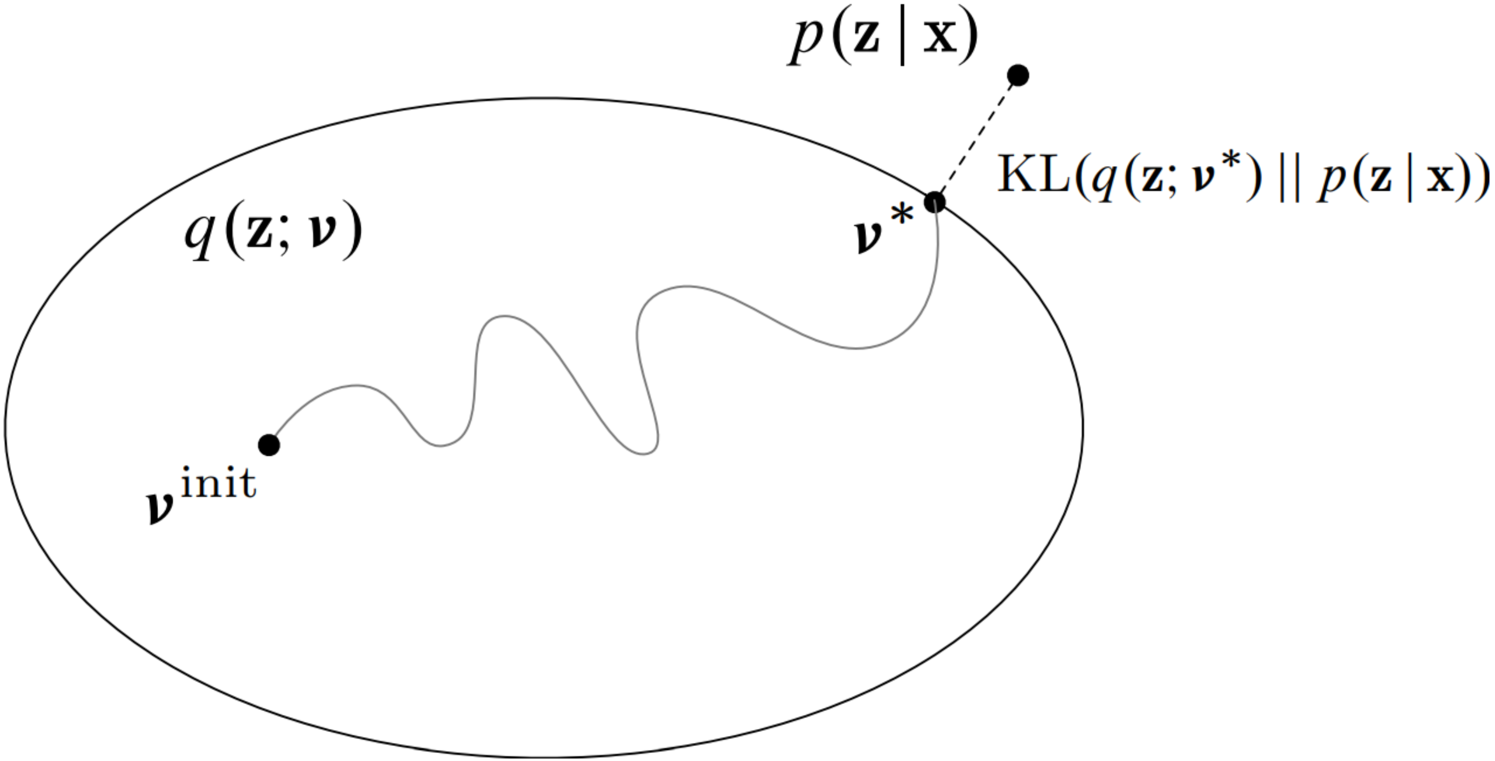
Finally, given a dataset $\mathbf{d} = \{\mathbf{x}_i | i = 1, ..., N\}$, the final objective is the sum $\sum_{\{\mathbf{x}_i \in \mathbf{d}\}} \text{ELBO}(\mathbf{x}_i;\nu)$.

