Introduction to Variational Auto-Encoders (VAE)

Tomasz Kajdanowicz, Piotr Bielak, Maciej Falkiewicz, Kacper Kania, Piotr Zieliński

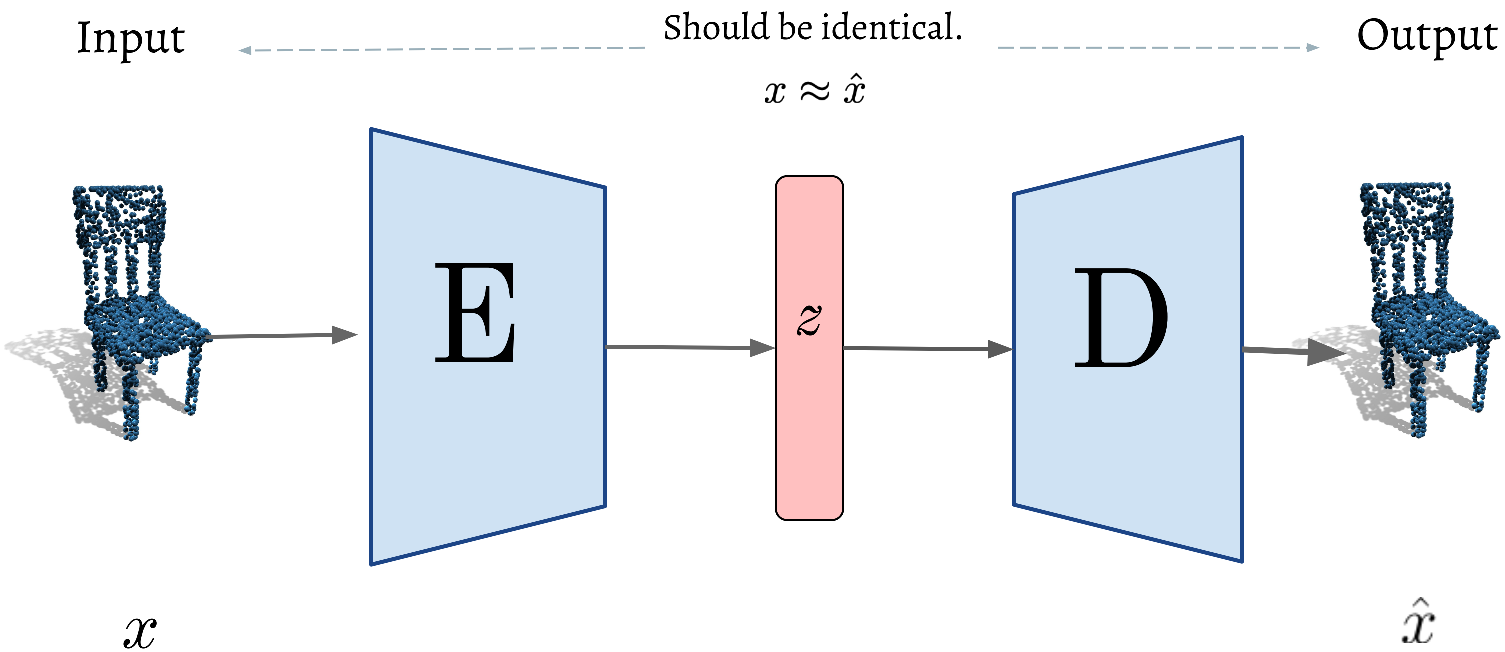
Rerefences

1. **Goodfellow, Ian, "NIPS 2016 Tutorial: Generative Adversarial Networks " arXiv preprint arXiv:1701.00160 (2016)**
2. **Doersch, C. (2016). Tutorial on variational autoencoders. arXiv preprint arXiv:1606.05908.**
3. **Welling, Max; Kingma, Diederik P. (2019). "An Introduction to Variational Autoencoders". Foundations and Trends in Machine Learning. 12 (4): 307–392. arXiv:1906.02691**

Further reading:

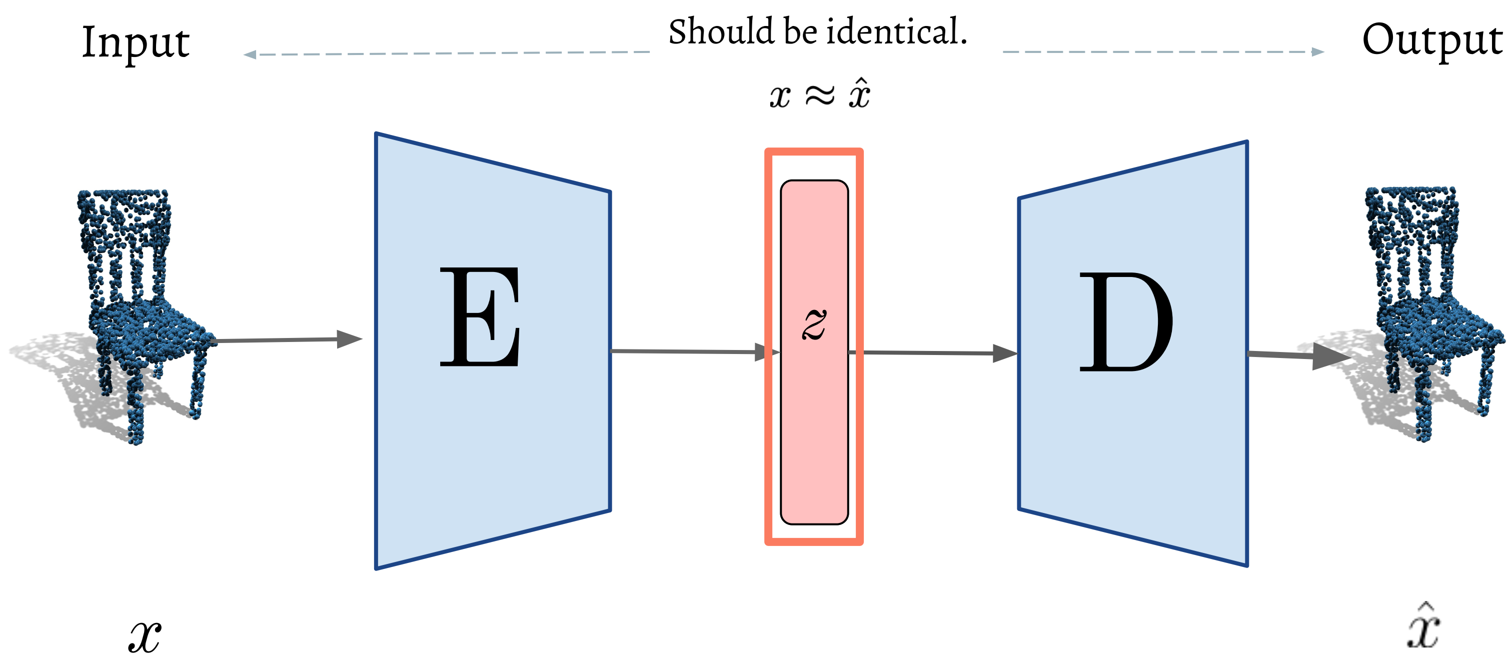
Presentation fully based on: presentation of Wojciech Stokowiec, Maciej Zięba, Maciej Zamorski, Tooploox, PLinML

Autoencoder (AE)



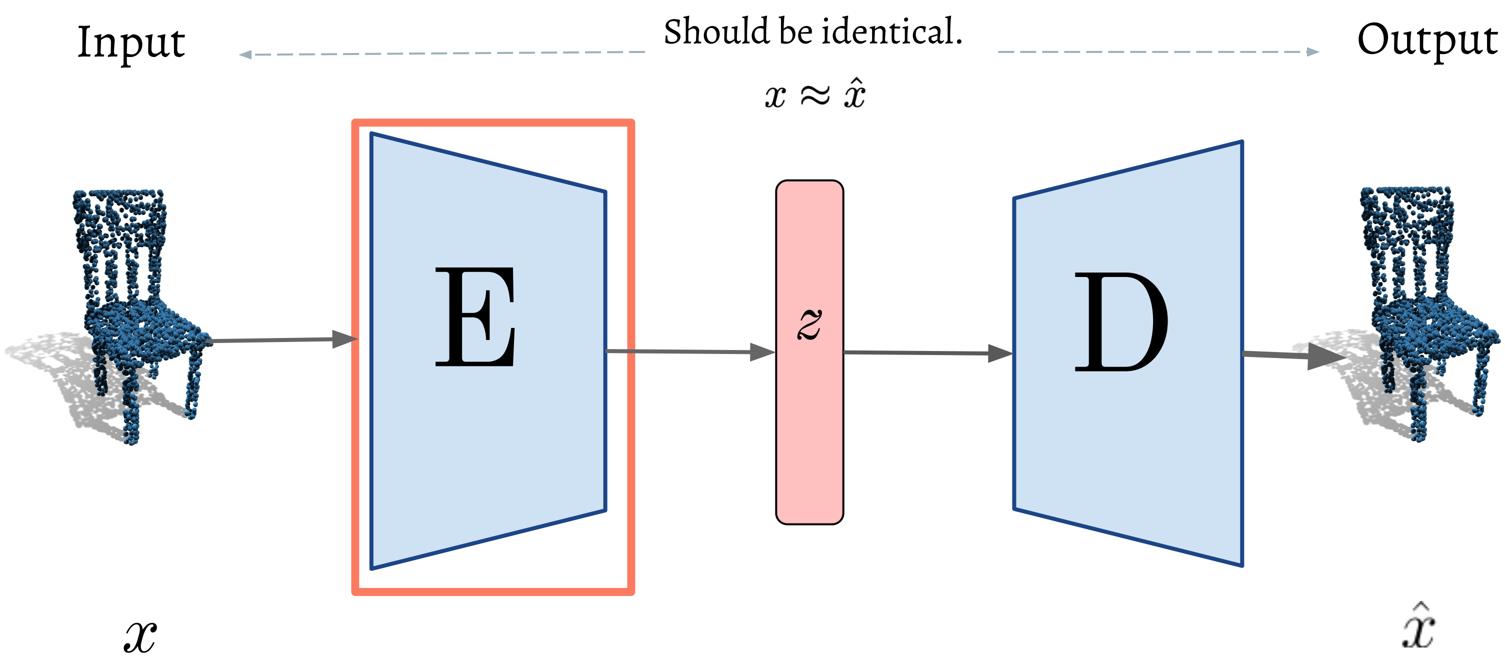
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

Autoencoder (AE)



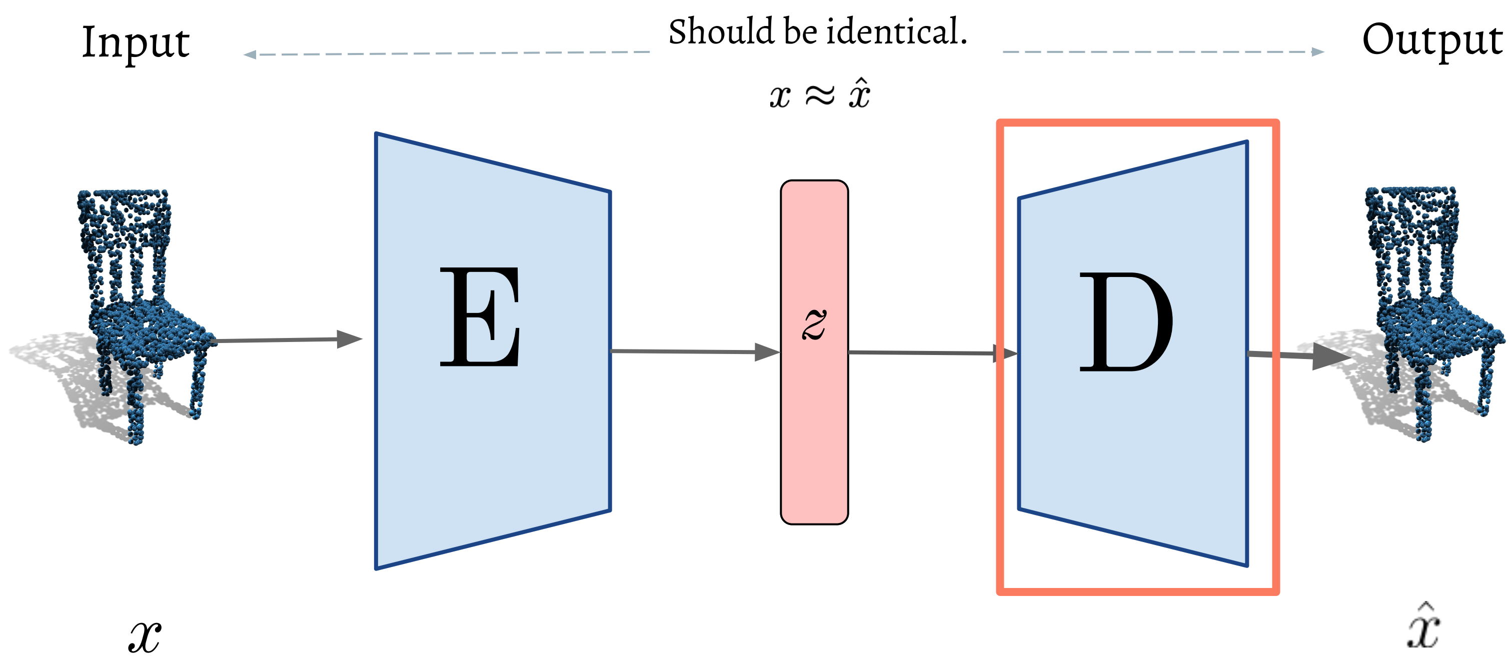
We want feature space to capture meaningful factors of variation in data

