CS 480/680
Introduction to Machine Learning

Lecture 18
Variational Autoencoders and Normalizing Flows
Deep Generative Models Part I

Kathryn Simone

19 November 2024

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

Two significant and distinct goals in machine learning

Discriminative Model:
Learn a predictor given the observations.

Examples:
Perceptron, Support Vector Machines
Decision Trees, MLPs, CNNs

$$y = f(x)$$

Generative Model:
Describe the process that generated the data.

Examples:
KDE, GMMs

$$x \sim p(x)$$

Given $X_1, \ldots, X_n \sim D$, can we generate $X_{n+1}, X_{n+2}, \ldots$?

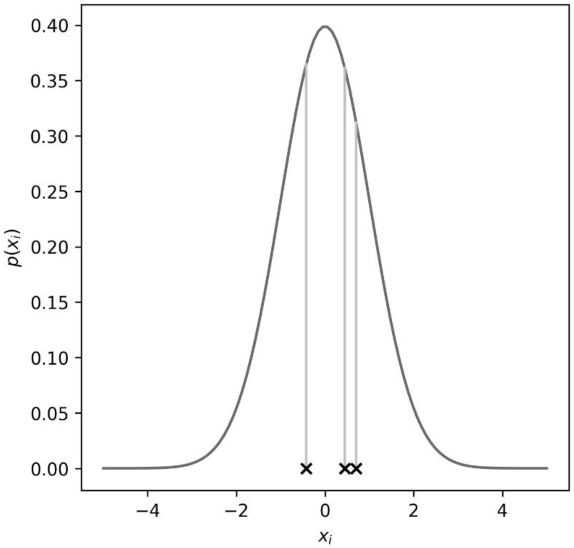
We have encountered a few generative models

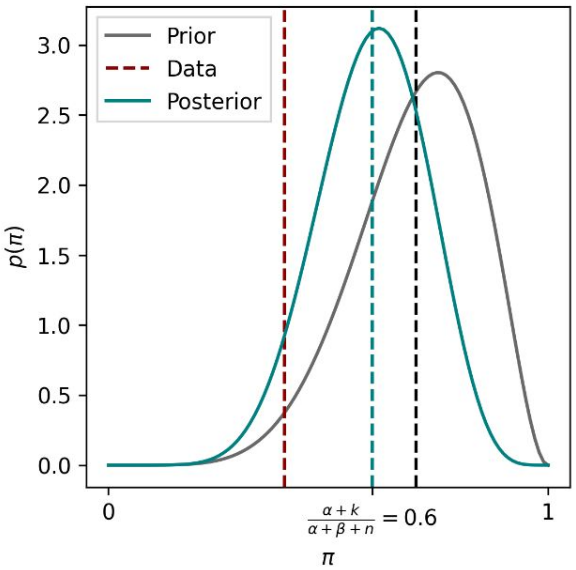
$$p(\boldsymbol{x} \mid \mu, \sigma^2) = \prod_{i=1}^{n} p(x_i \mid \mu, \sigma^2)$$
$$= \prod_{i=1}^{n} \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\sigma^2}(x_i - \mu)^2}.$$

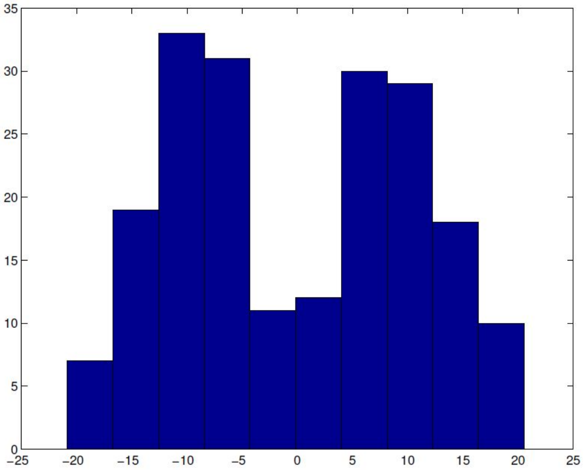
$$p(\pi \mid \boldsymbol{y}) \propto p(\boldsymbol{y} \mid \pi) p(\pi)$$

$$p(x) = (1-\pi)\mathcal{N}_{\mu_1, \sigma_1^2}(x) + \pi \mathcal{N}_{\mu_2, \sigma_2^2}(x)$$

$$\ell^t(\theta) \geq \sum_n \Big[-D_{\mathrm{KL}}\left(q_n(z_n) \,\|\, p(z_n \mid y_n, \theta)\right) + \log p(y_n \mid \theta) \Big]$$