Remark that

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

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).


$p(\mathbf{z} \mid \mathbf{x})$
$\mathrm{KL}(q(\mathbf{z}; \boldsymbol{\nu}^*) \mid\mid p(\mathbf{z} \mid \mathbf{x}))$
$q(\mathbf{z}; \boldsymbol{\nu})$
$\boldsymbol{\nu}^*$
$\boldsymbol{\nu}^{\text{init}}$

Variational inference

How do we optimize the parameters ν? 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 ν, with respect to which we compute the gradient, are of the distribution $q(\mathbf{z}|\mathbf{x};\nu)$ we integrate over.

Solutions:

- Score function estimators:

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

- Elliptical standardization (Kucukelbir et al, 2016).

Variational auto-encoders

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.

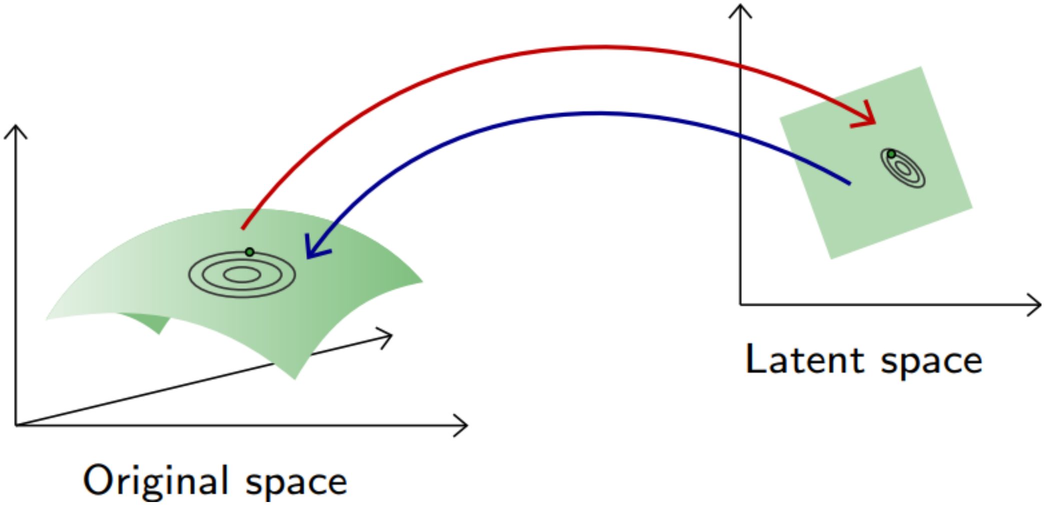
A variational auto-encoder is a deep latent variable model where:

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

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



Credits: Francois Fleuret, EE559 Deep Learning, EPFL, 2018.

As before, we can use variational inference, but to jointly optimize the generative and the inference networks parameters θ and φ.

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

Unbiased gradients of the ELBO with respect to the generative model parameters θ 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.

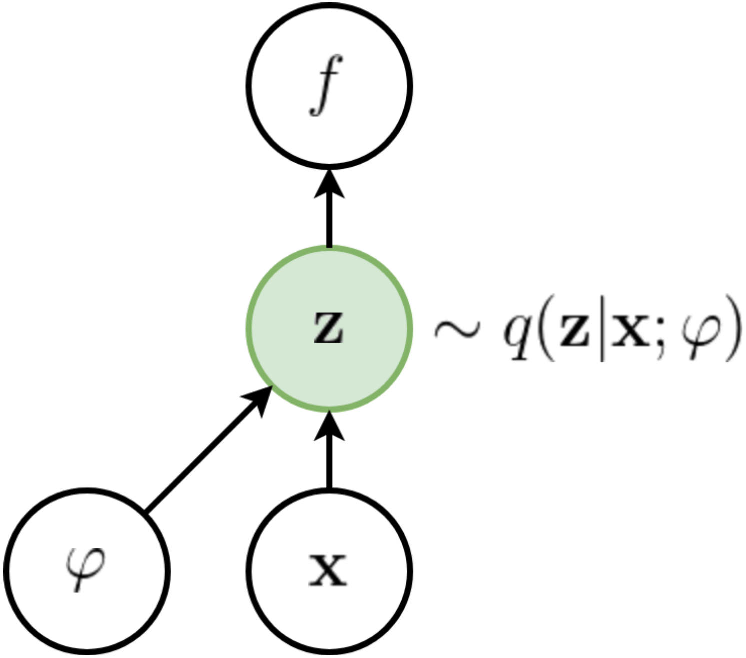
However, gradients with respect to the inference model parameters φ are more difficult to obtain:

