Deep Learning

Lecture 11: Auto-encoders and variational auto-encoders

Prof. Gilles Louppe
g.louppe@uliege.be

Today

Learn a model of the data.

- Auto-encoders
- Variational inference
- Variational auto-encoders

"The brain has about 10^{14} synapses and we only live for about 10^{9} seconds. So we have a lot more parameters than data. This motivates the idea that we must do a lot of unsupervised learning since the perceptual input (including proprioception) is the only place we can get 10^{5} dimensions of constraint per second."

Geoffrey Hinton, 2014.

- "Pure" Reinforcement Learning (cherry)
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- Supervised Learning (icing)
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- Unsupervised/Predictive Learning (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**
    - **(Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)**

"We need tremendous amount of information to build machines that have common sense and generalize."

Yann LeCun, 2016.

Deep unsupervised learning

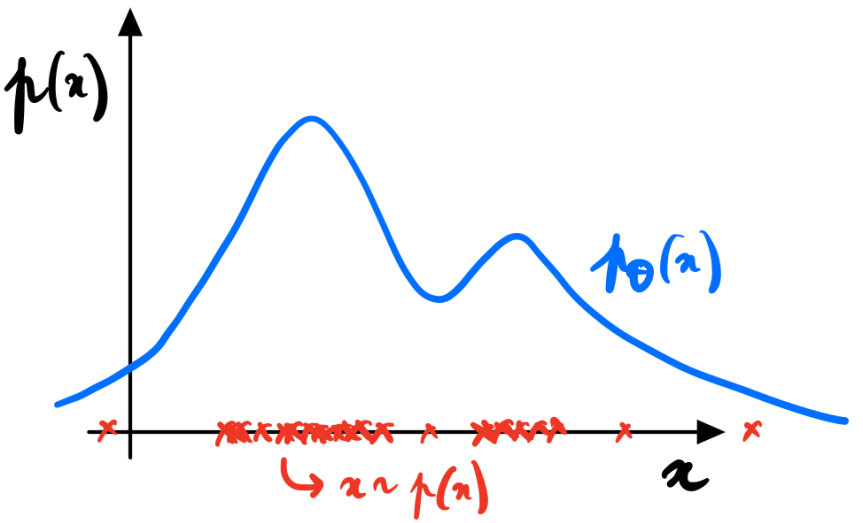
Deep unsupervised learning is about learning a model of the data, explicitly or implicitly, without requiring labels.

- Generative models: recreate the raw data distribution (e.g., the distribution of natural images).
- Self-supervised learning: solve puzzle tasks that require semantic understanding (e.g., predict a missing word in a sequence).

Generative models

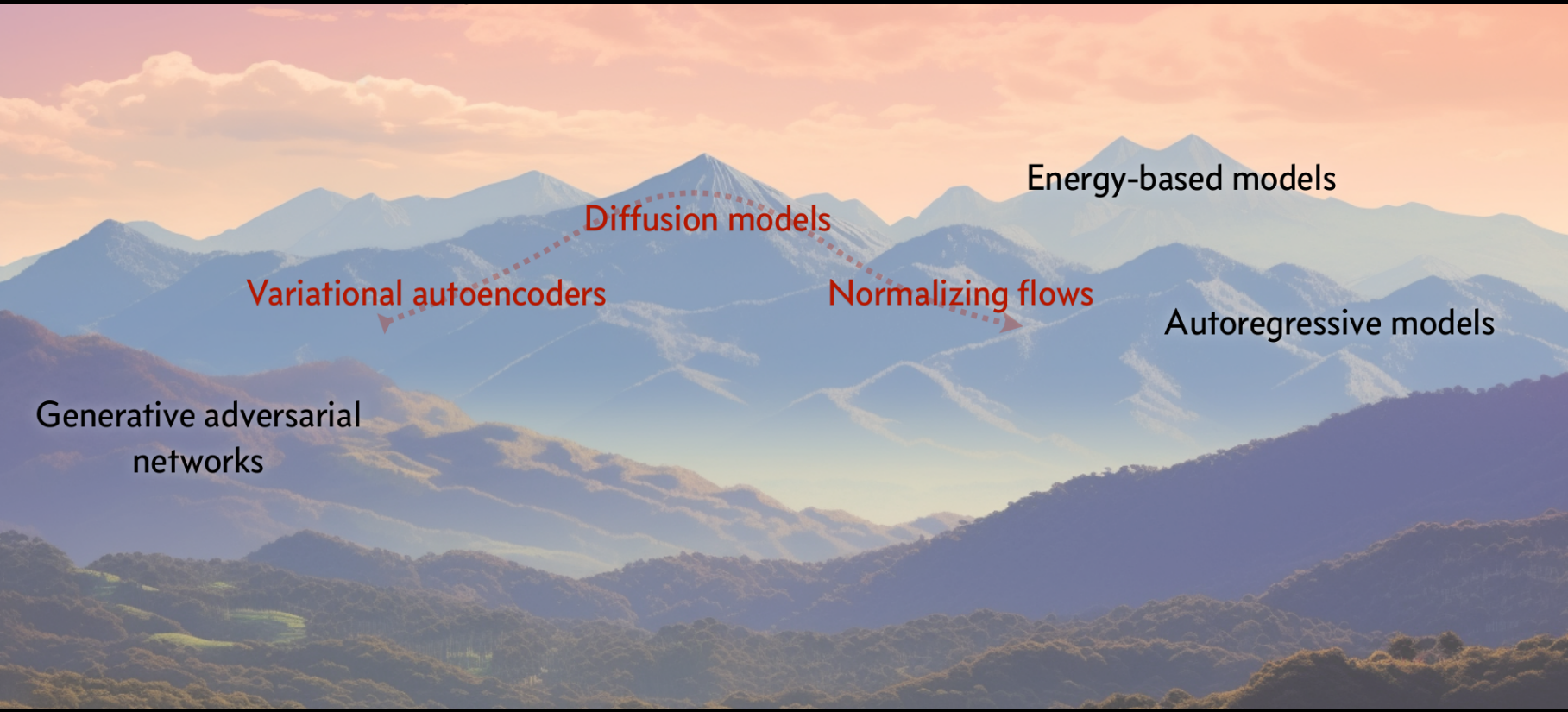
A (deep) generative model is a probabilistic model p_θ that can be used as a simulator of the data.

Formally, a generative model defines a probability distribution $p_\theta(\mathbf{x})$ over the data $\mathbf{x} \in \mathcal{X}$, where the parameters θ are learned to match the (unknown) data distribution $p(\mathbf{x})$.



Variational auto-encoders
(Kingma and Welling, 2013)

Diffusion models
(Midjourney, 2023)



—

Credits: Karsten et al, 2022; Siddharth Mishra-Sharma, 2023.

What can we do with generative models?