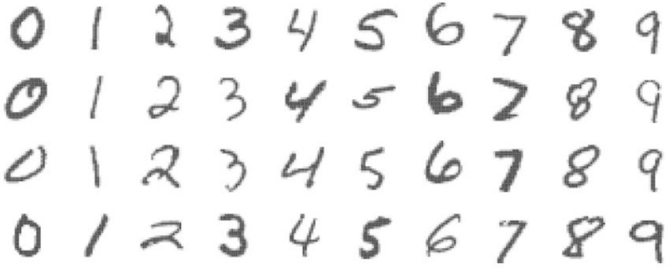




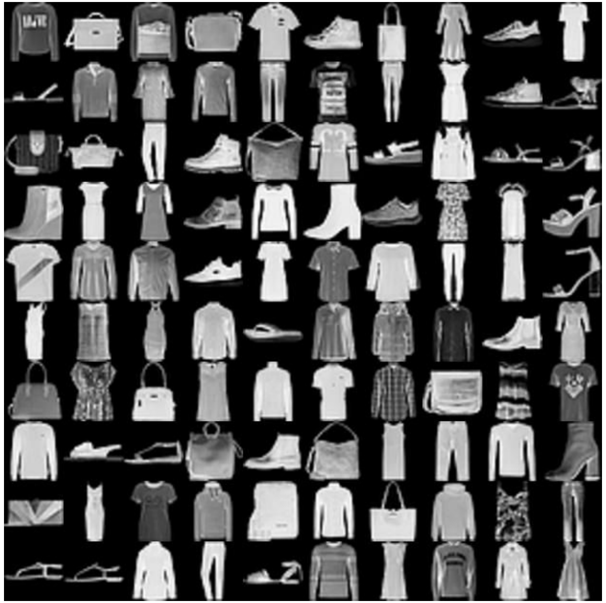
Left: Lecture 3; Middle: Lecture 9; Right: Lecture 12

Fitting a probability distribution to real-world data is hard

MNIST



Fashion MNIST



Probabilistic modeling is nevertheless essential

Conditional generative model:

$$p(x|c)$$

Examples:

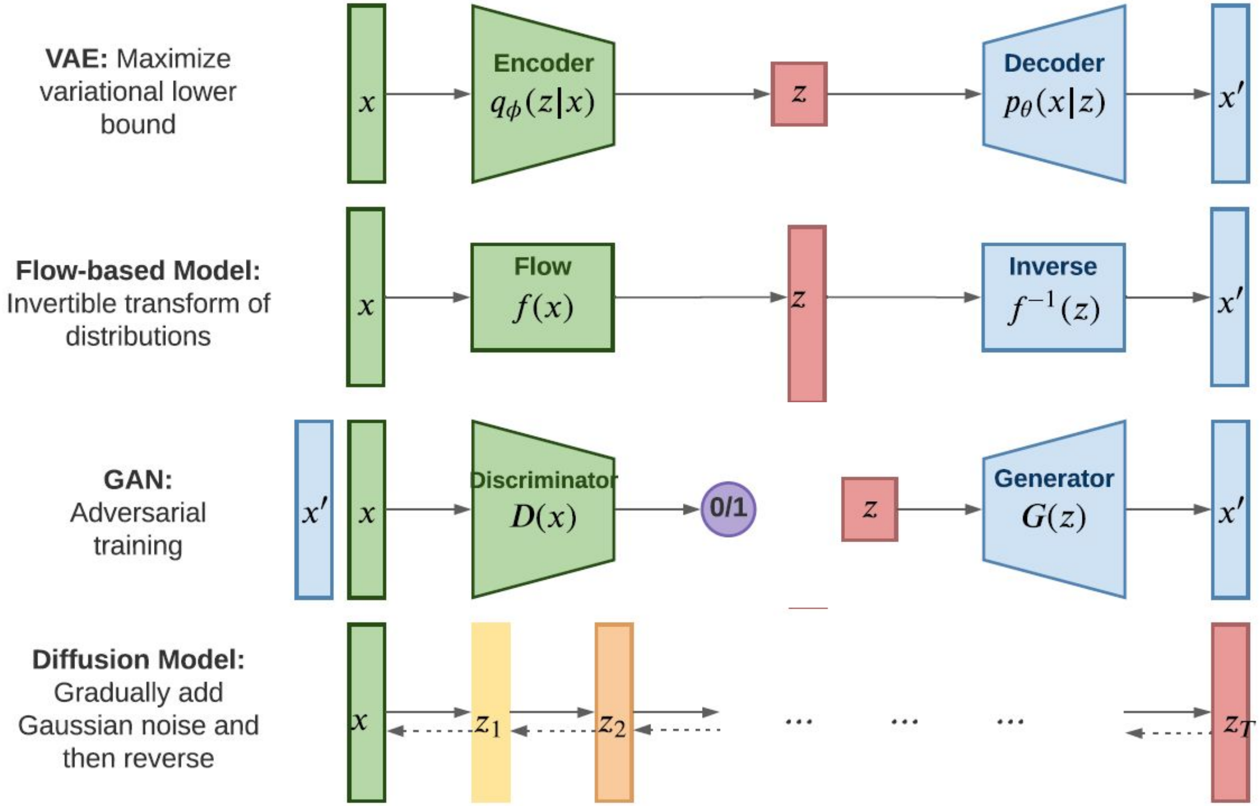
- c = image, x = text
- c = initial prompt, x = continuation
- c = text prompt, x = image
- c = image, x = image