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

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

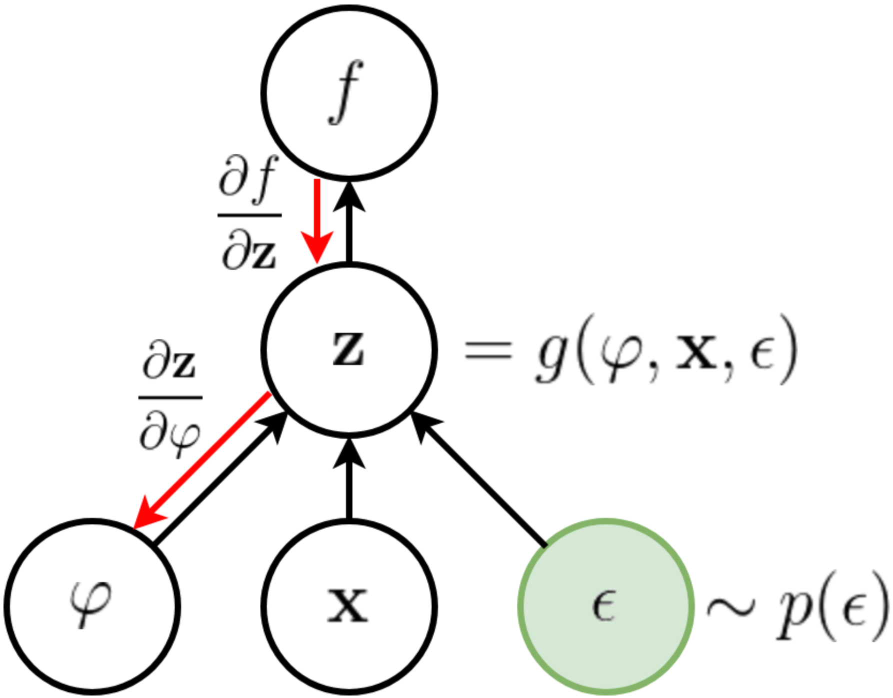
The reparameterization trick consists in re-expressing the variable $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x};\varphi)$ as some differentiable and invertible transformation of another random variable ϵ, given $\mathbf{x}$ and φ,

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

and where the distribution of ϵ is independent of $\mathbf{x}$ or φ.

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_φ, then a common reparameterization is:

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

f
$\frac{\partial f}{\partial \mathbf{z}}$
$\mathbf{z}$
$= g(\varphi, \mathbf{x}, \epsilon)$
$\frac{\partial \mathbf{z}}{\partial \varphi}$
φ
$\mathbf{x}$
ϵ
$\sim p(\epsilon)$

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

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

Therefore,

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

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:

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

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:

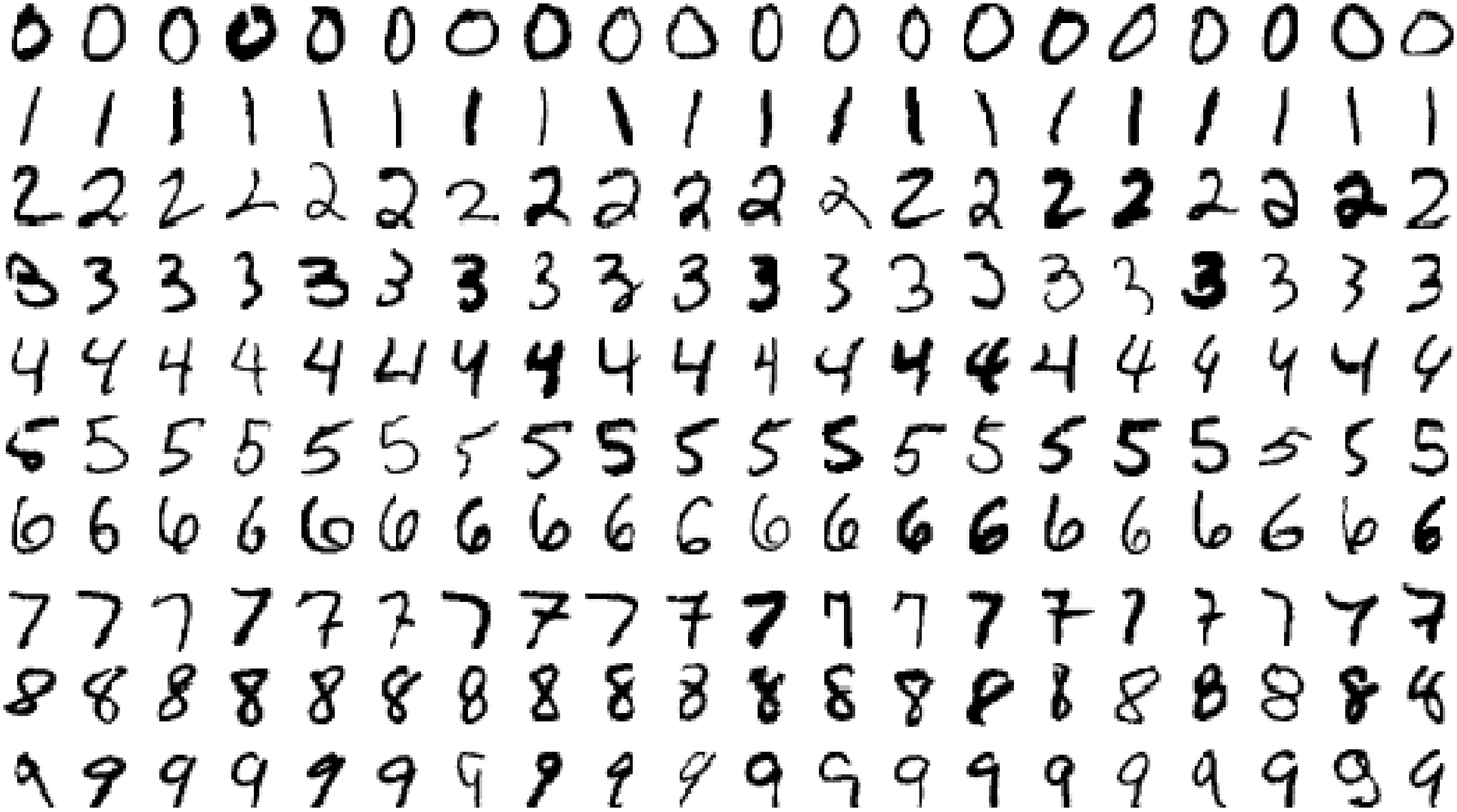
$$\begin{aligned}\text{ELBO}(\mathbf{x};\theta,\varphi) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[\log p(\mathbf{x},\mathbf{z};\theta) - \log q(\mathbf{z}|\mathbf{x};\varphi)\right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[\log p(\mathbf{x}|\mathbf{z};\theta)\right] - KL(q(\mathbf{z}|\mathbf{x};\varphi)||p(\mathbf{z}))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