Produce samples

$$\mathbf{x} \sim p(\mathbf{x}|\theta)$$

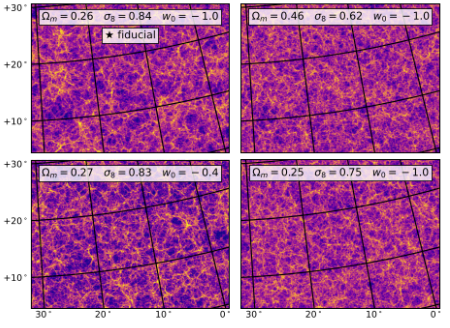
Evaluate densities

$$p(\mathbf{x}|\theta)$$

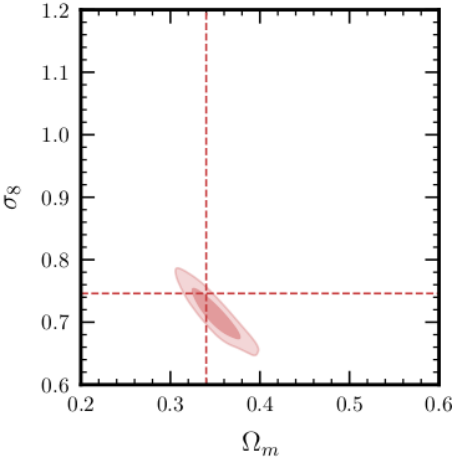
Encode complex priors

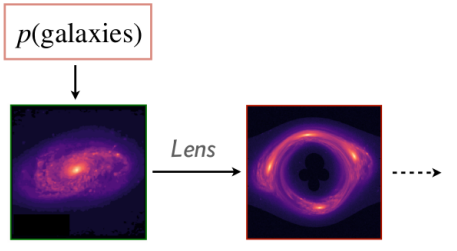
$$p(\mathbf{x})$$

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

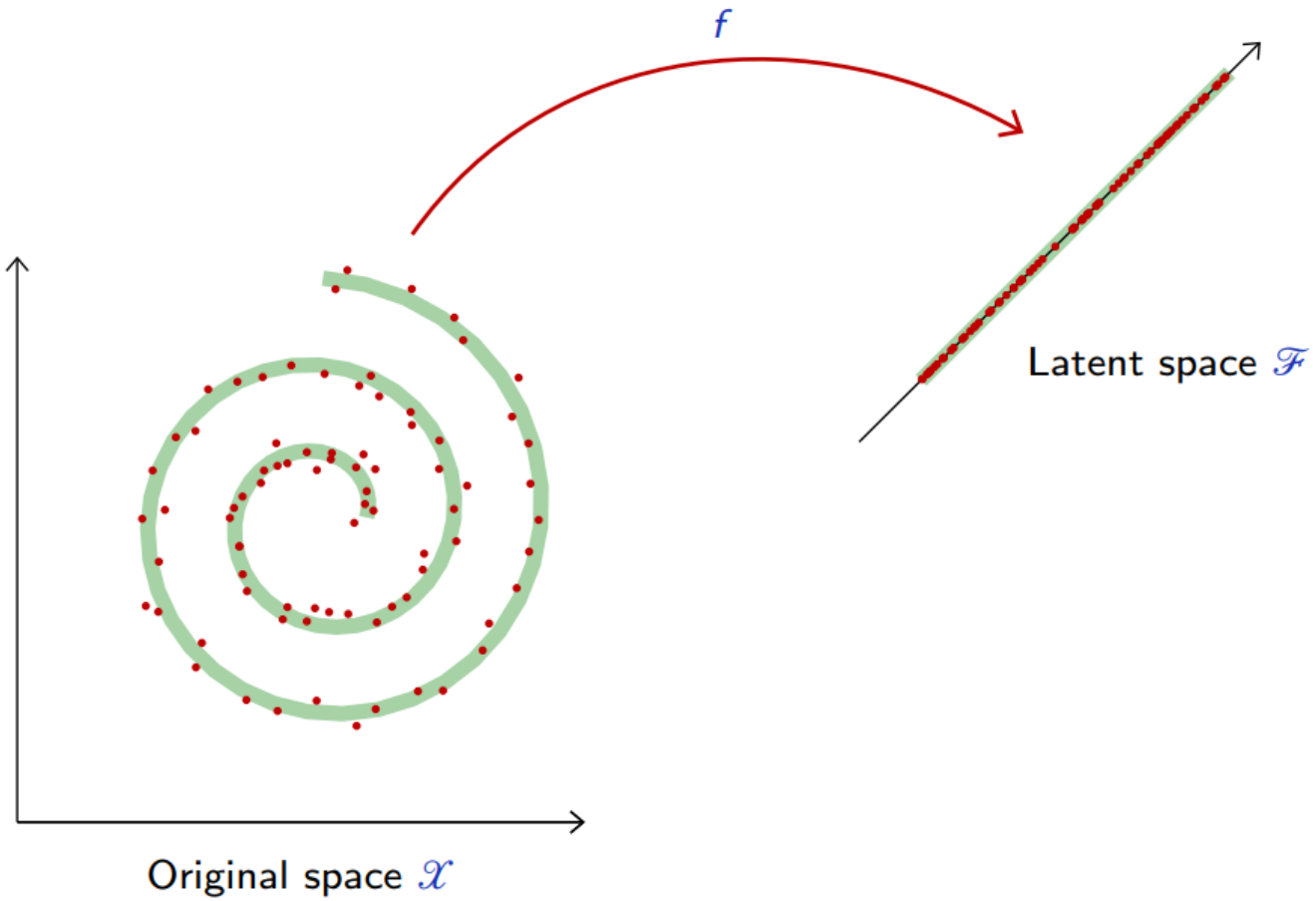


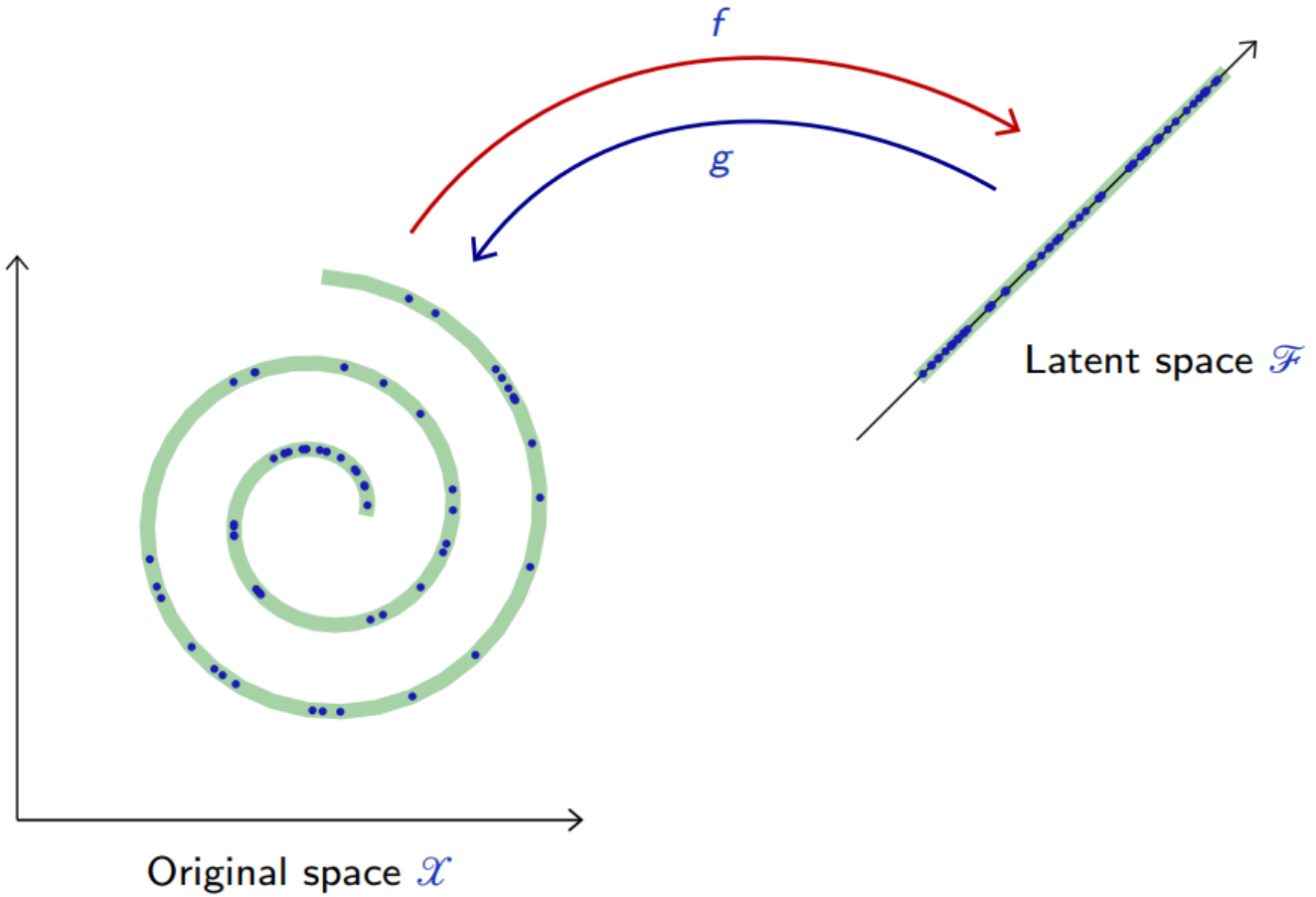
[Kacprzak et al 2022]





Auto-encoders


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

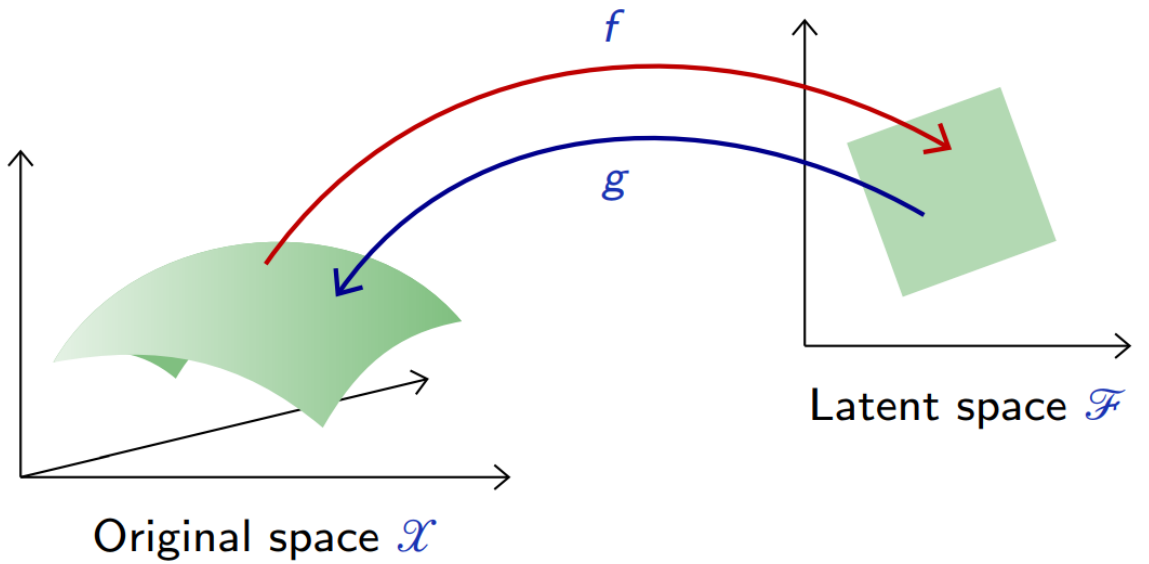

f
g
Latent space $\mathcal{F}$
Original space $\mathcal{X}$

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.