Autoencoder (AE)



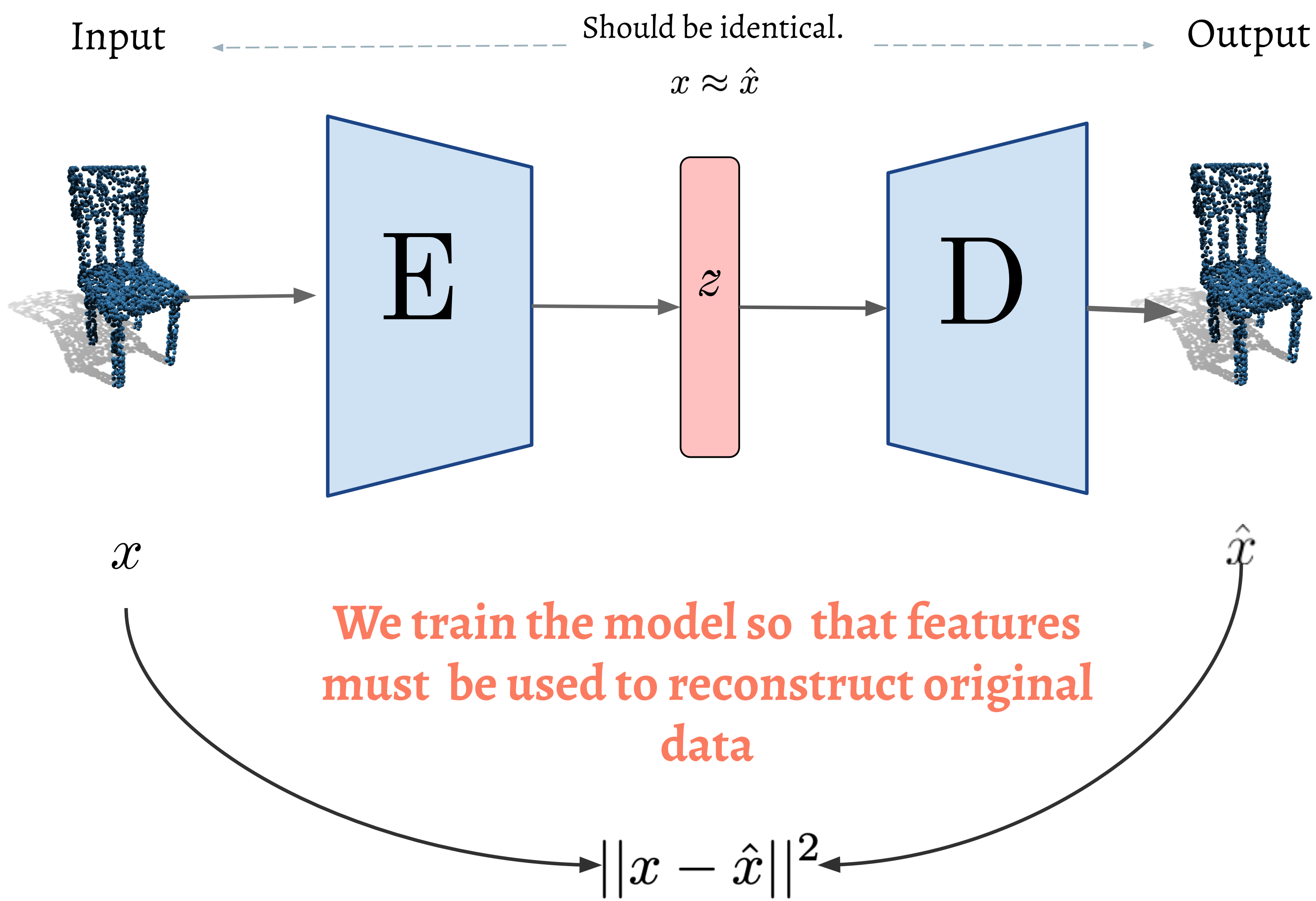
We have encoding network (E) that encodes input example in feature space

Autoencoder (AE)



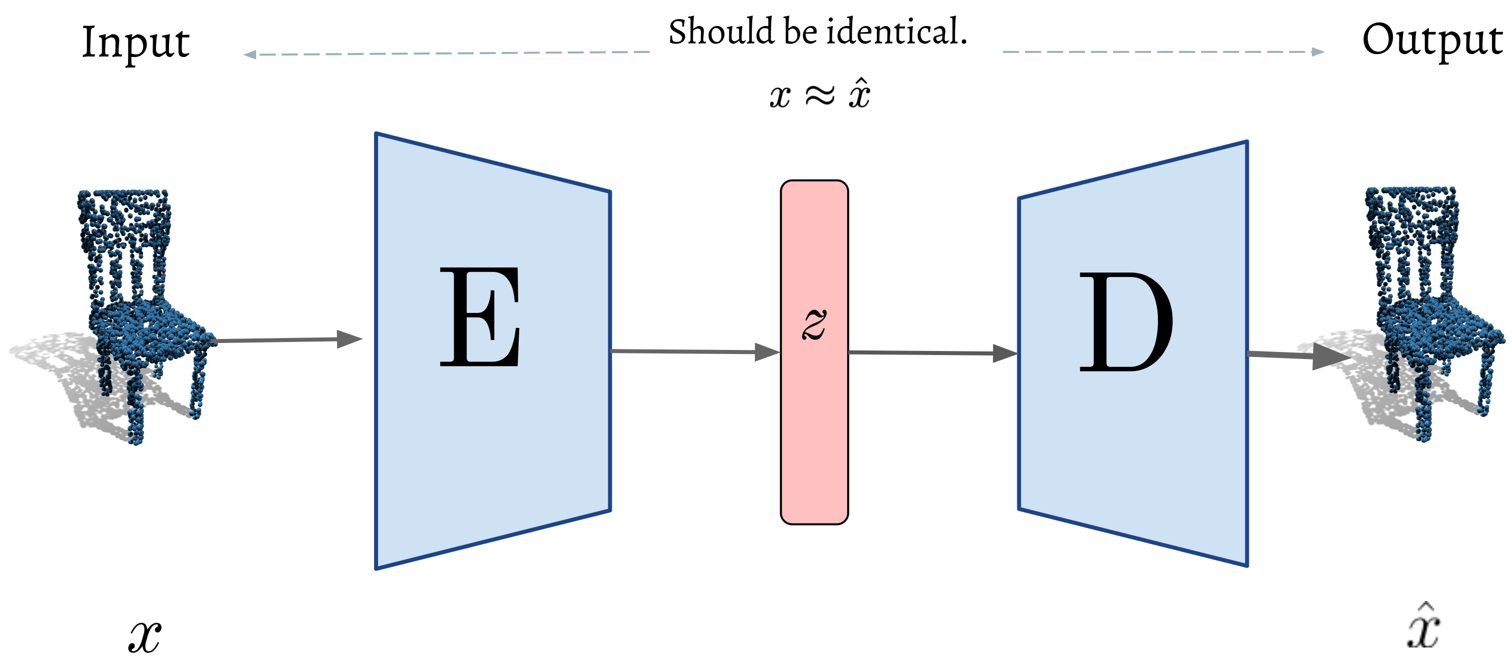
We have decoding network (G) that restores examples from features space

Autoencoder (AE)



We train the model so that features must be used to reconstruct original data

Autoencoder (AE)

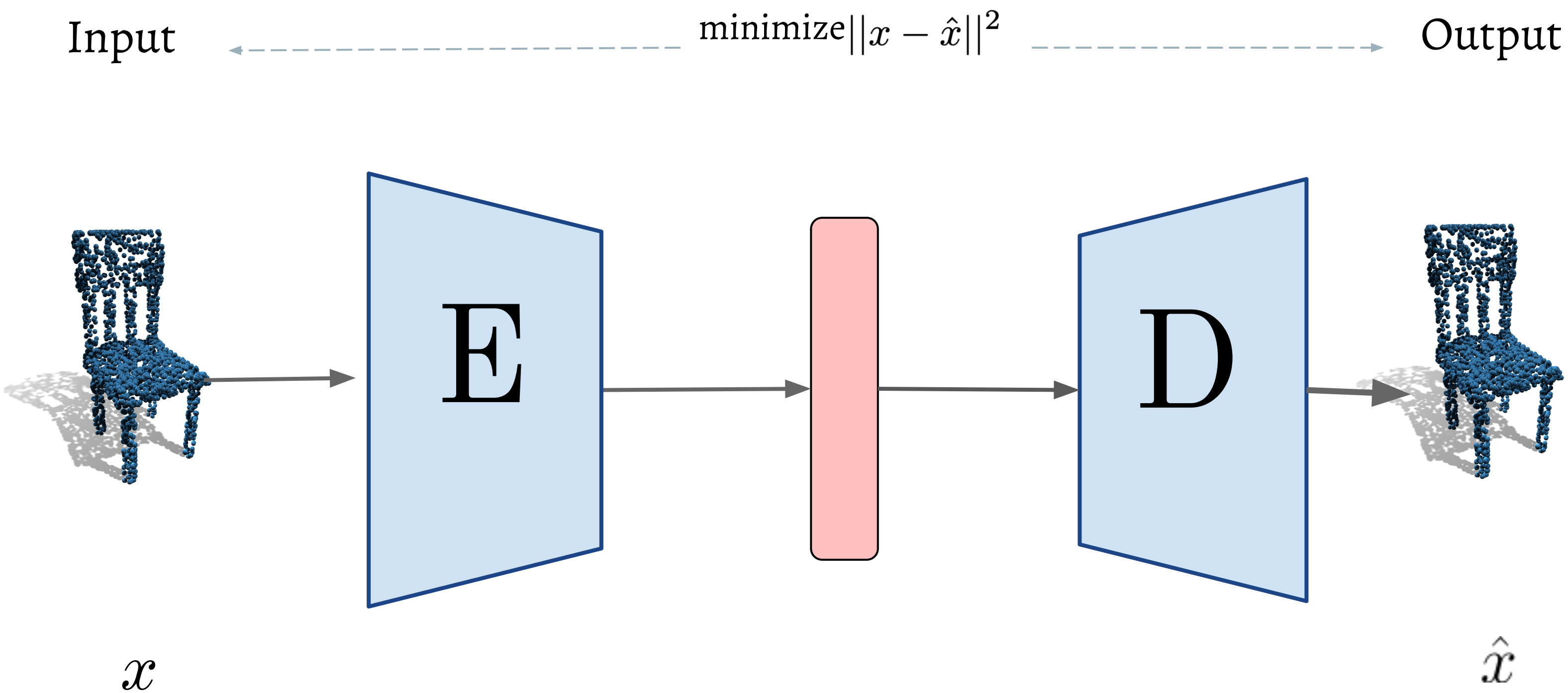


Fun fact: if linear activations are used, then optimal weights for an autoencoder can be obtained with PCA.