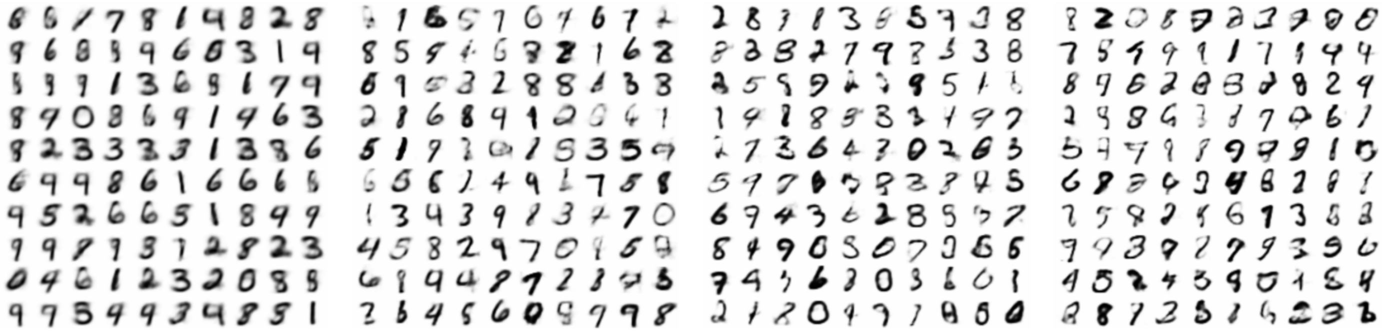
where the KL divergence can be expressed analytically as

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

which allows to evaluate its derivative without approximation.