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 $g(\cdot; \theta_g)$, training consists of minimizing an empirical estimate of that loss,

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

For example, when the auto-encoder is linear,

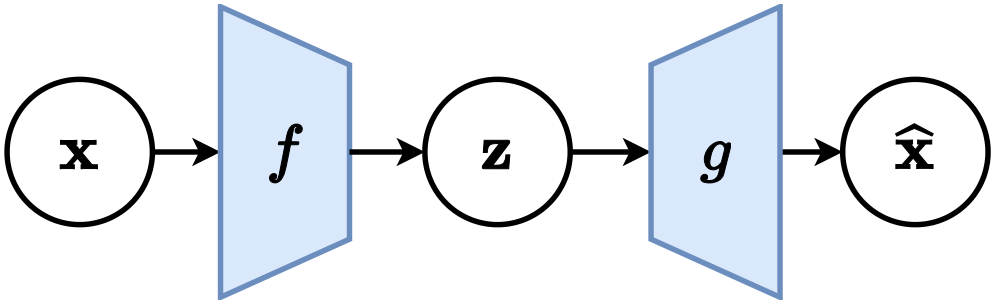
$$f : \mathbf{z} = \mathbf{U}^T \mathbf{x}$$
$$g : \hat{\mathbf{x}} = \mathbf{U}\mathbf{z},$$

with $\mathbf{U} \in \mathbb{R}^{p \times d}$, 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

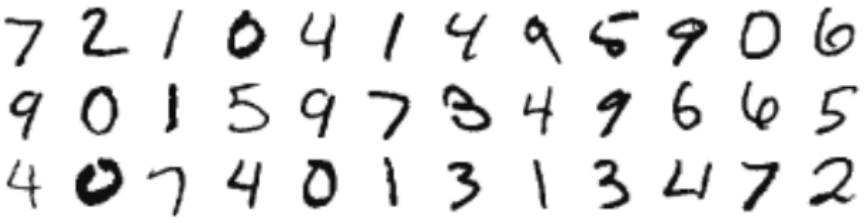


Better results can be achieved with more sophisticated classes of mappings than linear projections: use deep neural networks for f and g.

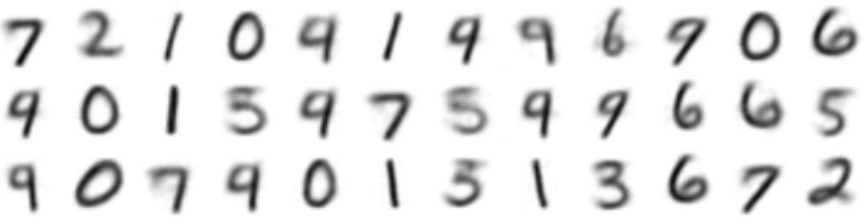
For instance,

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

X (original samples)

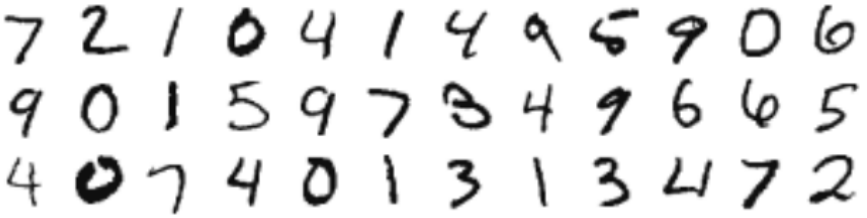


$g \circ f(X)$ (CNN, $d = 2$)

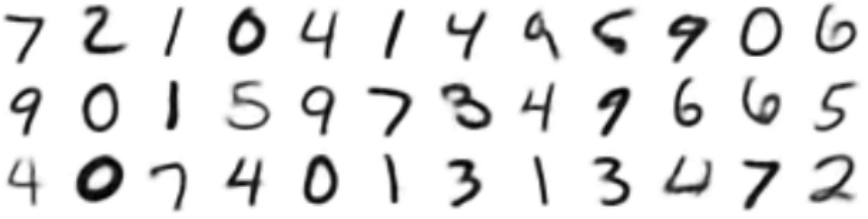


$g \circ f(X)$ (PCA, $d = 2$)

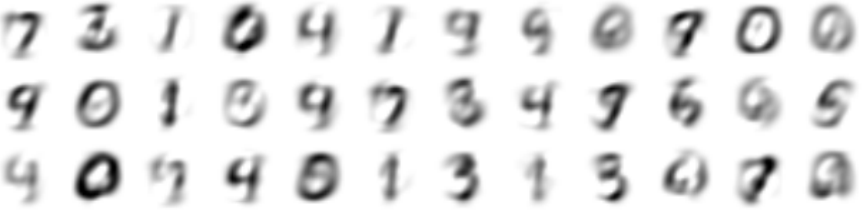
X (original samples)



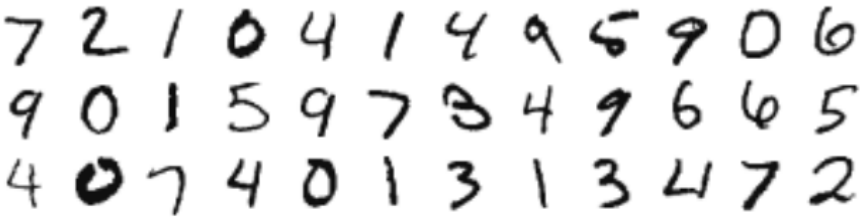
$g \circ f(X)$ (CNN, $d = 8$)



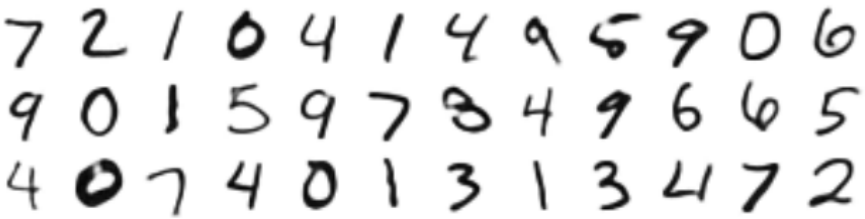
$g \circ f(X)$ (PCA, $d = 8$)



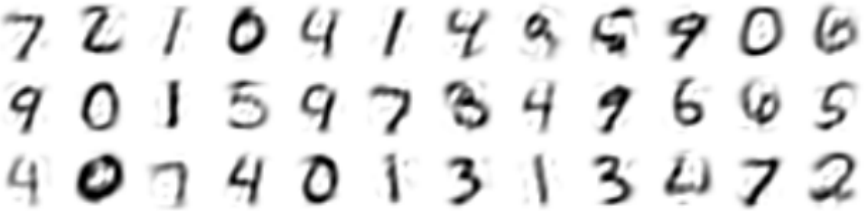
X (original samples)



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

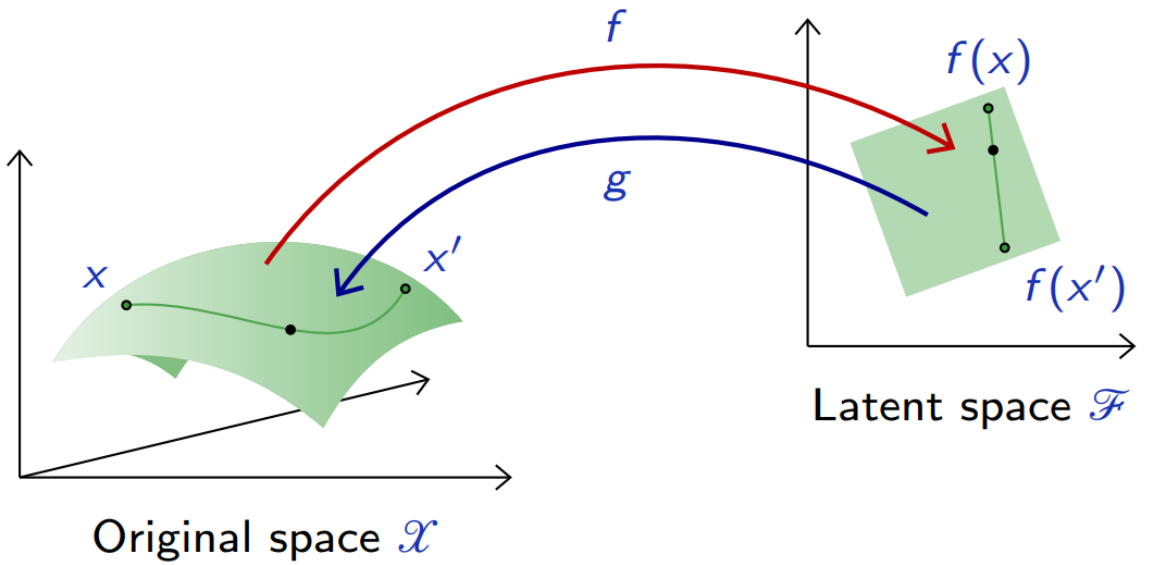


$g \circ f(X)$ (PCA, $d = 32$)