Variational Autoencoders

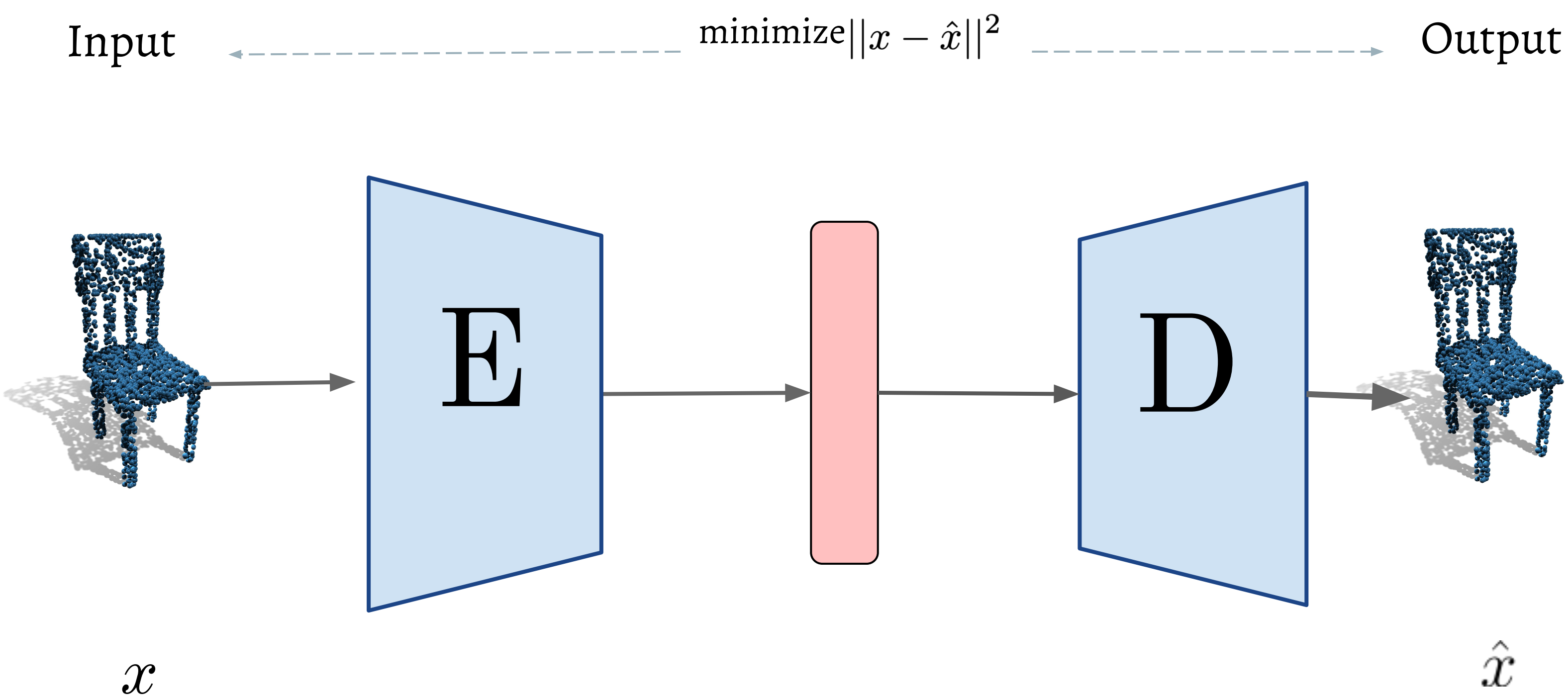
(in pictures)

Variational Autoencoder (VAE)



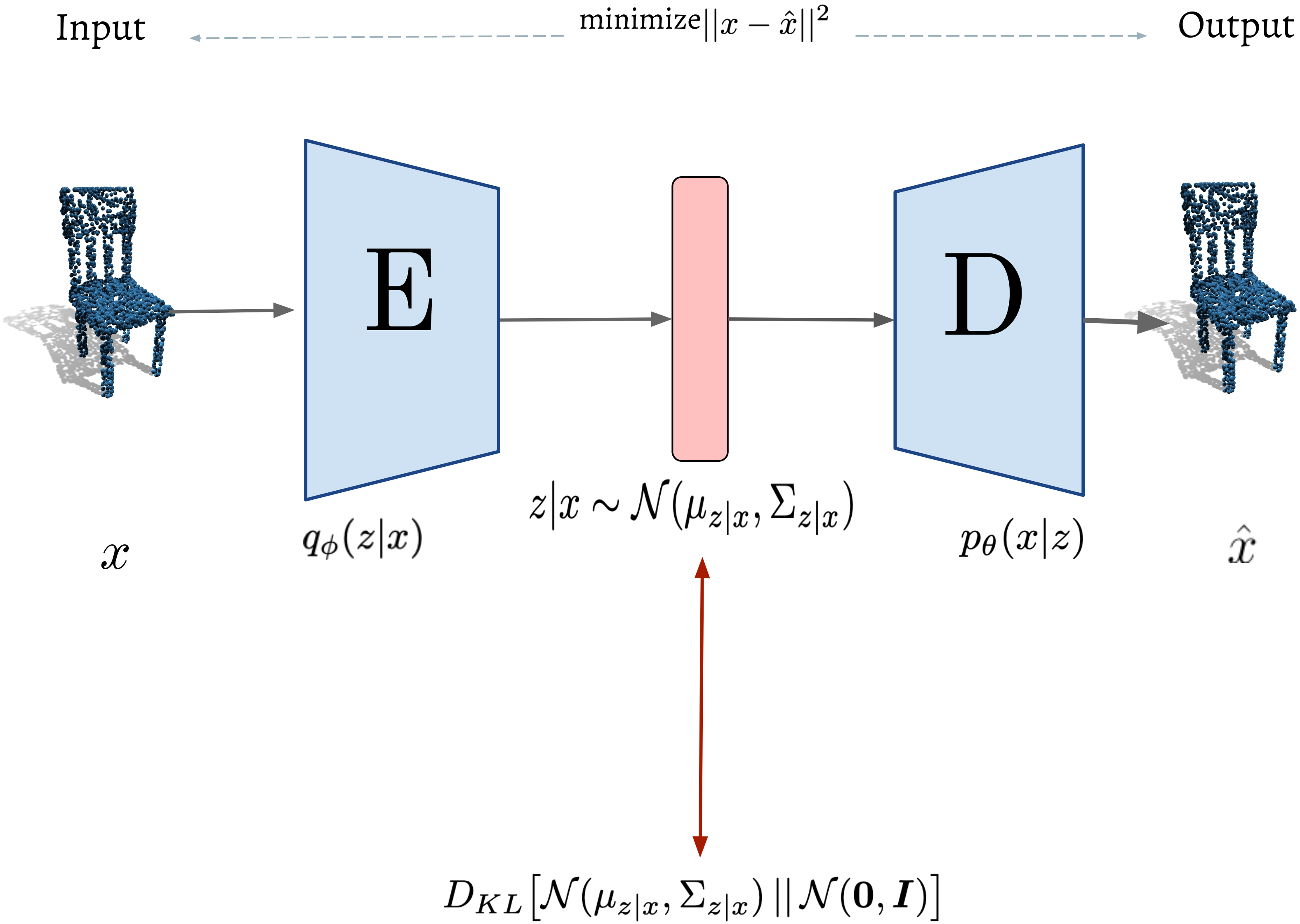
Start with a simple autoencoder ...

Variational Autoencoder (VAE)

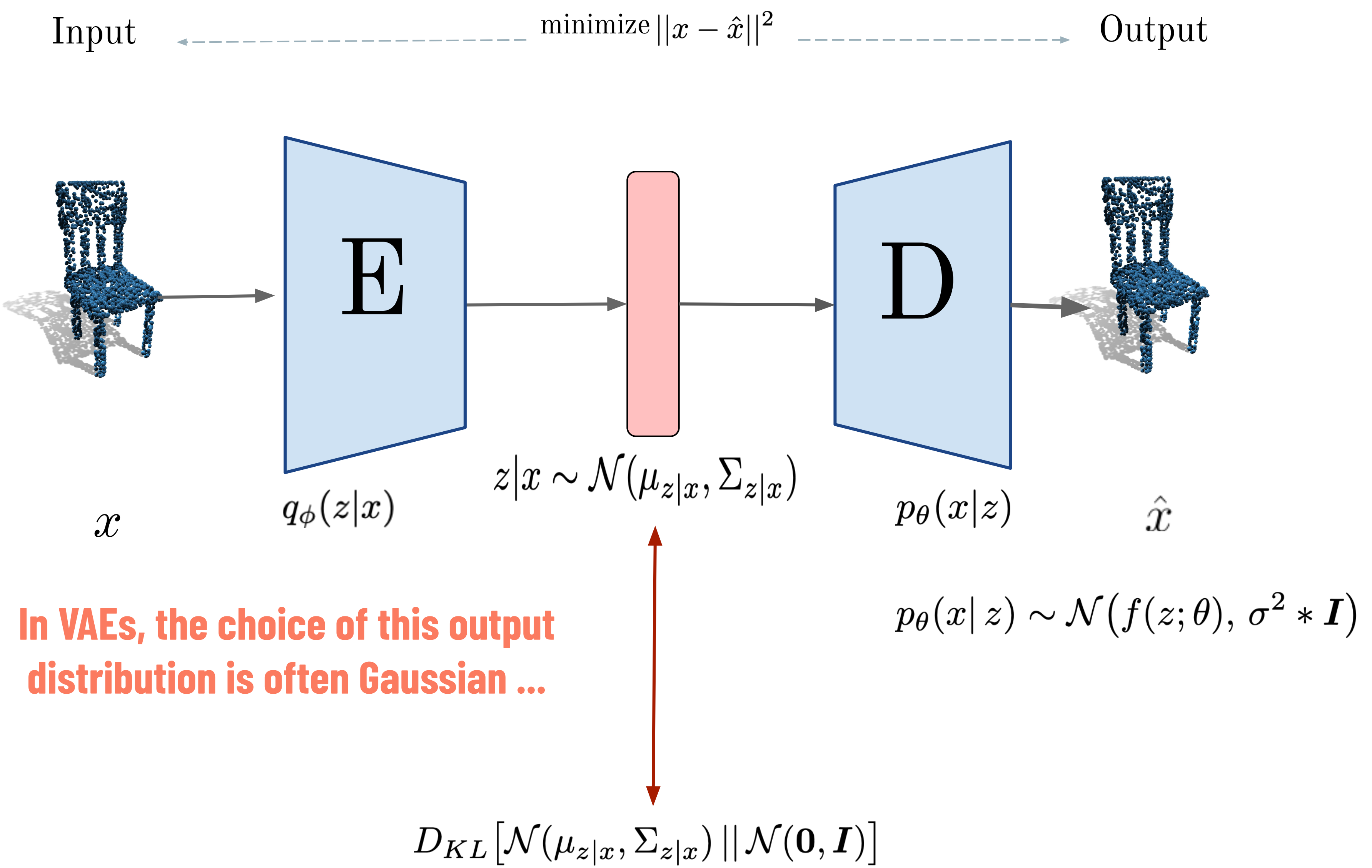


... add a probabilistic spin!

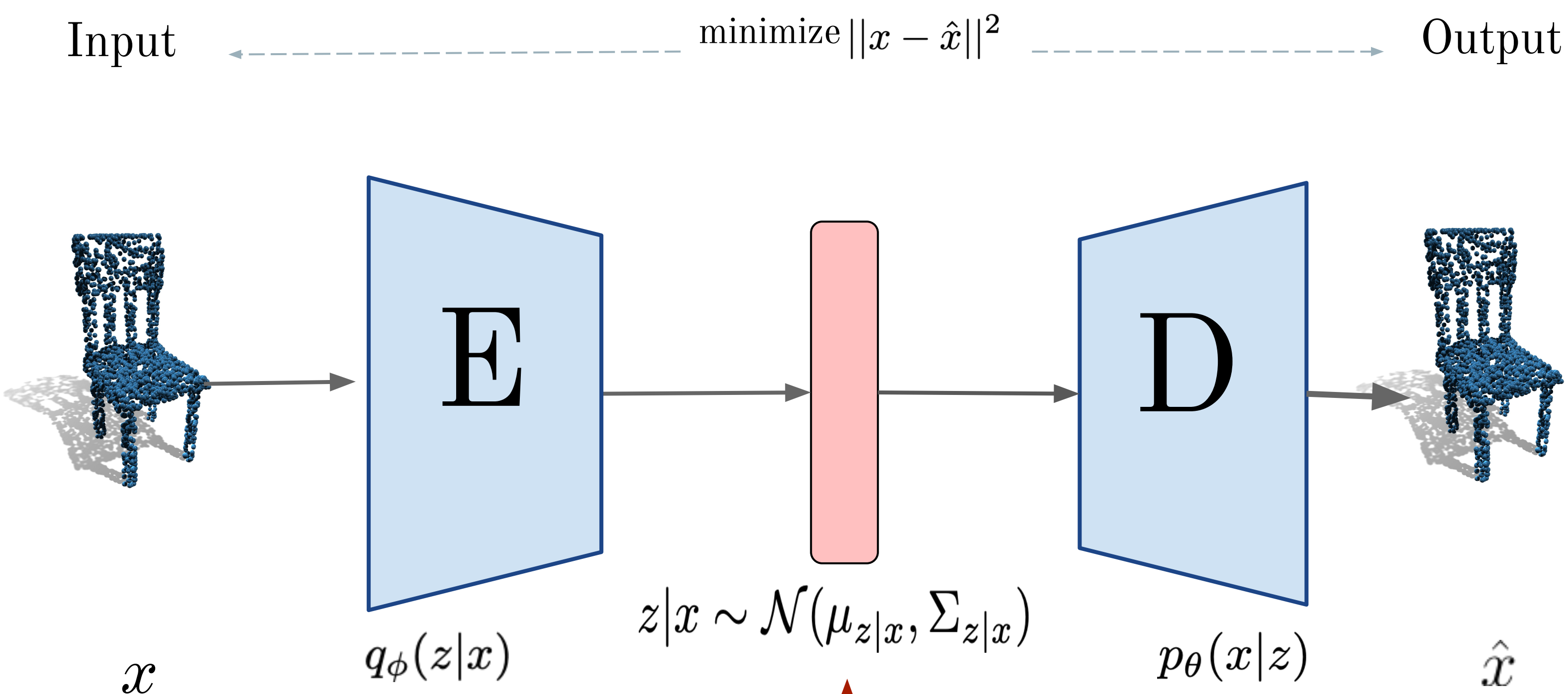
Variational Autoencoder (VAE)