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



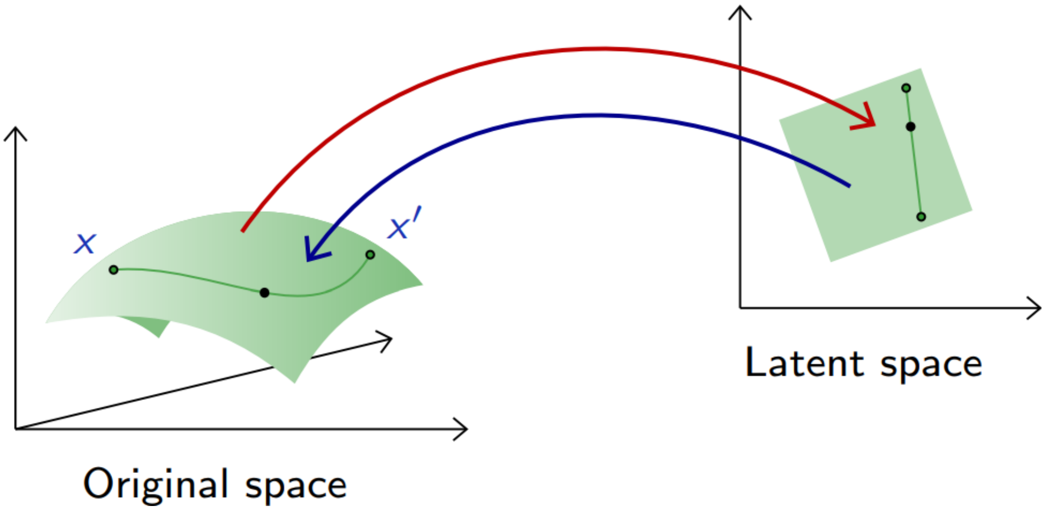


(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

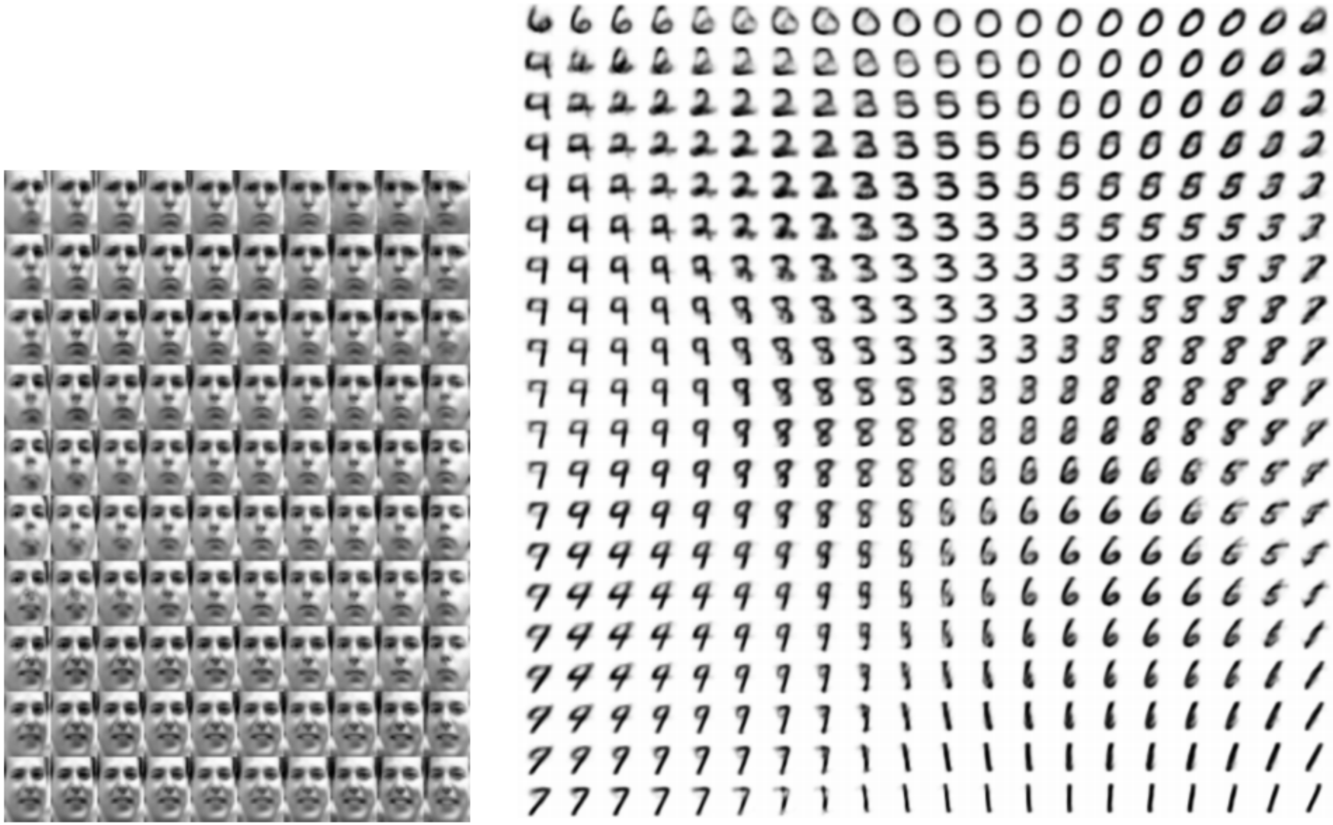
Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

(Kingma and Welling, 2013)

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.



Credits: Francois Fleuret, EE559 Deep Learning, EPFL, 2018.



(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)

Some further examples

Random walks in latent space.