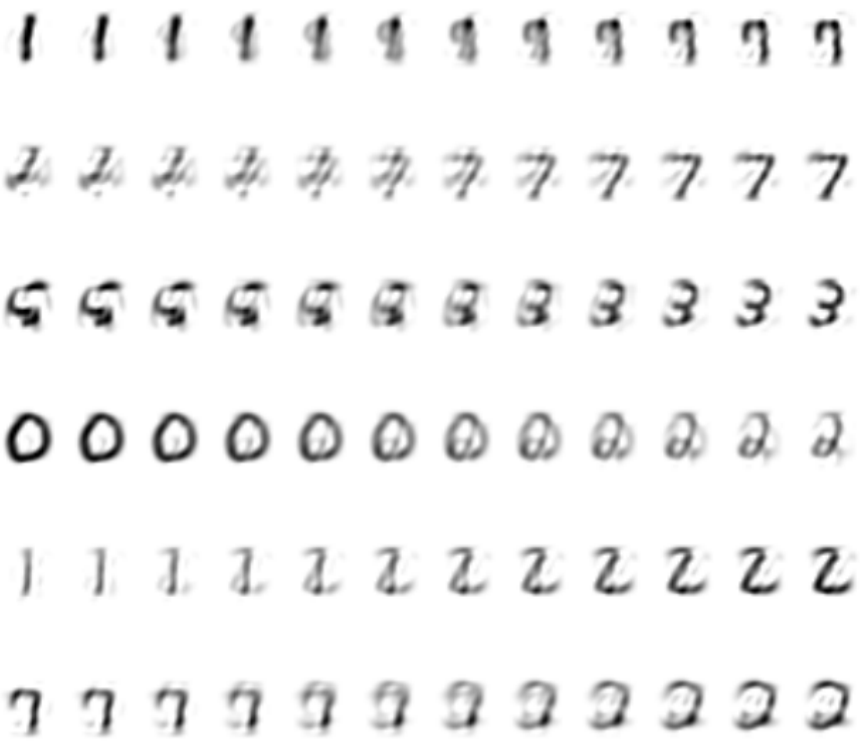


Interpolation

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 ($d = 32$)

Denoising auto-encoders

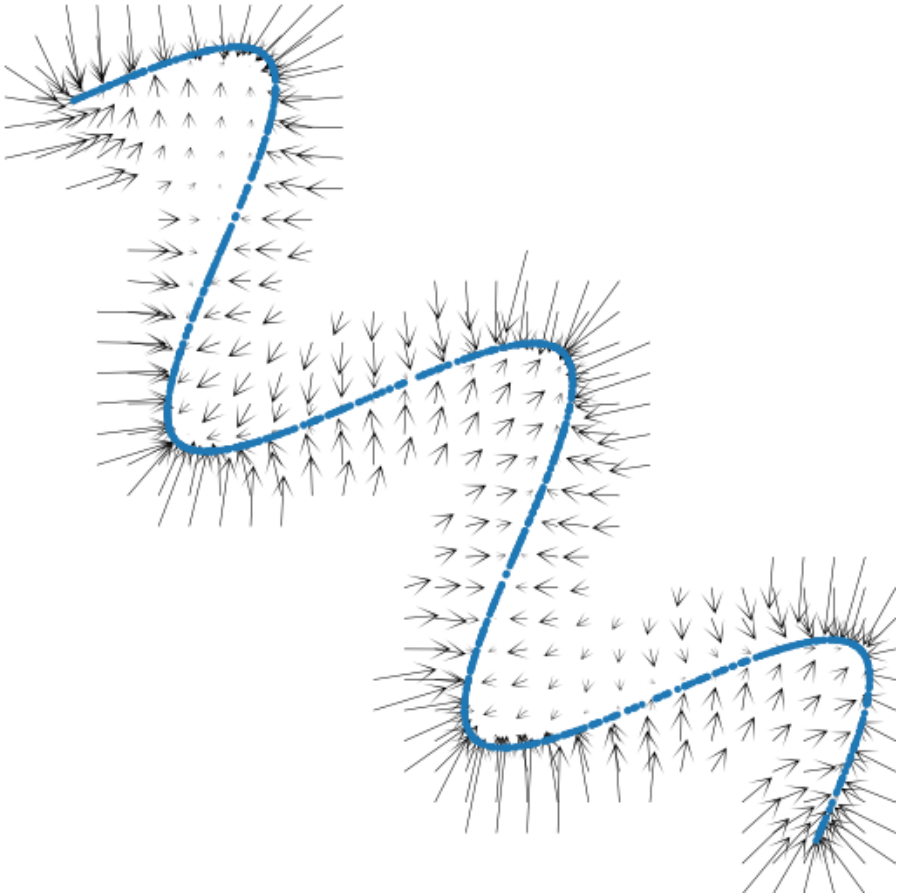
Besides dimension reduction, auto-encoders can capture dependencies between signal components to restore degraded or noisy signals. In this case, the composition

$$h = g \circ f : \mathcal{X} \to \mathcal{X}$$

is a denoising auto-encoder.

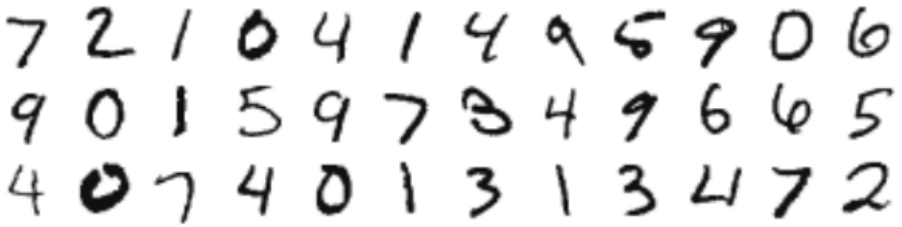
The goal is to optimize h such that a perturbation $\tilde{\mathbf{x}}$ of the signal $\mathbf{x}$ is restored to $\mathbf{x}$, hence

$$h(\tilde{\mathbf{x}}) \approx \mathbf{x}.$$

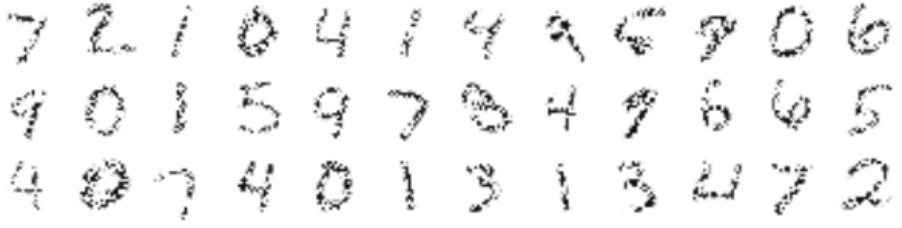


—

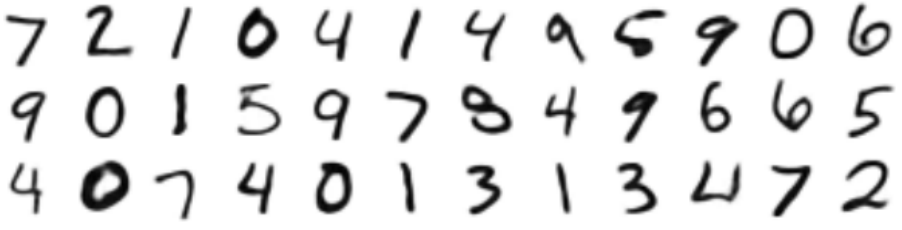
Original



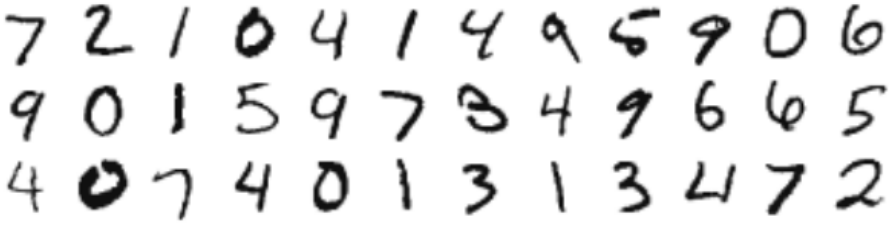
Corrupted ($p = 0.5$)



Reconstructed

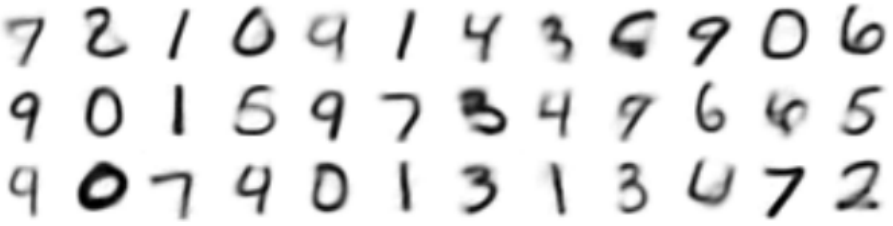


Original

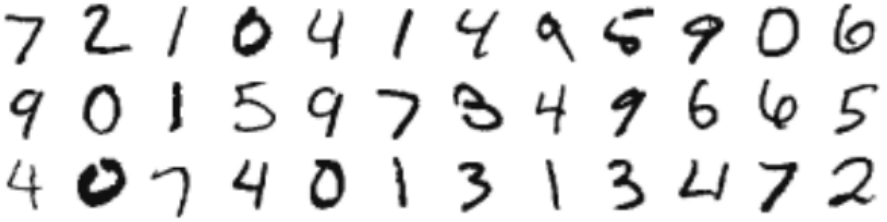


Corrupted ($p = 0.9$)

Reconstructed

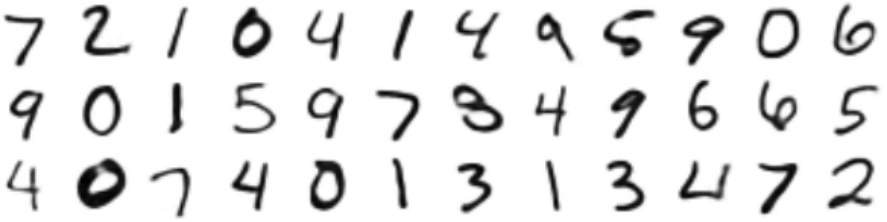


Original



Corrupted ($\sigma = 4$)