Variational Autoencoder (VAE)



Variational Autoencoder (VAE)

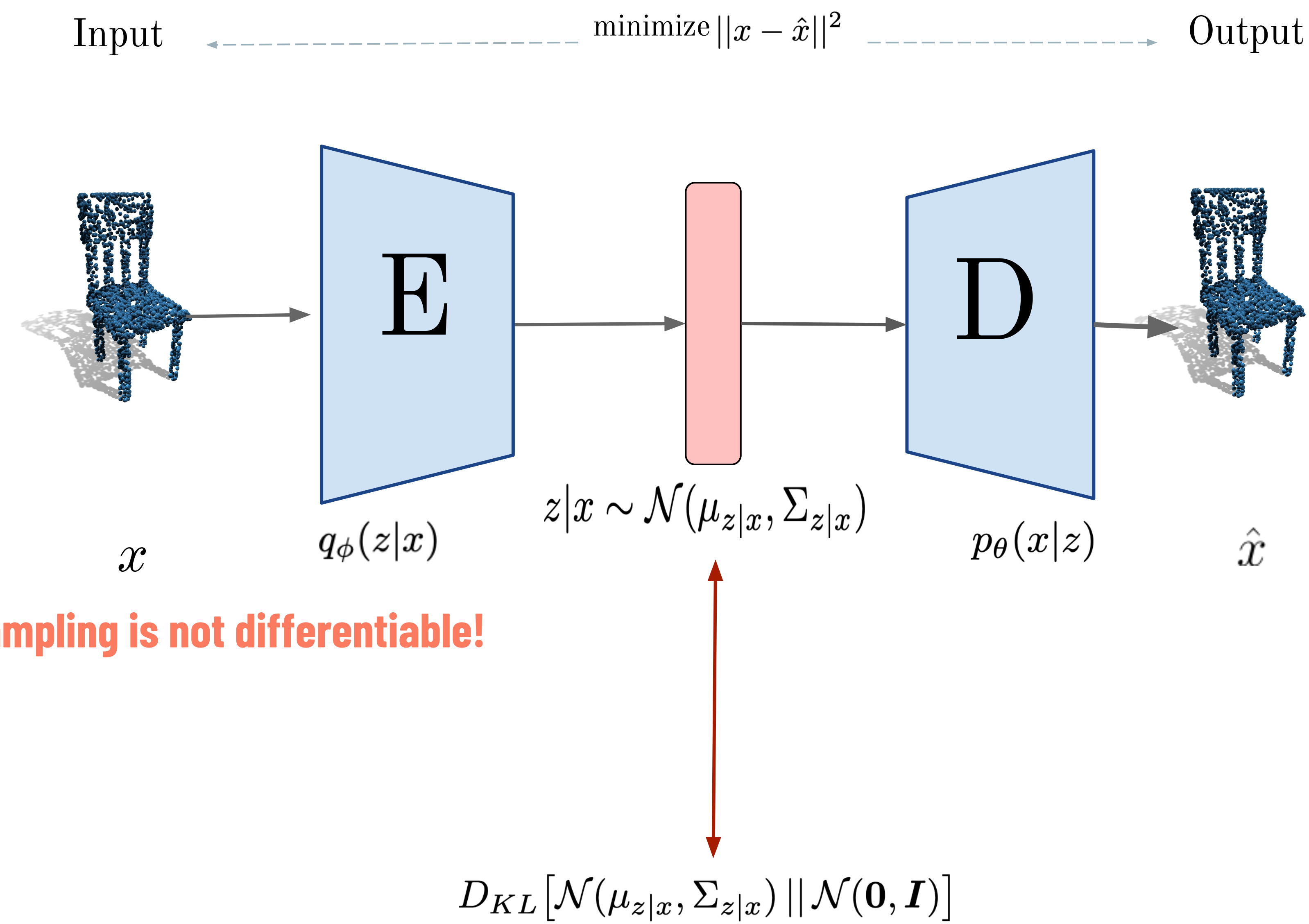


... with mean $f(z;\theta)$ and covariance equal to the identity matrix times some scalar, usually set to 1

$$p_\theta(x|\,z) \sim \mathcal{N}\left(f(z;\theta),\, \sigma^2 * \boldsymbol{I}\right)$$

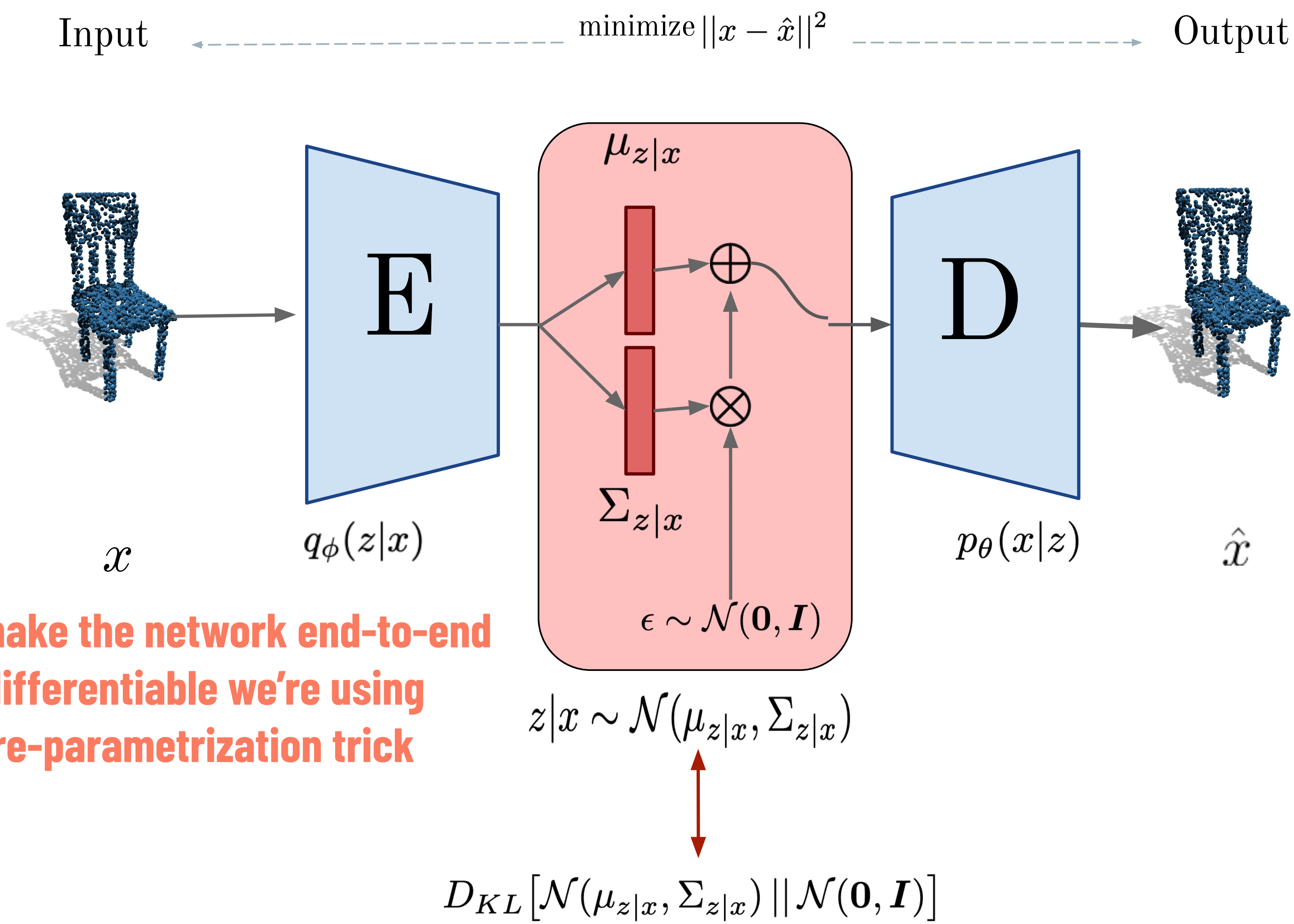
$$D_{KL}\left[\mathcal{N}(\mu_{z|x}, \Sigma_{z|x})\,||\,\mathcal{N}(\mathbf{0}, \boldsymbol{I})\right]$$

Variational Autoencoder (VAE)



Sampling is not differentiable!

Variational Autoencoder (VAE)



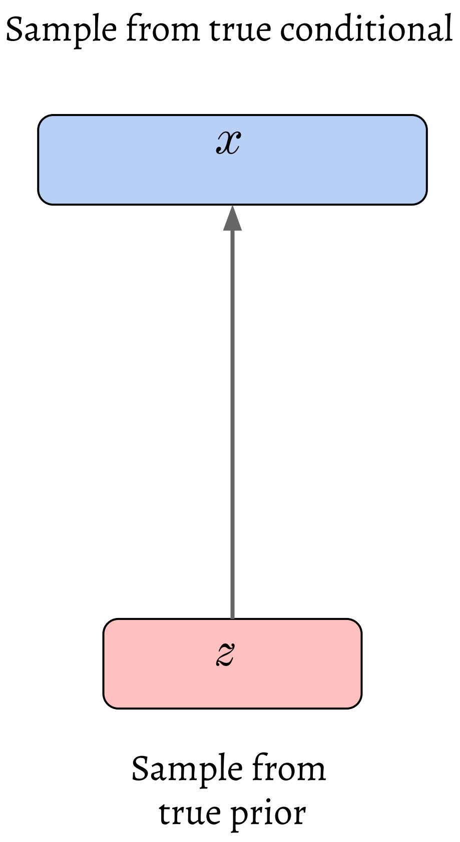
To make the network end-to-end differentiable we're using re-parametrization trick

Variational Autoencoders

(in equations)

Variational Autoencoder (VAE)

We assume that training data *x* is generated from a latent (unobserved) random variable *Z* with samples denoted as *z* with unknown distribution.



How to represent this model, to make the tractable?

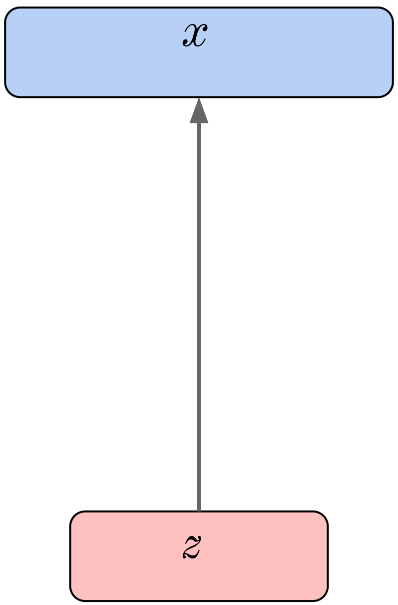
- Chose prior o be simple and computationally inexpensive (gaussian).

- Approximate the conditional using neural network

Variational Autoencoder (VAE)

We assume that training data x is generated from a latent (unobserved) random variable Z with samples denoted as z with unknown distribution.

Sample from true conditional



Sample from true prior

How to learn the parameters of the approximation of the true conditional?

- Maximize the likelihood of the training data!

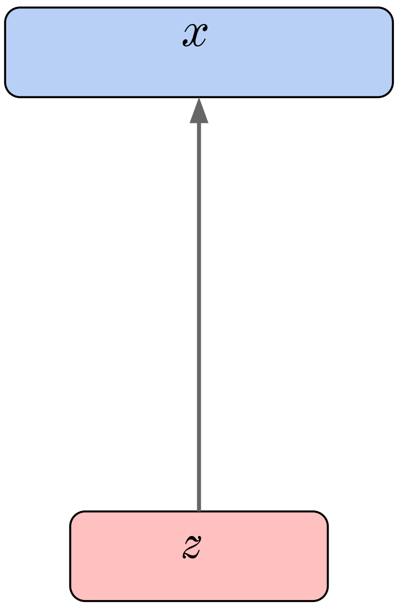
$$p(x) = \int p_\theta(x|z)p(z)dz$$

Would have to integrate over all the possible value of prior!

Variational Autoencoder (VAE)

We assume that training data *x* is generated from a latent (unobserved) random variable *Z* with samples denoted as *z* with unknown distribution.

Sample from true conditional



Sample from true prior

How to learn the parameters of the approximation of the true conditional?