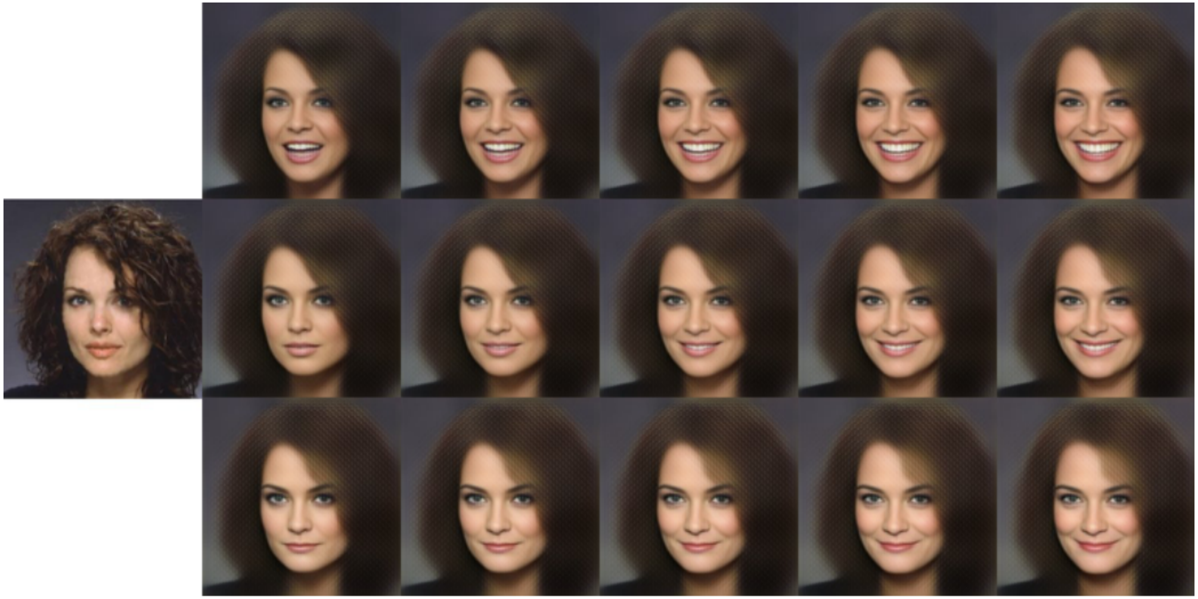
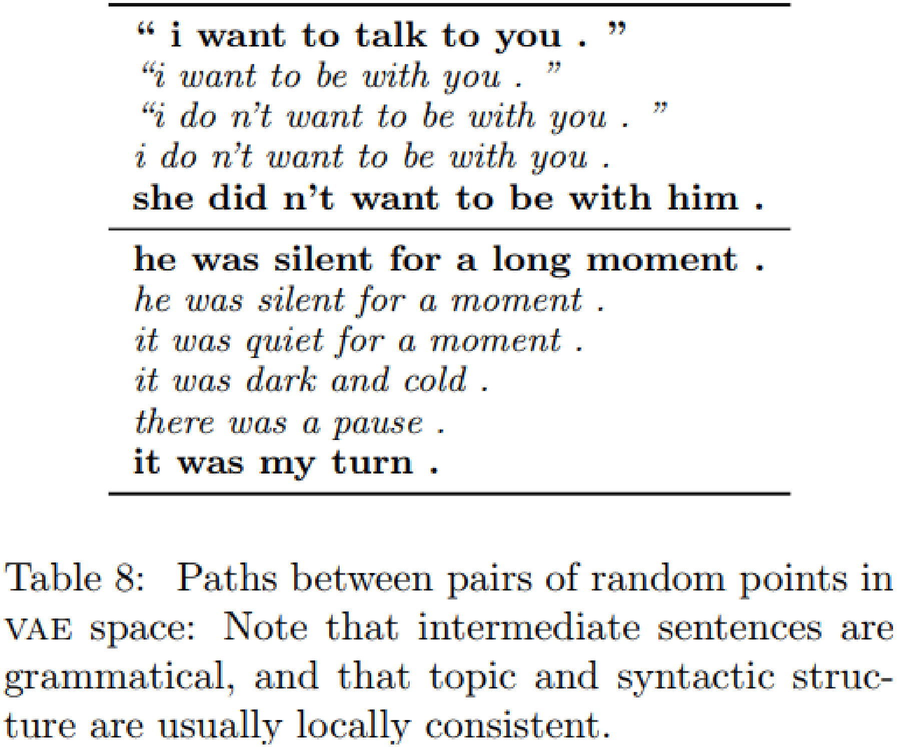


Figure 7: Decoupling attribute vectors for smiling (x-axis) and mouth open (y-axis) allows for more flexible latent space transformations. Input shown at left with reconstruction adjacent. (model: VAE from Lamb 16 on CelebA)

(White, 2016)