Reconstructed

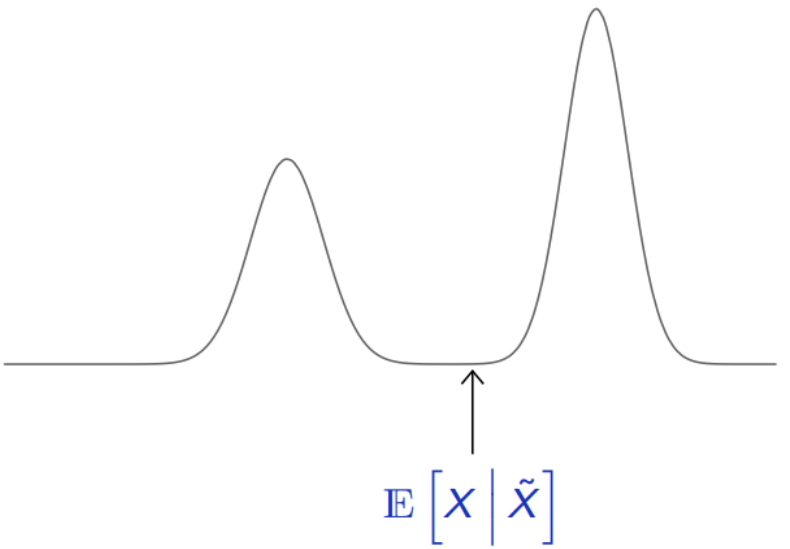


A fundamental weakness of denoising auto-encoders is that the posterior $p(\mathbf{x}|\tilde{\mathbf{x}})$ is possibly multi-modal.

If we train an auto-encoder with the quadratic loss (i.e., implicitly assuming a Gaussian likelihood), then the best reconstruction is

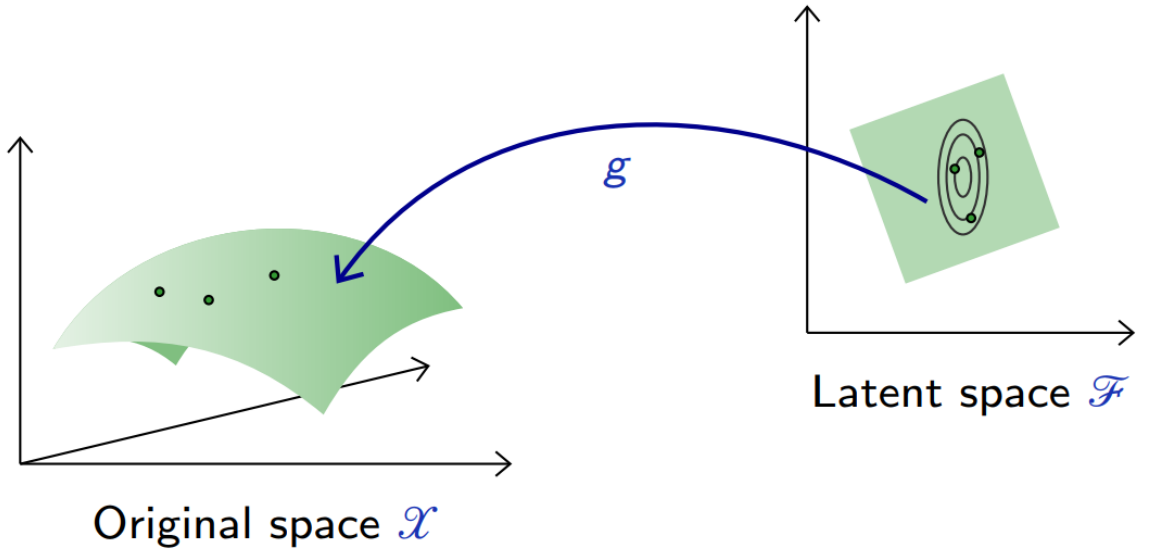
$$h(\tilde{\mathbf{x}}) = \mathbb{E}[\mathbf{x}|\tilde{\mathbf{x}}],$$

which may be very unlikely under $p(\mathbf{x}|\tilde{\mathbf{x}})$.



Sampling from an AE's latent space

The generative capability of the decoder g in an auto-encoder 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.

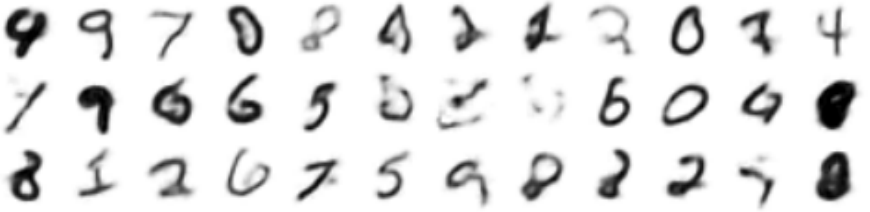


For instance, a factored Gaussian model with diagonal covariance matrix,

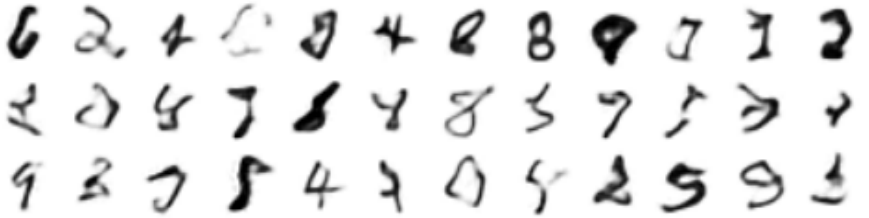
$$q(\mathbf{z}) = \mathcal{N}(\hat{\mu}, \hat{\Sigma}),$$

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

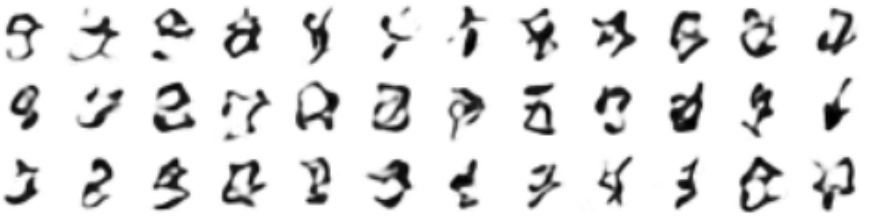
Autoencoder sampling ($d = 8$)



Autoencoder sampling ($d = 16$)



Autoencoder sampling ($d = 32$)

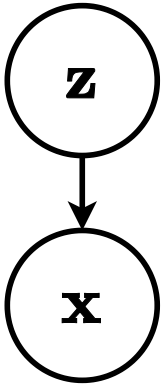


These results are not satisfactory because the density model on the latent space is too simple and inadequate.

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

Variational inference

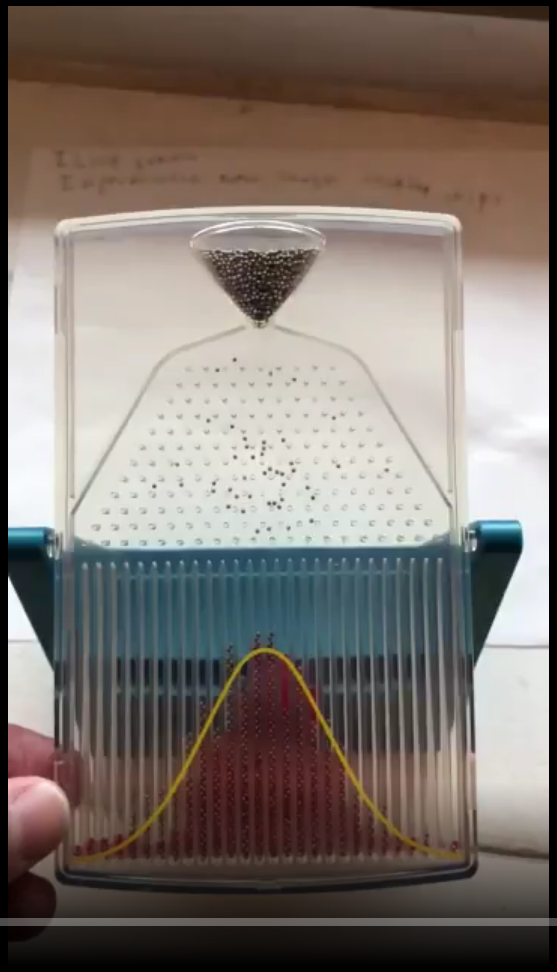
Latent variable model



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 defines a joint probability distribution $p_\theta(\mathbf{x}, \mathbf{z})$, which decomposes as

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



0:00 / 0:45

How to fit a latent variable model?

$$
\begin{aligned}
\theta^* &= \arg\max_\theta p_\theta(\mathbf{x}) \\
&= \arg\max_\theta \int p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \\
&= \arg\max_\theta \mathbb{E}_{p(\mathbf{z})} \left[p_\theta(\mathbf{x}|\mathbf{z}) \right] d\mathbf{z} \\
&\approx \arg\max_\theta \frac{1}{N} \sum_{i=1}^{N} p_\theta(\mathbf{x}|\mathbf{z}_i)
\end{aligned}
$$

How to fit a latent variable model?

$$
\begin{aligned}
\theta^* &= \arg\max_\theta p_\theta(\mathbf{x}) \\
&= \arg\max_\theta \int p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \\
&= \arg\max_\theta \mathbb{E}_{p(\mathbf{z})} \left[p_\theta(\mathbf{x}|\mathbf{z}) \right] d\mathbf{z} \\
&\approx \arg\max_\theta \frac{1}{N} \sum_{i=1}^{N} p_\theta(\mathbf{x}|\mathbf{z}_i)
\end{aligned}
$$

The curse of dimensionality will lead to poor estimates of the expectation.