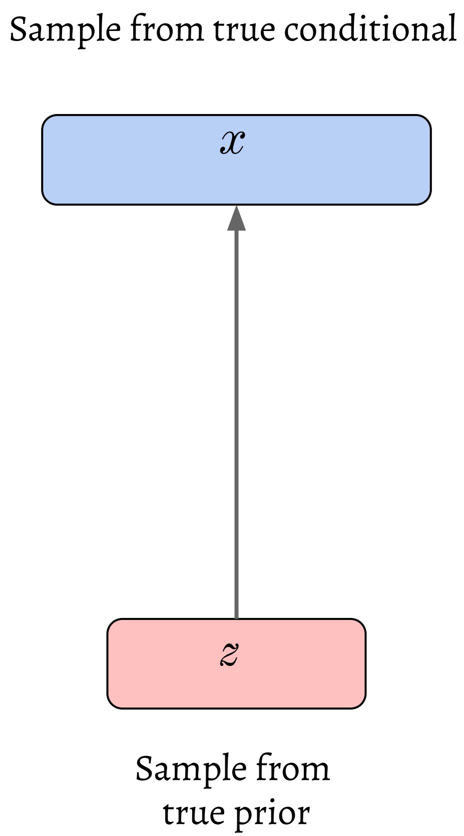
- Maximize the likelihood of the training data!

$$p(x) = \int p_\theta(x|\, z)p(z)dz$$

The key idea of VAE is to attempt to sample values of Z that are likely to have produced data and compute conditional from those

Variational Autoencoder (VAE)

We assume that training data *x* is generated from a latent (unobserved) random variable *Z* with samples denoted as *z* with unknown distribution.



How to learn the parameters of the approximation of the true conditional?

- Maximize the likelihood of the training data!

$$p(x) = \int p_\theta(x|\, z)p(z)dz$$

Add another network $q_\phi(z \mid x)$, the inference network to approximate the true posterior $p_\theta(z|x)$

Variational Autoencoder (VAE) - training objective

$$
\begin{aligned}
\log p_\theta(x) &= \mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x)\Big] \quad (p_\theta(x) \text{ is independent of } z) \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x\,|\,z)p(z)}{p(z\,|\,x)}\right] \quad (\text{Bayes' Rule}) \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x\,|\,z)p(z)}{p(z\,|\,x)}\frac{q_\phi(z\,|\,x)}{q_\phi(z\,|\,x)}\right] \quad (\text{Multiply by } 1) \\
&= \mathbb{E}_z\big[\log p_{\theta(x\,|\,z)}\big] - \mathbb{E}_z\left[\log \frac{q_\phi(z\,|\,x)}{p(z)}\right] + \mathbb{E}_z\left[\log \frac{q_\phi(z\,|\,x)}{p(z\,|\,x)}\right] \quad (\text{Logarithms}) \\
&= \underbrace{\mathbb{E}_z\big[\log p_{\theta(x\,|\,z)}\big] - D_{KL}(q_\phi(z\,|\,x)\,||\,p(z))}_{\mathcal{L}(x,\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z\,|\,x)\,||\,p(z|x))}_{\geq 0}
\end{aligned}
$$

Evidence Lower Bound (**ELBO**)

$$\log p_\theta(x) \geq \mathcal{L}(x,\theta,\phi)$$

MLE training objective

$$\theta^*, \phi* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x_i,\theta,\phi)$$

Variational Autoencoder (VAE) - training objective

$$\log p_\theta(x) = \underbrace{\mathbb{E}_z\left[\log p_{\theta(x\,|\,z)}\right] - D_{KL}(q_\phi(z\,|\,x)\,||\,p(z))}_{\mathcal{L}(x,\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z\,|\,x)\,||\,p(z|x))}_{\geq 0}$$

"Reconstruction error"

We want to maximize this

Approximate posterior regularization

Variational Autoencoder (VAE) - training objective

$$\log p_\theta(x) = \underbrace{\mathbb{E}_z\big[\log p_{\theta(x\,|\,z)}\big] - D_{KL}(q_\phi(z\,|\,x)\,||\,p(z))}_{\mathcal{L}(x,\theta,\phi)} + \boxed{\underbrace{D_{KL}(q_\phi(z\,|\,x)\,||\,p(z|x))}_{\geq 0}}$$

The KL divergence of the approximation and true posterior distribution, is greater or equal to zero. We can't optimize directly.

$$\log p_\theta(x) \geq \mathcal{L}(x_i, \theta, \phi)$$

$$\log p_\theta(x) \geq \mathbb{E}_z\big[\log p_{\theta(x\,|\,z)}\big] - \; D_{KL}(q_\phi(z\,|\,x)\,||\,p(z))$$

Variational Autoencoder (VAE) - reconstruction term

$$\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_{\theta(x\mid z)}\right]$$

This called the reconstruction term

Let see why training loss is $||x - \hat{x}||^2$ while the objective talks about expectations and log probabilities

Variational Autoencoder (VAE) - reconstruction term

we decided to model the conditional distribution as multivariate gaussian

$$\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_{\theta(x\mid z)}\right] p_\theta(x|\, z) \sim \mathcal{N}\left(f(z;\theta),\, \sigma^2 * \boldsymbol{I}\right)$$

Recall that

$$p(x|\mu,\Sigma) = \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

$$\ln p(x|\mu,\Sigma) = -\frac{1}{2}\ln|\Sigma| - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) + C$$

Variational Autoencoder (VAE) - reconstruction term

we decided to model the conditional distribution as multivariate gaussian

$$\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_{\theta(x\,|\,z)}\right] p_\theta(x|\,z) \sim \mathcal{N}\left(f(z;\theta),\, \sigma^2 * \boldsymbol{I}\right)$$

$$\ln p(x|\mu,\Sigma) = -\frac{1}{2}\ln|\Sigma| - \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) + C$$

$$\ln p(x|\mu,\sigma^2 I) = -\frac{1}{2}\ln|\sigma^2 I| - \frac{1}{2}(x-\mu)^T(\sigma^2 I)^{-1}(x-\mu) + C$$

$$= \cdots$$

Only L-2 norm is dependent on model parameters!

$$= -\frac{1}{2}\ln\sigma^2 - \frac{1}{2}||x-\mu||^2/\sigma^2 + C$$

$$\ln p_\theta(x|f(z;\,\theta), I) = \boxed{-\frac{1}{2}||x - f(z;\,\theta)||^2 + C}$$

Variational Autoencoder (VAE) - Kullback-Leibler divergence

$$D_{KL}(q_\phi(z \mid x) \,||\, p(z))$$

How can we calculate KL divergence between the approximate posterior and prior?

Variational Autoencoder (VAE) - Kullback-Leibler divergence

$$D_{KL}[\mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \,||\, \mathcal{N}(0, I)]$$

Both approximate posterior and prior are assumed to be multivariate gaussian, we can do this analytically!

Variational Autoencoder (VAE) - Kullback-Leibler divergence

$$D_{KL}[\mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \mid\mid \mathcal{N}(0, I)] =$$

$$\frac{1}{2}\left(\mathrm{tr}(\Sigma_{z|x}) + \mu_{z|x}^T \mu_{z|x} - k - \log\det(\Sigma_{z|x})\right)$$

Variational Autoencoder (VAE) - Kullback-Leibler divergence

$$D_{KL}[\mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \,||\, \mathcal{N}(0, I)] =$$

$$\frac{1}{2}\left(\boxed{\mathrm{tr}(\Sigma_{z|x})} + \mu_{z|x}^T \mu_{z|x} - k - \log\det(\Sigma_{z|x})\right)$$

Easy, just sum!

Variational Autoencoder (VAE) - Kullback-Leibler divergence

$$D_{KL}[\mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \,||\, \mathcal{N}(0, I)] =$$

$$\frac{1}{2}\left(\operatorname{tr}(\Sigma_{z|x}) + \boxed{\mu_{z|x}^T \mu_{z|x}} - k - \log\det(\Sigma_{z|x})\right)$$

Easy, simple dot product!

Variational Autoencoder (VAE) - Kullback-Leibler divergence

$$D_{KL}[\mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \,||\, \mathcal{N}(0, I)] =$$

$$\frac{1}{2}\left(\mathrm{tr}(\Sigma_{z|x}) + \mu_{z|x}^T \mu_{z|x} - \boxed{k} - \log\det(\Sigma_{z|x})\right)$$

Easy, scalar.
Equal to dimensionality of
the distribution

Variational Autoencoder (VAE) - Kullback-Leibler divergence

$$D_{KL}[\mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \,||\, \mathcal{N}(0, I)] =$$

$$\frac{1}{2}\left(\mathrm{tr}(\Sigma_{z|x}) + \mu_{z|x}^T \mu_{z|x} - k - \boxed{\log\det(\Sigma_{z|x})}\right)$$

Easy, as we assumed that the covariance matrix of approximate posterior is diagonal.

Variational Autoencoder (VAE) - Kullback-Leibler divergence

$$D_{KL}[\mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \mid\mid \mathcal{N}(0, I)] =$$

$$\frac{1}{2} \sum_{i=1}^{k} (\sigma_i^2 + \mu_i^2 - \log \sigma_i^2 - 1)$$

Variational Autoencoder (VAE) - objective summary

$$\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_{\theta(x\mid z)}\right] - D_{KL}\left[q_\phi(z\mid x)\,||\,p(z)\right]$$

We train the model to maximize this expression consisting of two terms:

- Reconstruction term
- Regularization term on the approximate posterior

Variational Autoencoder (VAE) - objective summary

$$\boxed{\mathbb{E}_{z \sim q_\phi(z|x)}\left[\log p_{\theta(x \mid z)}\right]} - D_{KL}\left[q_\phi(z \mid x) \,||\, p(z)\right]$$

Is maximized by minimizing the reconstruction error

Variational Autoencoder (VAE) - objective summary

$$\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_{\theta(x\,|\,z)}\right] - \boxed{D_{KL}\left[q_\phi(z\,|\,x)\,||\,p(z)\right]}$$

Has a simple analytical formula, that can be easily minimized using gradient methods

Variational Autoencoder (VAE) - objective summary

$$\underbrace{\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_{\theta(x\mid z)}\right]}_{-\frac{1}{2}||x-\hat{x})||^2+C} - \underbrace{D_{KL}\left[q_\phi(z\mid x)\,||\,p(z)\right]}_{\frac{1}{2}\sum_{i=1}^{k}\left(\sigma_i^2+\mu_i^2-\log\sigma_i^2-1\right)}$$

ELBO in theory vs ELBO in “almost code”

Normalizing flows

Tomasz Kajdanowicz, Piotr Bielak, Maciej Falkiewicz,
Kacper Kania, Piotr Zieliński

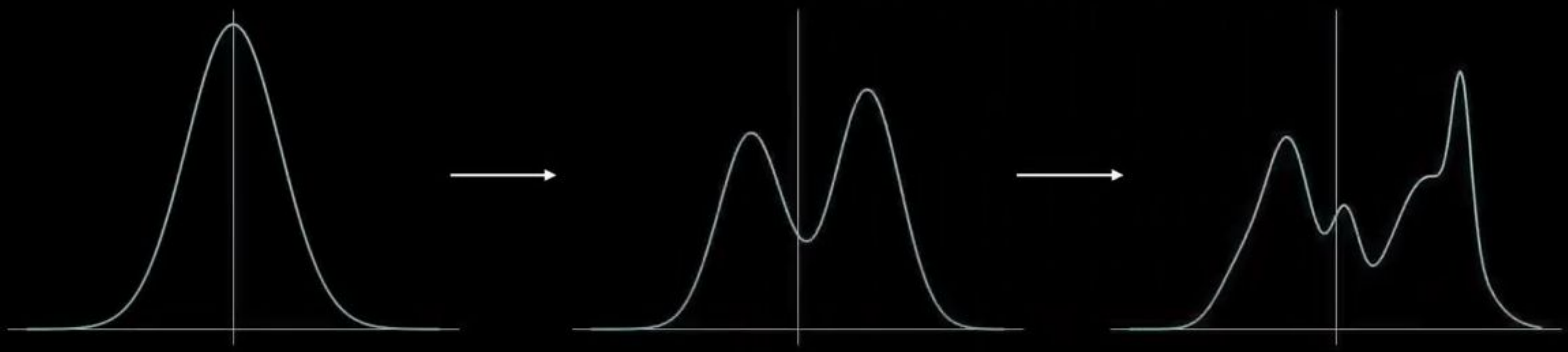
Rerefences

1. Tabak, E. G., & Turner, C. V. (2013). A family of nonparametric density estimation algorithms. Communications on Pure and Applied Mathematics, 66(2), 145-164.
2. **Dinh, L., Krueger, D., & Bengio, Y. (2014). Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516.**
3. **Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. arXiv preprint arXiv:1505.05770.**
4. **Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real nvp." arXiv preprint arXiv:1605.08803 (2016).**

Further reading:

Adam Kosiorek blog post - http://akosiorek.github.io/ml/2018/04/03/norm_flows.html