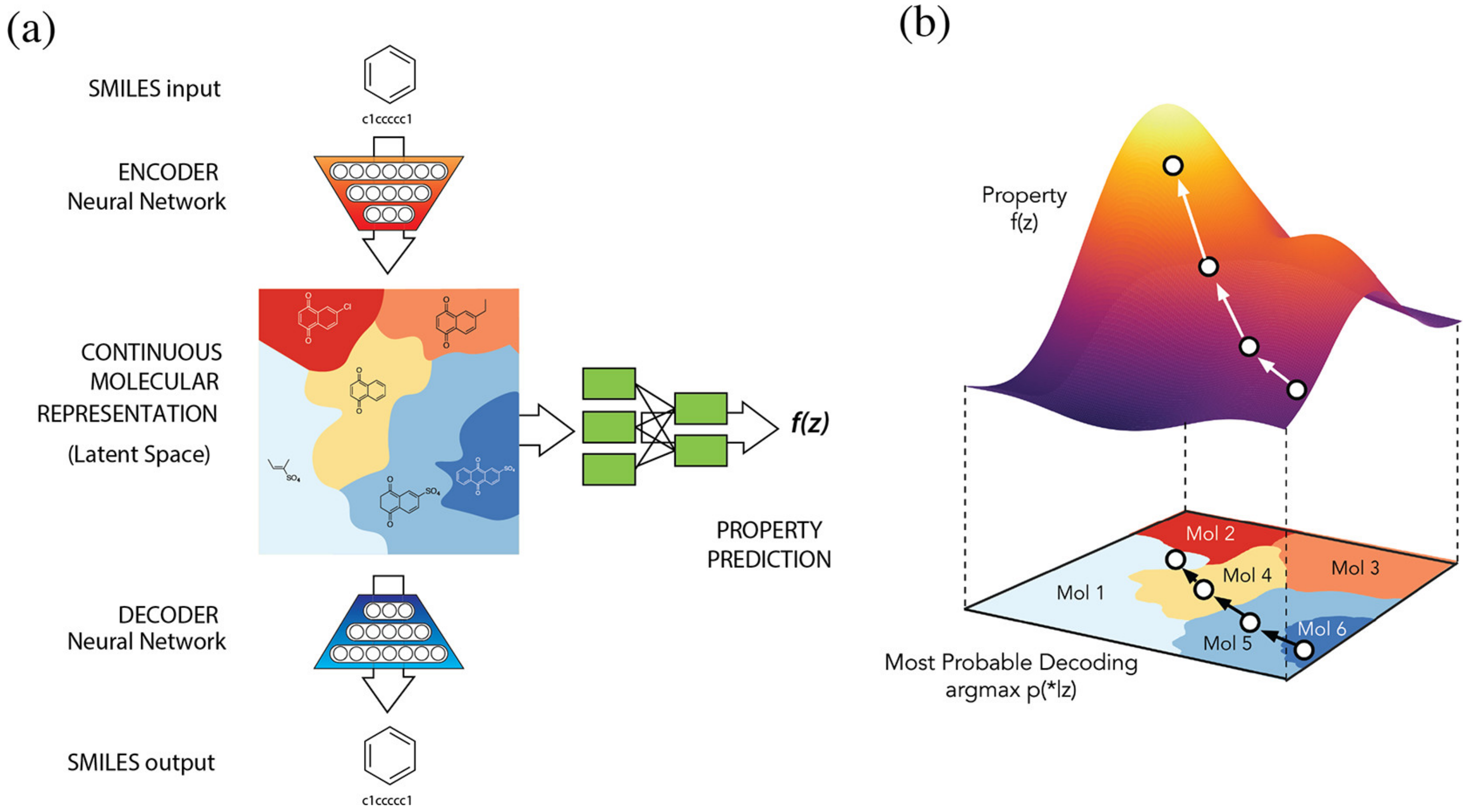


“ i want to talk to you . ”
“i want to be with you . ”
“i do n’t want to be with you . ”
i do n’t want to be with you .
she did n’t want to be with him .
he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn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

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)

Impersonation by encoding-decoding an unknown face.



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

The end.

References

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