Variational inference

Let us instead consider a variational approach to fit the model parameters θ.

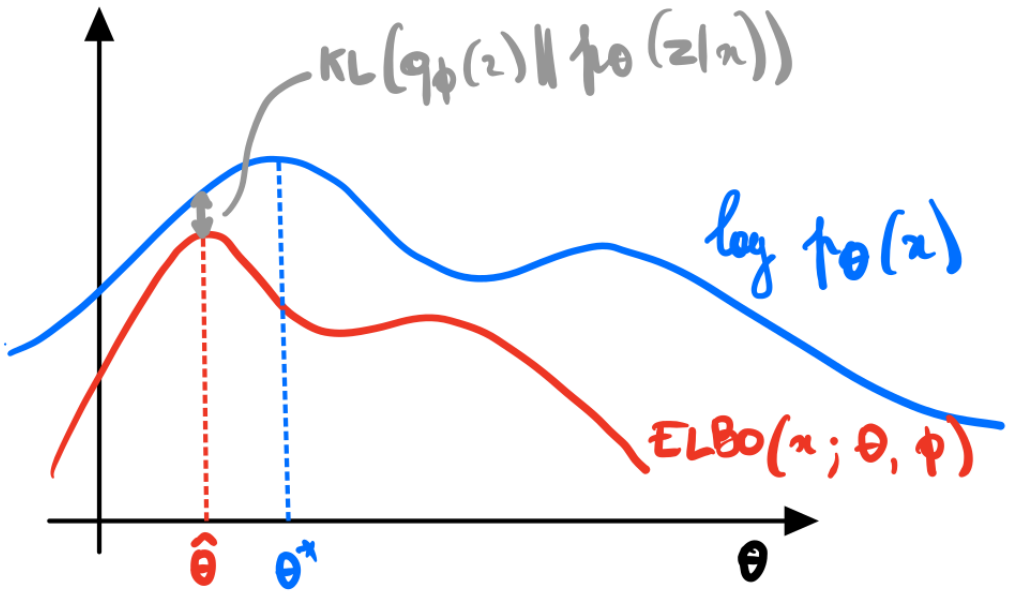
Using a variational distribution $q_\phi(\mathbf{z})$ over the latent variables $\mathbf{z}$, we have

$$\begin{aligned}
\log p_\theta(\mathbf{x}) &= \log \mathbb{E}_{p(\mathbf{z})} \left[p_\theta(\mathbf{x}|\mathbf{z}) \right] \\
&= \log \mathbb{E}_{q_\phi(\mathbf{z})} \left[\frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q_\phi(\mathbf{z})} \right] \\
&\geq \mathbb{E}_{q_\phi(\mathbf{z})} \left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q_\phi(\mathbf{z})} \right] \quad (\mathrm{ELBO}(\mathbf{x}; \theta, \phi)) \\
&= \mathbb{E}_{q_\phi(\mathbf{z})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) \right] - \mathrm{KL}(q_\phi(\mathbf{z}) || p(\mathbf{z}))
\end{aligned}$$

Using the Bayes rule, we can also write

$$\begin{aligned}
\mathrm{ELBO}(\mathbf{x};\theta,\phi) &= \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z})}\frac{p_\theta(\mathbf{x})}{p_\theta(\mathbf{x})}\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z})}\left[\log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z})}p_\theta(\mathbf{x})\right] \\
&= \log p_\theta(\mathbf{x}) - \mathrm{KL}(q_\phi(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x})).
\end{aligned}$$

Therefore, $\log p_\theta(\mathbf{x}) = \mathrm{ELBO}(\mathbf{x};\theta,\phi) + \mathrm{KL}(q_\phi(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x}))$.



Provided the KL gap remains small, the model parameters can now be optimized by maximizing the ELBO,

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \mathrm{ELBO}(\mathbf{x}; \theta, \phi).$$

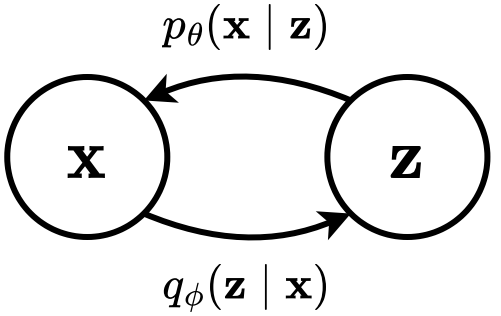
Optimization

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \mathrm{ELBO}(\mathbf{x}; \theta, \phi).$$

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

In general, the gradient of the ELBO is intractable to compute, but we can estimate it with Monte Carlo integration.

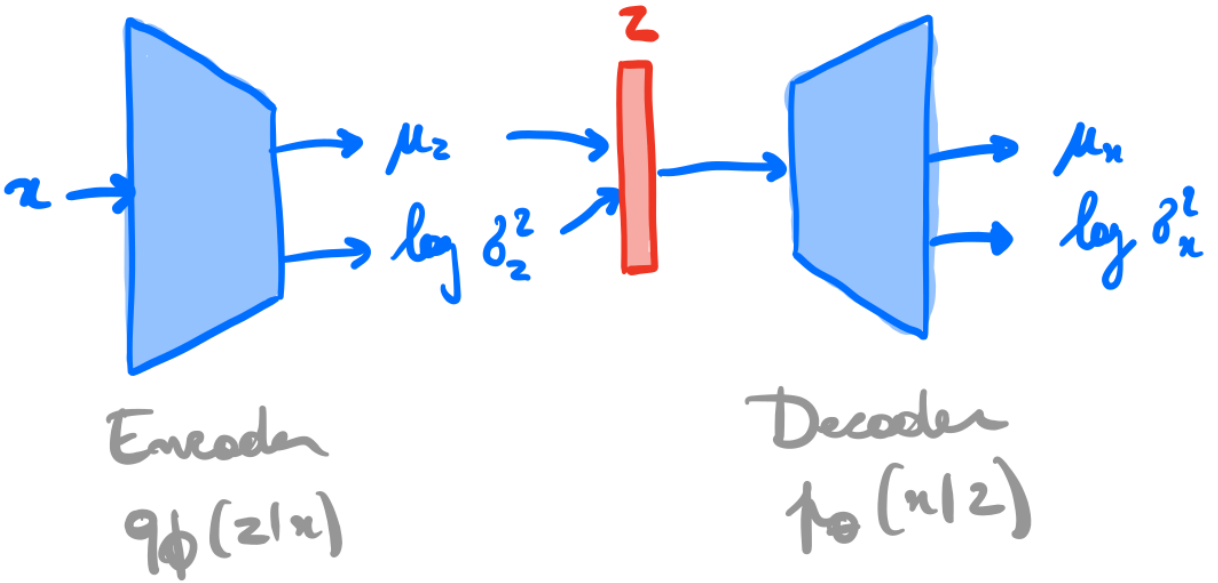
Variational auto-encoders



So far we assumed a prescribed probabilistic model motivated by domain knowledge. We will now directly learn a stochastic generating process $p_\theta(\mathbf{x}|\mathbf{z})$ with a neural network.

We will also amortize the inference process by learning a second neural network $q_\phi(\mathbf{z}|\mathbf{x})$ approximating the posterior, conditionally on the observed data $\mathbf{x}$.

Variational auto-encoders



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

- The prior $p(\mathbf{z})$ is prescribed, and usually chosen to be Gaussian.
- The likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ is parameterized with a generative network NN_θ (or decoder) that takes as input $\mathbf{z}$ and outputs parameters $\varphi = \text{NN}_\theta(\mathbf{z})$ to the data distribution. E.g.,

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

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

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

As before, we can use variational inference to jointly optimize the encoder and decoder networks parameters ϕ and θ, but now in expectation over the data distribution $p(\mathbf{x})$:

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

Interpretation:

- Given some decoder network set at 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 encoder network set at 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, as

$$\begin{aligned}\nabla_\theta \mathrm{ELBO}(\mathbf{x};\theta,\phi) &= \nabla_\theta \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\nabla_\theta(\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}))\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\nabla_\theta \log p_\theta(\mathbf{x},\mathbf{z})\right],\end{aligned}$$

which can be estimated with Monte Carlo integration.