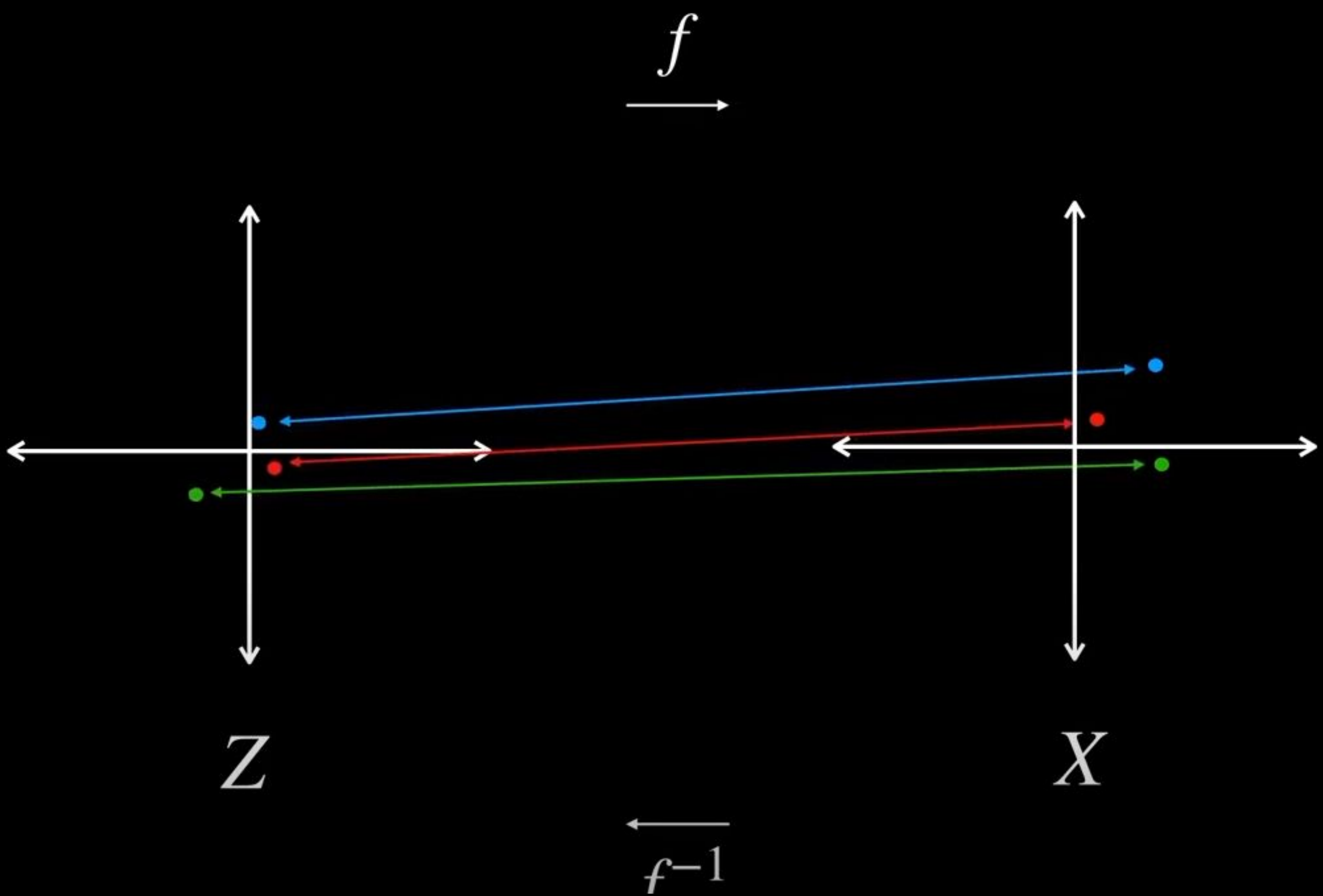
Presentation fully based on: presentation of Ari Seff, Princeton University, https://www.youtube.com/watch?v=i7LjDvsLWCg



$$z \sim p_\theta(z) = \mathcal{N}(z; 0, I)$$

$$x = f_\theta(z) = f_K \circ \ldots \circ f_2 \circ f_1(z)$$

each f_i is invertible (bijective)


f
Z
X
f^{-1}

$$z \sim p_\theta(z) = \mathcal{N}(z; 0, I)$$

$$x = f_\theta(z) = f_K \circ \ldots \circ f_2 \circ f_1(z)$$

$$p_\theta(x) \stackrel{?}{=} p_\theta(f_\theta^{-1}(x))$$

Change of variables formula

$f : Z \to X$, f is invertible

$p_\theta(z)$ defined over $z \in Z$

Change of variables formula:

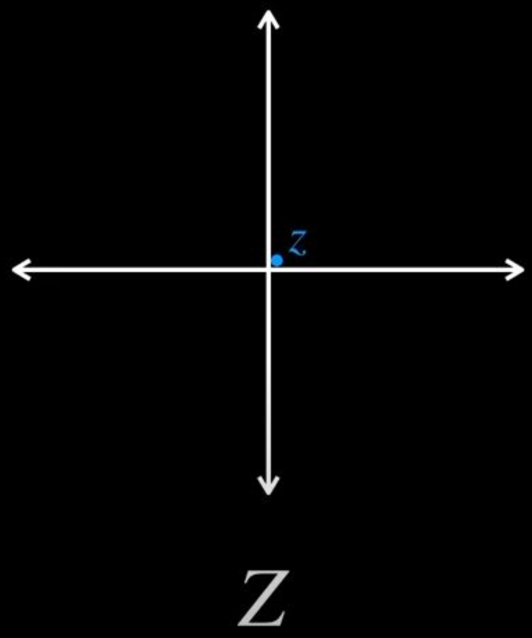
$$p_\theta(x) = p_\theta(f^{-1}(x)) \left| \det\left(\frac{\partial f^{-1}(x)}{\partial x} \right) \right|$$

$$p_\theta(x) = p_\theta(f^{-1}(x)) \left| \det\left(\frac{\partial f^{-1}(x)}{\partial x} \right) \right|$$

$$= p_\theta(z) \left| \det\left(\frac{\partial z}{\partial x} \right) \right|$$

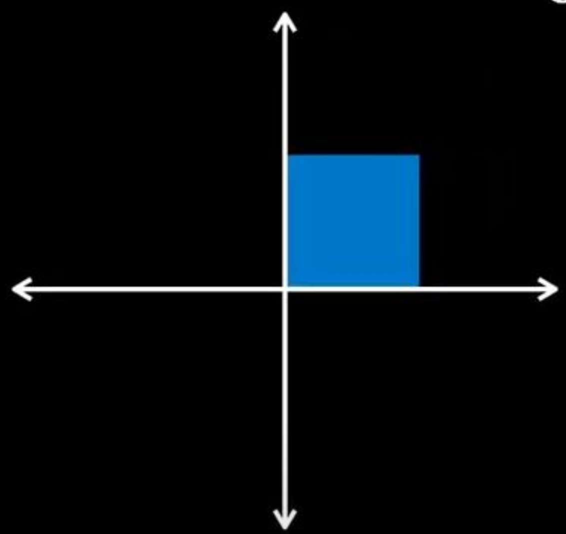
$$p_\theta(x) = p_\theta(z) \left| \det\left(\frac{\partial z}{\partial x} \right) \right|$$

Example

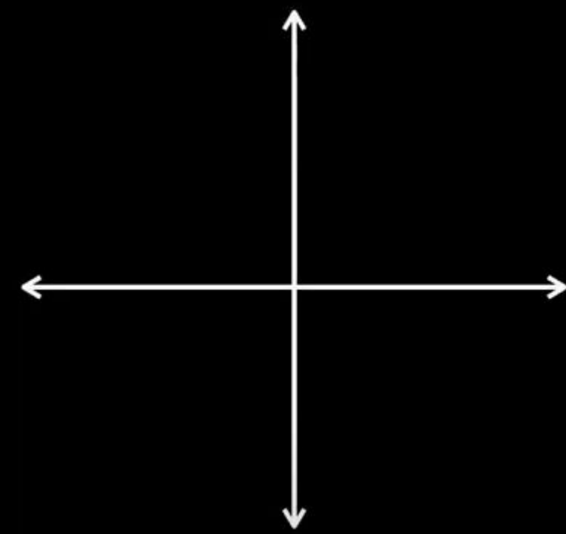


Example

$$J = \begin{bmatrix} \dfrac{\partial x_1}{\partial z_1} & \dfrac{\partial x_1}{\partial z_2} \\ \dfrac{\partial x_2}{\partial z_1} & \dfrac{\partial x_2}{\partial z_2} \end{bmatrix}$$



Z

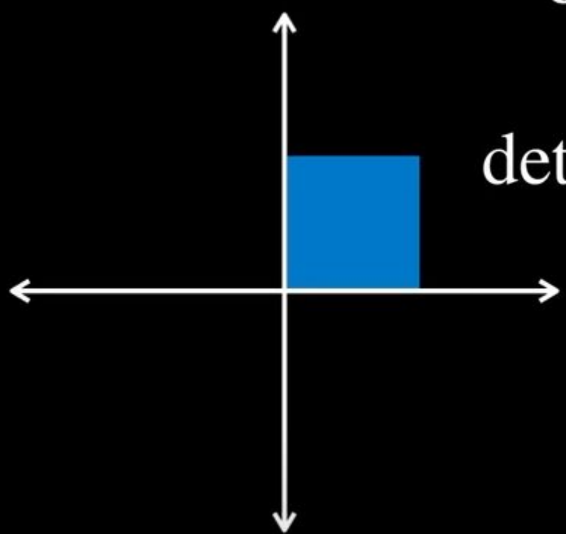


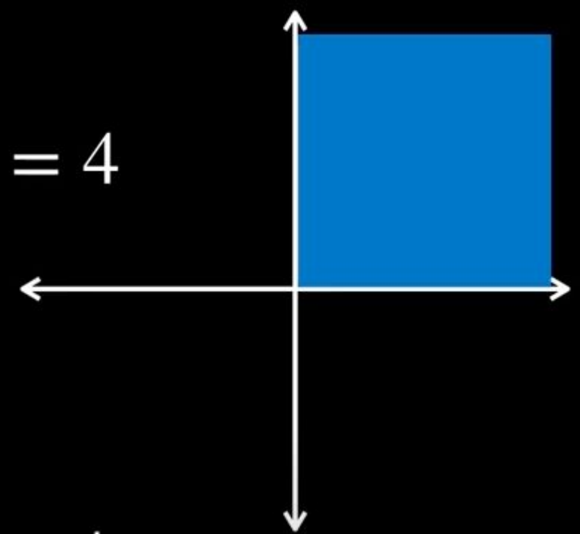
X

Example

$$J = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$\det(J) = 2 \cdot 2 - 0 \cdot 0 = 4$$





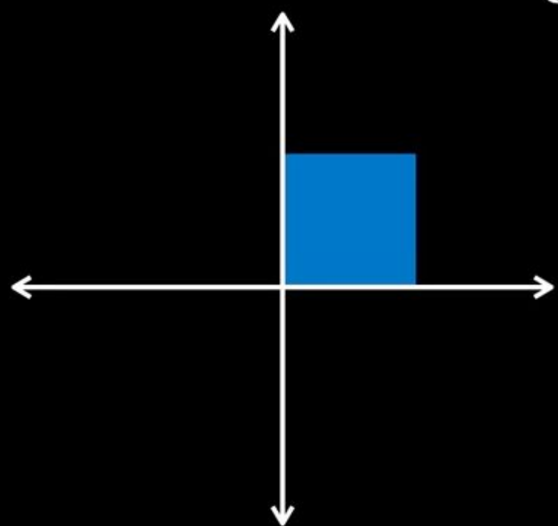
Z

$$p(x) = p(z) \left| \frac{1}{\det\left(\frac{\partial x}{\partial z}\right)} \right|$$

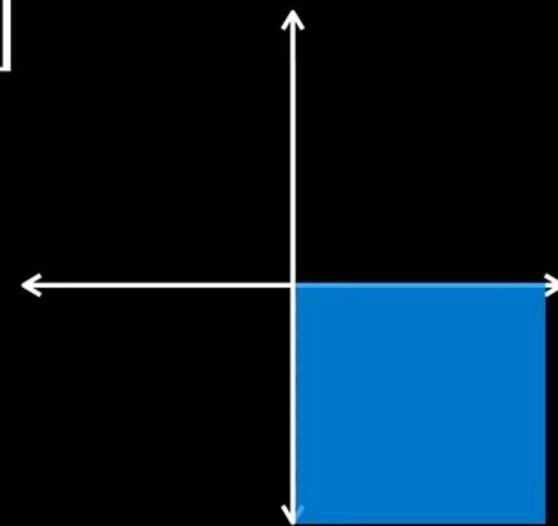
X

Example

$$J = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$



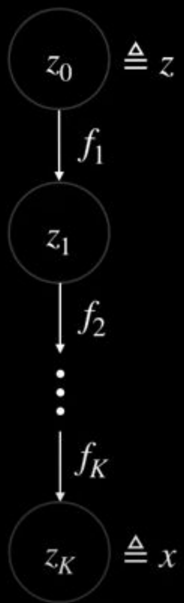
Z



X

$$p(x) = p(z)\left|\frac{1}{\det\left(\frac{\partial x}{\partial z}\right)}\right|$$

Normalizing flow



"normalizing flow"