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

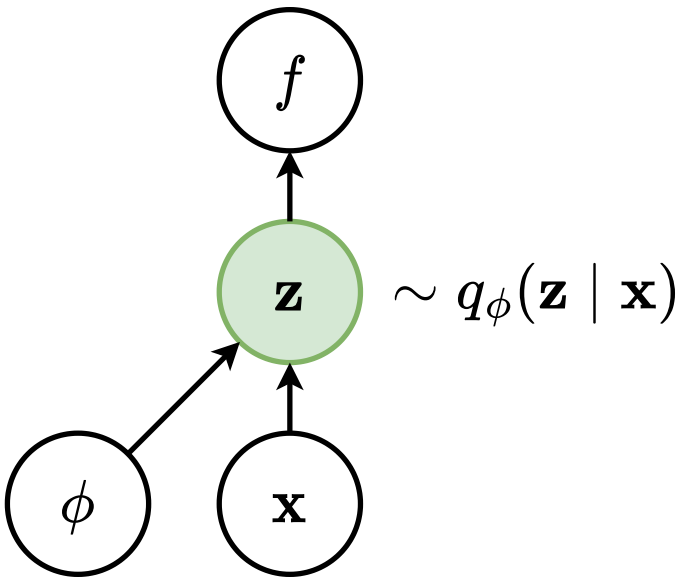
$$\begin{aligned}\nabla_\phi \mathrm{ELBO}(\mathbf{x};\theta,\phi) &= \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right] \\ &\neq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\nabla_\phi(\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}))\right].\end{aligned}$$

Reparameterization trick

Let us abbreviate

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

The computational graph of a Monte Carlo estimate of the ELBO would look like



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

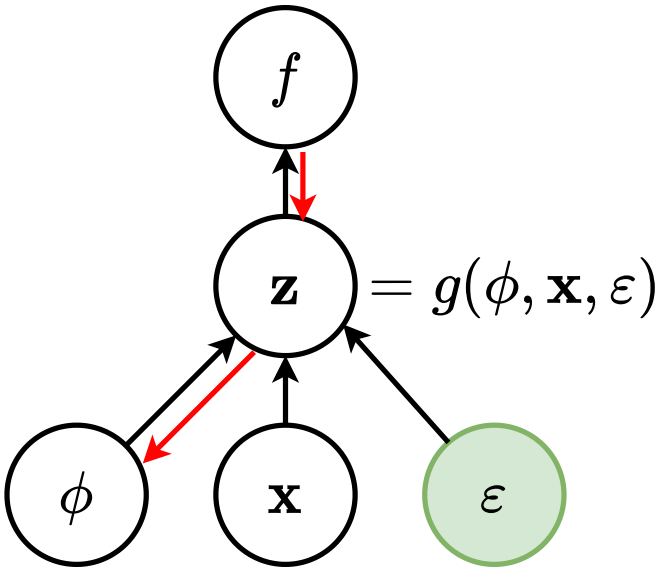
The reparameterization trick consists in re-expressing the variable

$$\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$$

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

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

such that the distribution of ϵ is independent of $\mathbf{x}$ or ϕ.



If $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu(\mathbf{x}; \phi), \sigma^2(\mathbf{x}; \phi))$, where $\mu(\mathbf{x}; \phi)$ and $\sigma^2(\mathbf{x}; \phi)$ are the outputs of the inference network NN_ϕ, then a common reparameterization is

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

Given this change of variable, the ELBO can be rewritten as

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

Therefore estimating the gradient of the ELBO with respect to ϕ is now easy, as

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

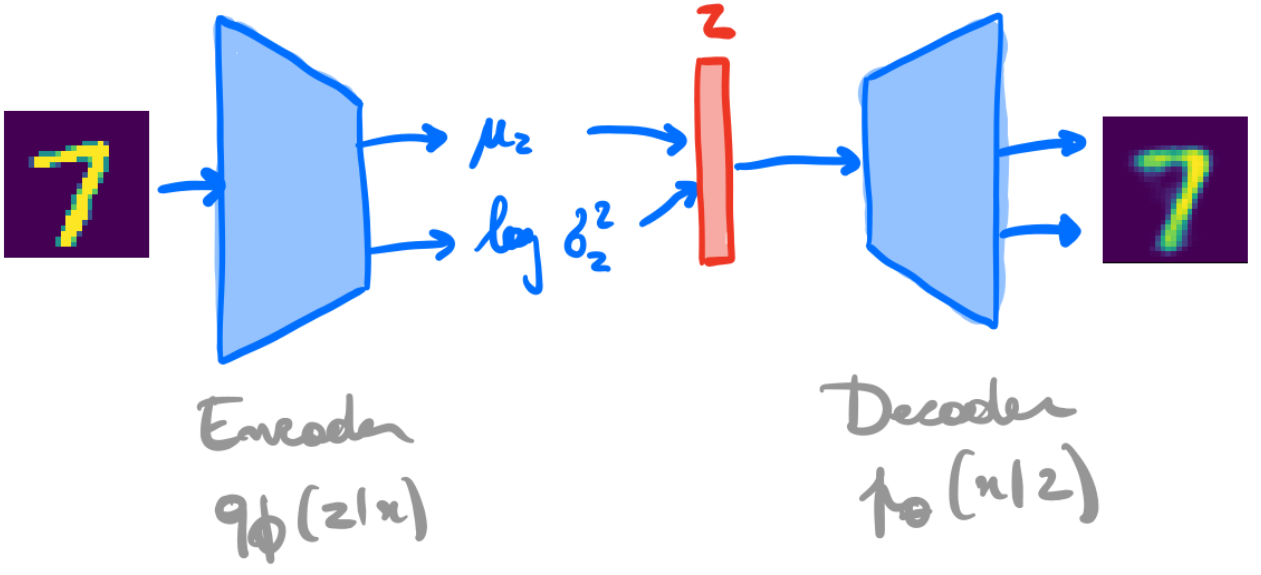
which we can now estimate with Monte Carlo integration.

The last required ingredient is the evaluation of the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ given the change of variable g. As long as g is invertible, we have

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

Step-by-step example

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


z
μ_z
$\log \sigma_z^2$
Encoder
$q_\phi(z|x)$
Decoder
$p_\theta(x|z)$

- Decoder $p_\theta(\mathbf{x}|\mathbf{z})$:

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

- Encoder $q_\phi(\mathbf{z}|\mathbf{x})$:

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

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

Plugging everything together, the objective can be expressed as

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

where the negative KL divergence can be expressed analytically as

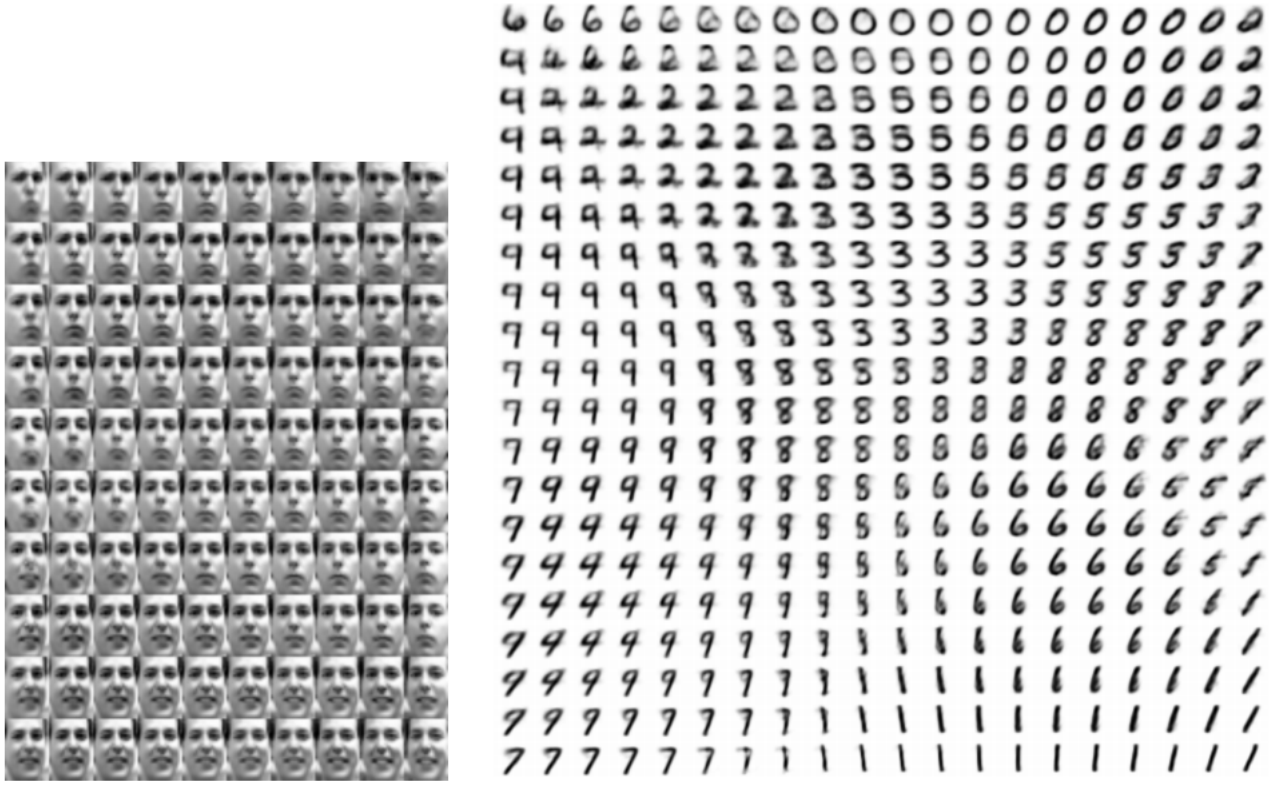
$$-\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) = \frac{1}{2}\sum_{j=1}^{d}\left(1+\log(\sigma_j^2(\mathbf{x};\phi)) - \mu_j^2(\mathbf{x};\phi) - \sigma_j^2(\mathbf{x};\phi)\right),$$

which allows to evaluate its derivative without approximation.

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

—

Credits: Kingma and Welling, 2013.



(a) Learned Frey Face manifold

(b) Learned MNIST manifold

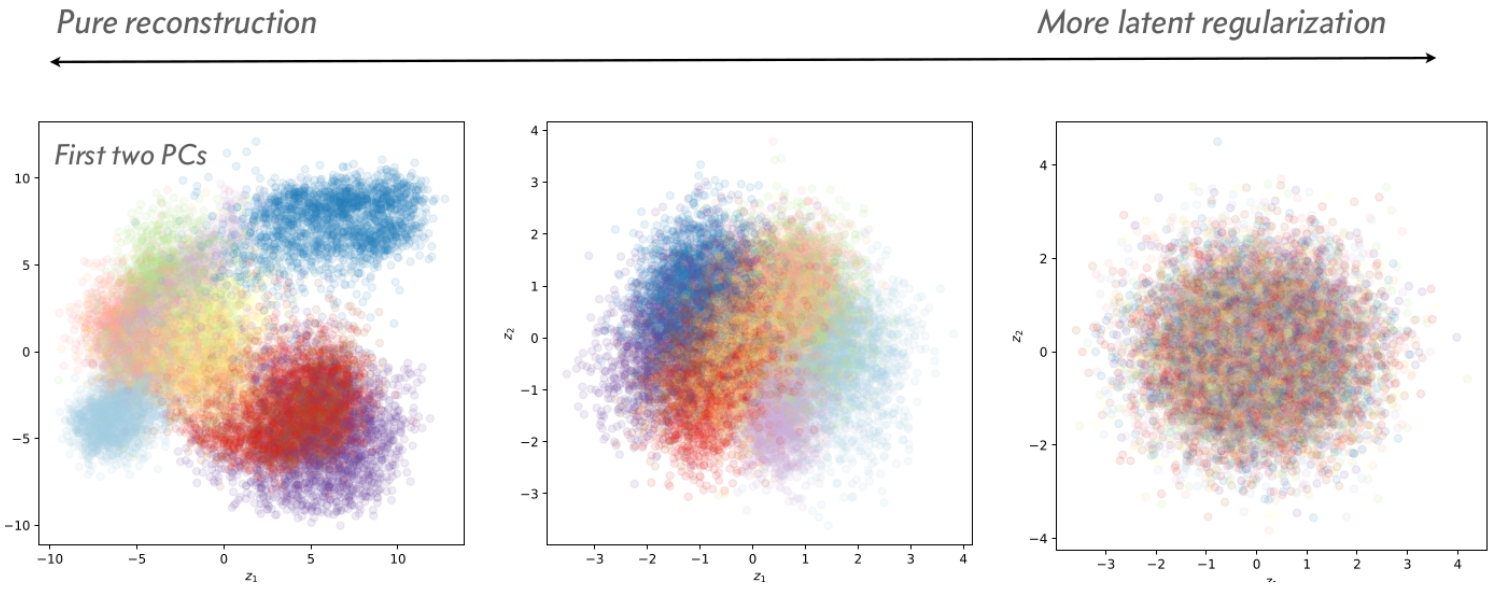
Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the 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 $\boldsymbol{\theta}$.

—

Credits: Kingma and Welling, 2013.

A semantically meaningful latent space

The prior-matching term $\mathbf{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$ enforces simplicity in the latent space, encouraging learned semantic structure and disentanglement.

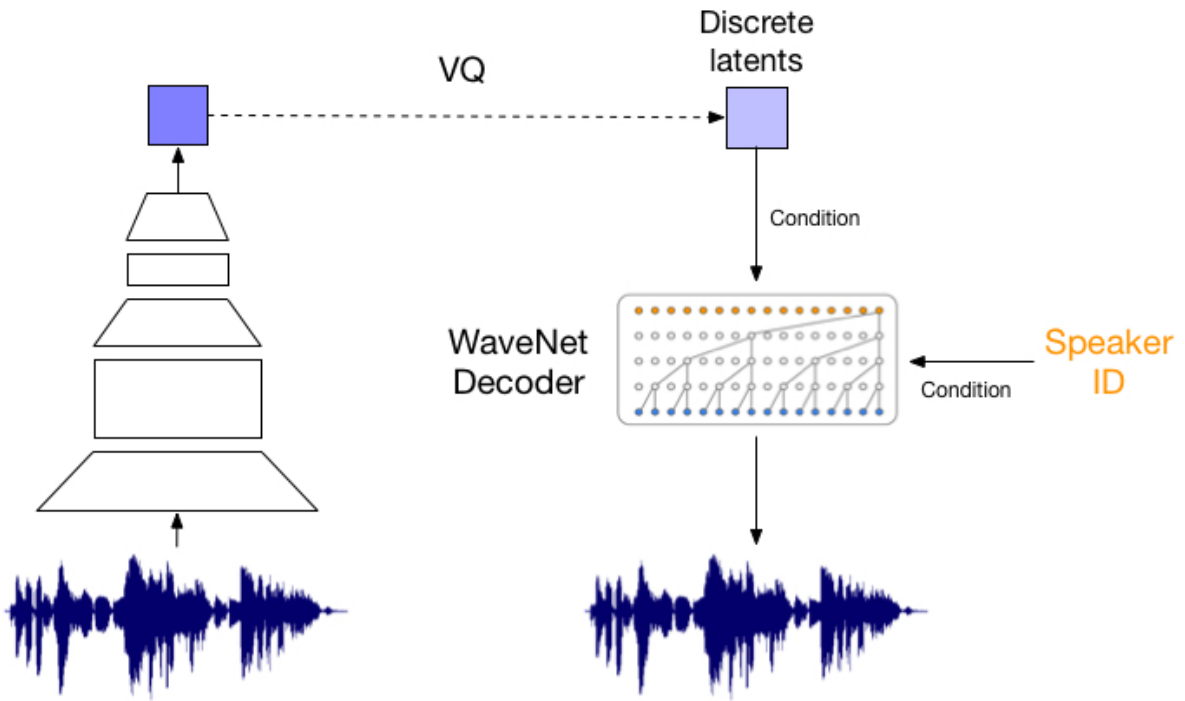


—

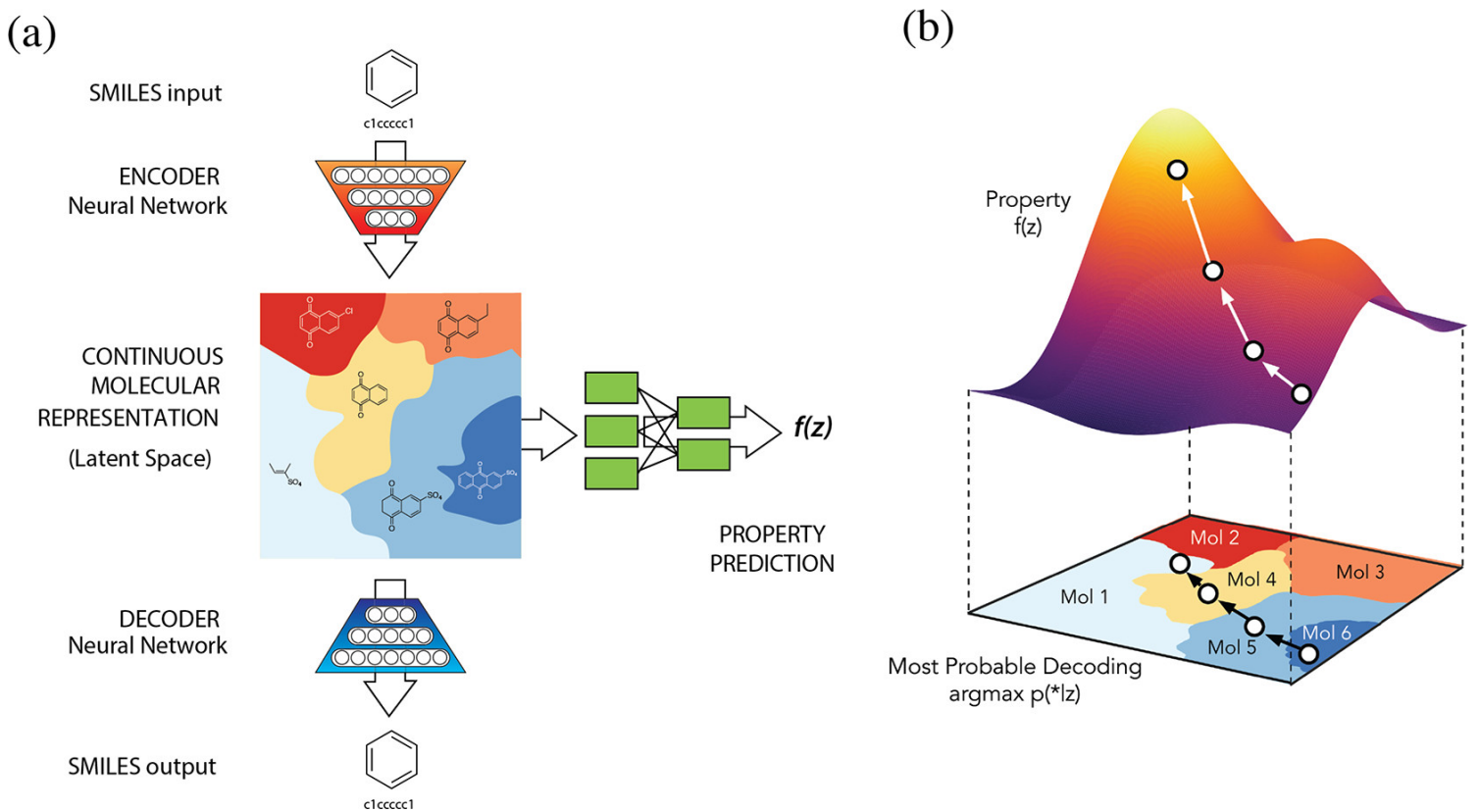
Credits: Siddharth Mishra-Sharma, 2023.

Some selected applications

Hierarchical **compression of images and other data**,
e.g., in video conferencing systems (Gregor et al, 2016).



Voice style transfer [demo] (van den Oord et al, 2017).



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

The end.