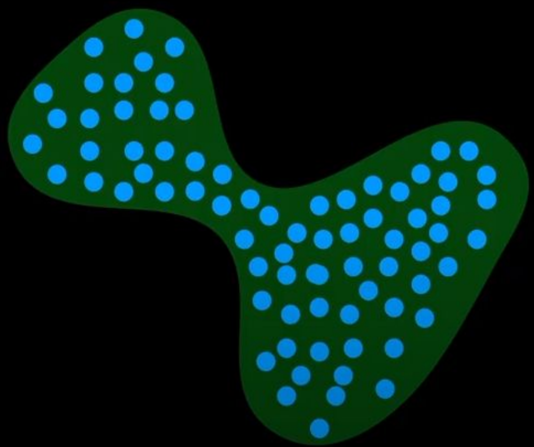
$$z \sim p_\theta(z)$$

$$x = f_\theta(z) = f_K \circ \ldots \circ f_2 \circ f_1(z)$$

$$p_\theta(x) = p_\theta(z) \prod_1^K \left| \det\left(\frac{\partial f_i^{-1}}{\partial z_i} \right) \right| = p_\theta(z) \left| \det\left(\frac{\partial f^{-1}}{\partial x} \right) \right|$$

Applications

$p_D(x)$



$p_\theta(x)$

$p_\theta(x \mid z)$ (likelihood)

latent variables

$z \longrightarrow$ 

$z \longrightarrow$ 

$z \longrightarrow$ 

$\wr$

$p_\theta(z)$ (prior)

observed variables

$p_\theta(x)$

x 

x 

x 

$\longrightarrow$ 

$p_\theta($  $)$ vs. $p_\theta($  $)$

How to use normalizing flows?

$$z \sim p_\theta(z)$$

$$x = f_\theta(z) = f_K \circ \ldots \circ f_2 \circ f_1(z)$$

$$p_\theta(x) = p_\theta(z) \left| \det\left(\frac{\partial f^{-1}}{\partial x} \right) \right|$$

How to use normalizing flows?

$$p_\theta(x) = p_\theta(z)\left|\det\left(\frac{\partial f^{-1}}{\partial x}\right)\right|$$

$$\log p_\theta(x) = \log p_\theta(z) + \log\left|\det\left(\frac{\partial f^{-1}}{\partial x}\right)\right|$$

$$= \log p_\theta(z) + \sum_{i=1}^{K}\log\left|\det\left(\frac{\partial f_i^{-1}}{\partial z_i}\right)\right|$$

How to use normalizing flows?

$$\log p_\theta(x) = \log p_\theta(z) + \sum_{i=1}^{K} \log \left| \det\left(\frac{\partial f_i^{-1}}{\partial z_i} \right) \right|$$

exact log-likelihood evaluation

exact posterior inference (via $z = f^{-1}(x)$)

Jacobian determinant computations are expensive

$$\log p_\theta(x) = \log p_\theta(z) + \sum_{i=1}^{K} \log \left| \det\left(\frac{\partial f_i^{-1}}{\partial z_i} \right) \right|$$

$$\begin{bmatrix} a_{1,1} & 0 & \cdots & \cdots & 0 \\ a_{2,1} & a_{2,2} & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ a_{n,1} & a_{n,2} & \cdots & \cdots & a_{n,n} \end{bmatrix}$$

Accepted as a workshop contribution at ICLR 2015

NICE: NON-LINEAR INDEPENDENT COMPONENTS ESTIMATION

Laurent Dinh David Krueger Yoshua Bengio*
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

arXiv:1410.8516v6 [cs.LG] 10 Apr 2015

ABSTRACT

We propose a deep learning framework for modeling complex high-dimensional densities called Non-linear Independent Component Estimation (NICE). It is based on the idea that a good representation is one in which the data has a distribution that is easy to model. For this purpose, a non-linear deterministic transformation of the data is learned that maps it to a latent space so as to make the transformed data conform to a factorized distribution, i.e., resulting in independent latent variables. We parametrize this transformation so that computing the determinant of the Jacobian and inverse Jacobian is trivial, yet we maintain the ability to learn complex non-linear transformations, via a composition of simple building blocks, each based on a deep neural network. The training criterion is simply the exact log-likelihood, which is tractable. Unbiased ancestral sampling is also easy. We show that this approach yields good generative models on four image datasets and can be used for inpainting.

1 INTRODUCTION

One of the central questions in unsupervised learning is how to capture complex data distributions that have unknown structure. Deep learning approaches (Bengio, 2009) rely on the learning of a representation of the data that would capture its most important factors of variation. This raises the question: *what is a good representation?* Like in recent work (Kingma and Welling, 2014; Rezende et al., 2014; Ozair and Bengio, 2014), we take the view that *a good representation is one in which the distribution of the data is easy to model*. In this paper, we consider the special case where we ask the learner to find a transformation $h = f(x)$ of the data into a new space such that the resulting distribution *factorizes*, i.e., *the components h_d are independent*:

$$p_H(h) = \prod_d p_{H_d}(h_d).$$

The proposed training criterion is directly derived from the log-likelihood. More specifically, we consider a change of variables $h = f(x)$, which assumes that f is invertible and the dimension of h is the same as the dimension of x, in order to fit a distribution p_H. The change of variable rule gives us:

$$p_X(x) = p_H(f(x))|\det \frac{\partial f(x)}{\partial x}|. \quad (1)$$

where $\frac{\partial f(x)}{\partial x}$ is the Jacobian matrix of function f at x. In this paper, we choose f such that the determinant of the Jacobian is trivially obtained. Moreover, its inverse f^{-1} is also trivially obtained, allowing us to sample from $p_X(x)$ easily as follows:

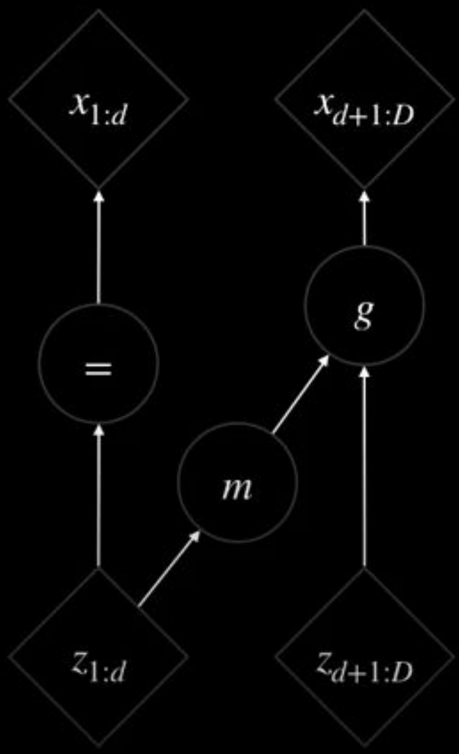
$$\begin{aligned} h &\sim p_H(h) \\ x &= f^{-1}(h) \end{aligned} \quad (2)$$

A key novelty of this paper is the design of such a transformation f that yields these two properties of "easy determinant of the Jacobian" and "easy inverse", while allowing us to have as much capacity

*Yoshua Bengio is a CIFAR Senior Fellow.

1

NICE - nonlinear independent components estimation
Coupling layers



$$x_{1:d} = z_{1:d}$$

$$x_{d+1:D} = g(z_{d+1:D}; m(z_{1:d}))$$

$$\frac{\partial x}{\partial z} = \begin{bmatrix} I_d & 0 \\ \frac{\partial x_{d+1:D}}{\partial z_{1:d}} & \frac{\partial x_{d+1:D}}{\partial z_{d+1:D}} \end{bmatrix}$$

Inversion of transformation

$$x_{1:d} = z_{1:d}$$

$$x_{d+1:D} = g(z_{d+1:D}; m(z_{1:d}))$$

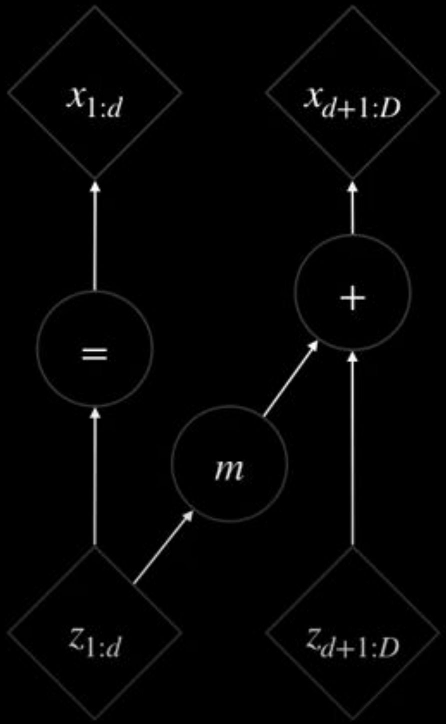
$$\Downarrow$$

$$z_{1:d} = x_{1:d}$$

$$z_{d+1:D} = g^{-1}(x_{d+1:D}; m(x_{1:d}))$$

Additive coupling layer

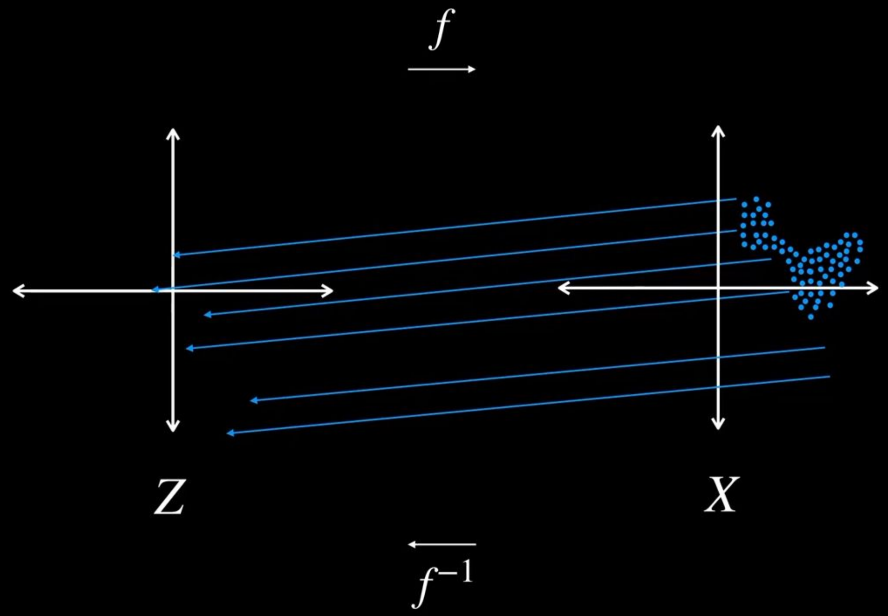
$$g(z_{d+1:D}; m(z_{1:d})) = z_{d+1:D} + m(z_{1:d})$$

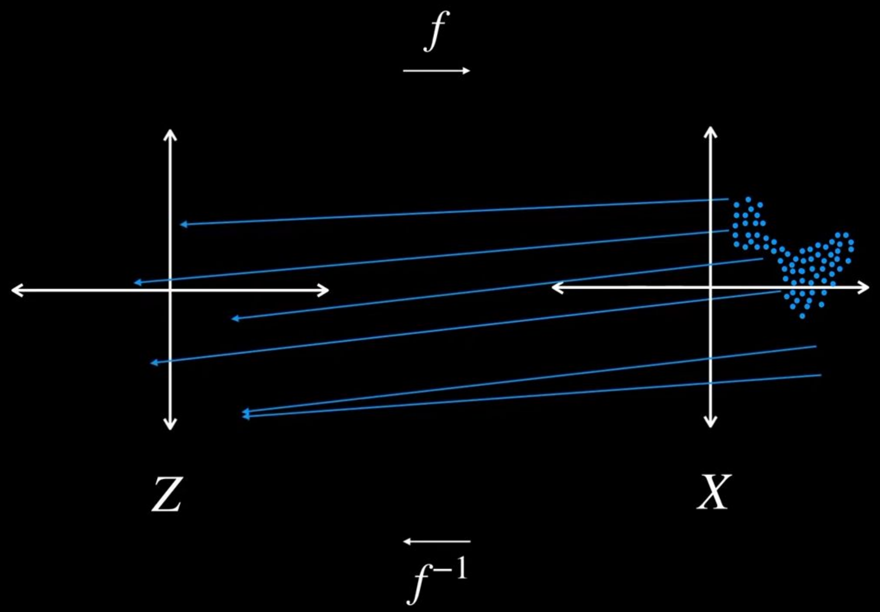


$$x_{1:d} = z_{1:d}$$

$$x_{d+1:D} = z_{d+1:D} + m(z_{1:d})$$

Limitations



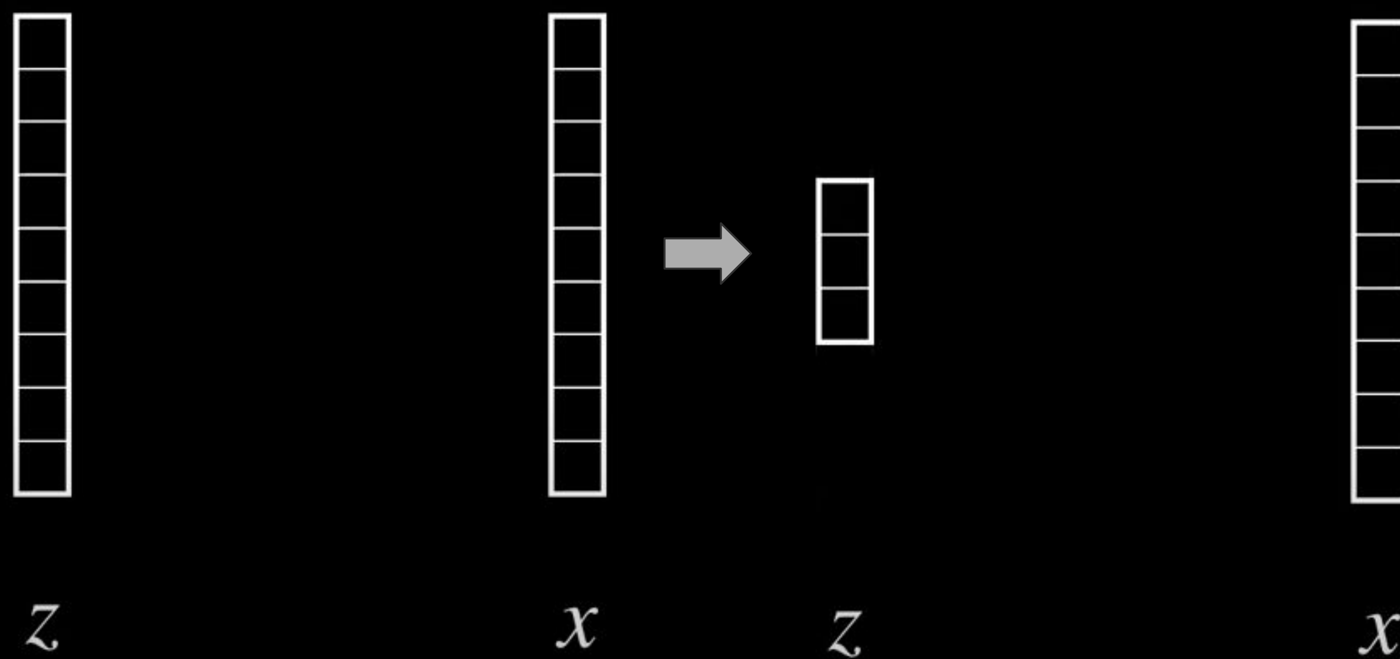


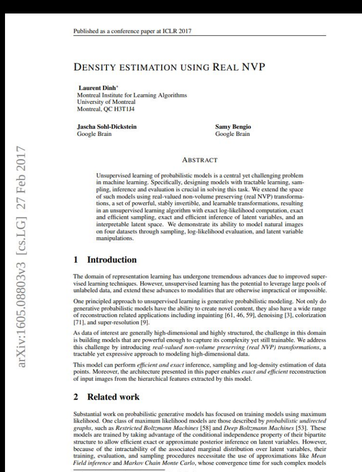
Scaling matrix

$$\begin{bmatrix} S_{1,1} & 0 & \dots & \dots & 0 \\ 0 & S_{2,2} & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & \dots & S_{D,D} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_D \end{bmatrix}$$

Scaling matrix S

$$\log p_\theta(x) = \sum_{i=1}^{D} \left[\log(p_\theta(f^{-1}(x)_i)) - \log(|S_{ii}|)\right]$$




Published as a conference paper at ICLR 2017

Density estimation using Real NVP

Laurent Dinh[*]
Montreal Institute for Learning Algorithms
University of Montreal
Montreal, QC H3T1J4

Jascha Sohl-Dickstein
Google Brain

Samy Bengio
Google Brain

arXiv:1605.08803v3 [cs.LG] 27 Feb 2017

Abstract

Unsupervised learning of probabilistic models is a central yet challenging problem in machine learning. Specifically, designing models with tractable learning, sampling, inference and evaluation is crucial in solving this task. We extend the space of such models using real-valued non-volume preserving (real NVP) transformations, a set of powerful, stably invertible, and learnable transformations, resulting in an unsupervised learning algorithm with exact log-likelihood computation, exact and efficient sampling, exact and efficient inference of latent variables, and an interpretable latent space. We demonstrate its ability to model natural images on four datasets through sampling, log-likelihood evaluation, and latent variable manipulations.

1 Introduction

The domain of representation learning has undergone tremendous advances due to improved supervised learning techniques. However, unsupervised learning has the potential to leverage large pools of unlabeled data, and extend these advances to modalities that are otherwise impractical or impossible.

One principled approach to unsupervised learning is generative probabilistic modeling. Not only do generative probabilistic models have the ability to create novel content, they also have a wide range of reconstruction related applications including inpainting [61, 46, 59], denoising [3], colorization [71], and super-resolution [9].

As data of interest are generally high-dimensional and highly structured, the challenge in this domain is building models that are powerful enough to capture its complexity yet still trainable. We address this challenge by introducing *real-valued non-volume preserving (real NVP) transformations*, a tractable yet expressive approach to modeling high-dimensional data.

This model can perform *efficient and exact* inference, sampling and log-density estimation of data points. Moreover, the architecture presented in this paper enables *exact and efficient* reconstruction of input images from the hierarchical features extracted by this model.

2 Related work

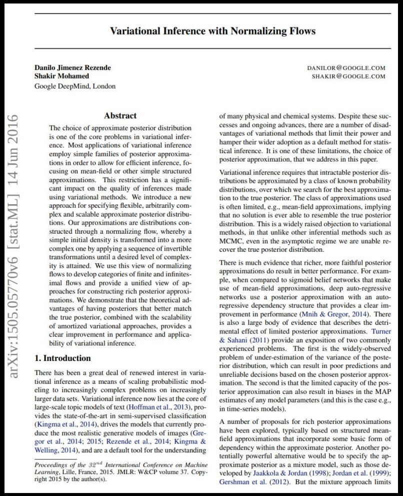
Substantial work on probabilistic generative models has focused on training models using maximum likelihood. One class of maximum likelihood models are those described by *probabilistic undirected graphs*, such as *Restricted Boltzmann Machines* [58] and *Deep Boltzmann Machines* [53]. These models are trained by taking advantage of the conditional independence property of their bipartite structure to allow efficient exact or approximate posterior inference on latent variables. However, because of the intractability of the associated marginal distribution over latent variables, their training, evaluation, and sampling procedures necessitate the use of approximations like *Mean Field inference* and *Markov Chain Monte Carlo*, whose convergence time for such complex models

[*]Work was done when author was at Google Brain.

$$z = (z^{(1)}, \ldots, z^{(L)})$$

each $z^{(i)}$ operates at a different scale

Other applications



Variational Inference with Normalizing Flows

Danilo Jimenez Rezende DANILOR@GOOGLE.COM
Shakir Mohamed SHAKIR@GOOGLE.COM
Google DeepMind, London

arXiv:1505.05770v6 [stat.ML] 14 Jun 2016

Abstract

The choice of approximate posterior distribution is one of the core problems in variational inference. Most applications of variational inference employ simple families of posterior approximations in order to allow for efficient inference, focusing on mean-field or other simple structured approximations. This restriction has a significant impact on the quality of inferences made using variational methods. We introduce a new approach for specifying flexible, arbitrarily complex and scalable approximate posterior distributions. Our approximations are distributions constructed through a normalizing flow, whereby a simple initial density is transformed into a more complex one by applying a sequence of invertible transformations until a desired level of complexity is attained. We use this view of normalizing flows to develop categories of finite and infinitesimal flows and provide a unified view of approaches for constructing rich posterior approximations. We demonstrate that the theoretical advantages of having posteriors that better match the true posterior, combined with the scalability of amortized variational approaches, provides a clear improvement in performance and applicability of variational inference.

1. Introduction

There has been a great deal of renewed interest in variational inference as a means of scaling probabilistic modeling to increasingly complex problems on increasingly larger data sets. Variational inference now lies at the core of large-scale topic models of text (Hoffman et al., 2013), provides the state-of-the-art in semi-supervised classification (Kingma et al., 2014), drives the models that currently produce the most realistic generative models of images (Gregor et al., 2014; 2015; Rezende et al., 2014; Kingma & Welling, 2014), and are a default tool for the understanding of many physical and chemical systems. Despite these successes and ongoing advances, there are a number of disadvantages of variational methods that limit their power and hamper their wider adoption as a default method for statistical inference. It is one of these limitations, the choice of posterior approximation, that we address in this paper.

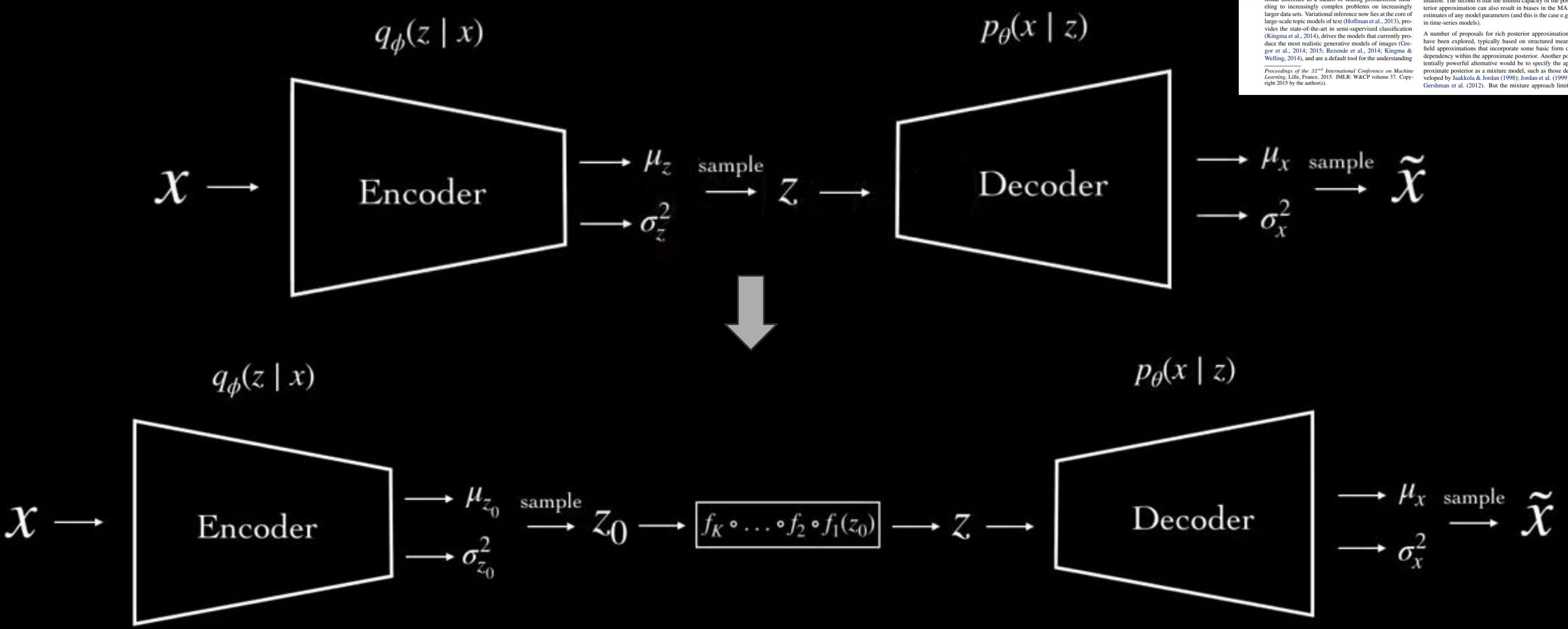
Variational inference requires that intractable posterior distributions be approximated by a class of known probability distributions, over which we search for the best approximation to the true posterior. The class of approximations used is often limited, e.g., mean-field approximations, implying that no solution is ever able to resemble the true posterior distribution. This is a widely raised objection to variational methods, in that unlike other inferential methods such as MCMC, even in the asymptotic regime we are unable recover the true posterior distribution.

There is much evidence that richer, more faithful posterior approximations do result in better performance. For example, when compared to sigmoid belief networks that make use of mean-field approximations, deep auto-regressive networks use a posterior approximation with an auto-regressive dependency structure that provides a clear improvement in performance (Mnih & Gregor, 2014). There is also a large body of evidence that describes the detrimental effect of limited posterior approximations. Turner & Sahani (2011) provide an exposition of two commonly experienced problems. The first is the widely-observed problem of under-estimation of the variance of the posterior distribution, which can result in poor predictions and unreliable decisions based on the chosen posterior approximation. The second is that the limited capacity of the posterior approximation can also result in biases in the MAP estimates of any model parameters (and this is the case e.g., in time-series models).

A number of proposals for rich posterior approximations have been explored, typically based on structured mean-field approximations that incorporate some basic form of dependency within the approximate posterior. Another potentially powerful alternative would be to specify the approximate posterior as a mixture model, such as those developed by Jaakkola & Jordan (1998); Jordan et al. (1999); Gershman et al. (2012). But the mixture approach limits

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

$$\underline{p_\theta(x \mid z)}\,p_\theta(z)$$



Summary

Normalizing flows:

- handle complex probability distributions
- there are some tricks to make it computationally tractable
- only a few of the ways of using them shown in the presentation