(a) Teddy bears swimming at the Olympics 400m Butterfly event.

(b) A cute corgi lives in a house made out of sushi.

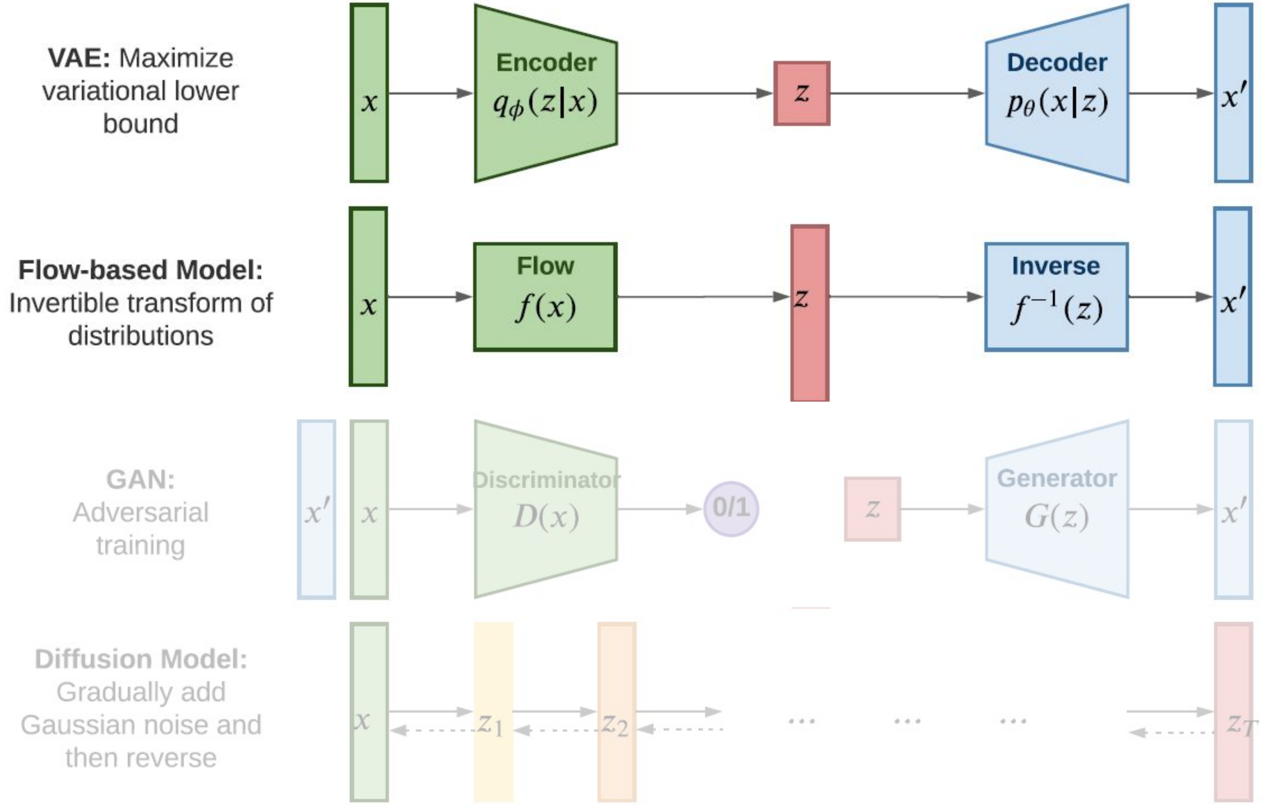
(c) A cute sloth holding a small treasure chest. A bright golden glow is coming from the chest.

Deep generative models use neural networks to learn a parametrized representation of the distribution



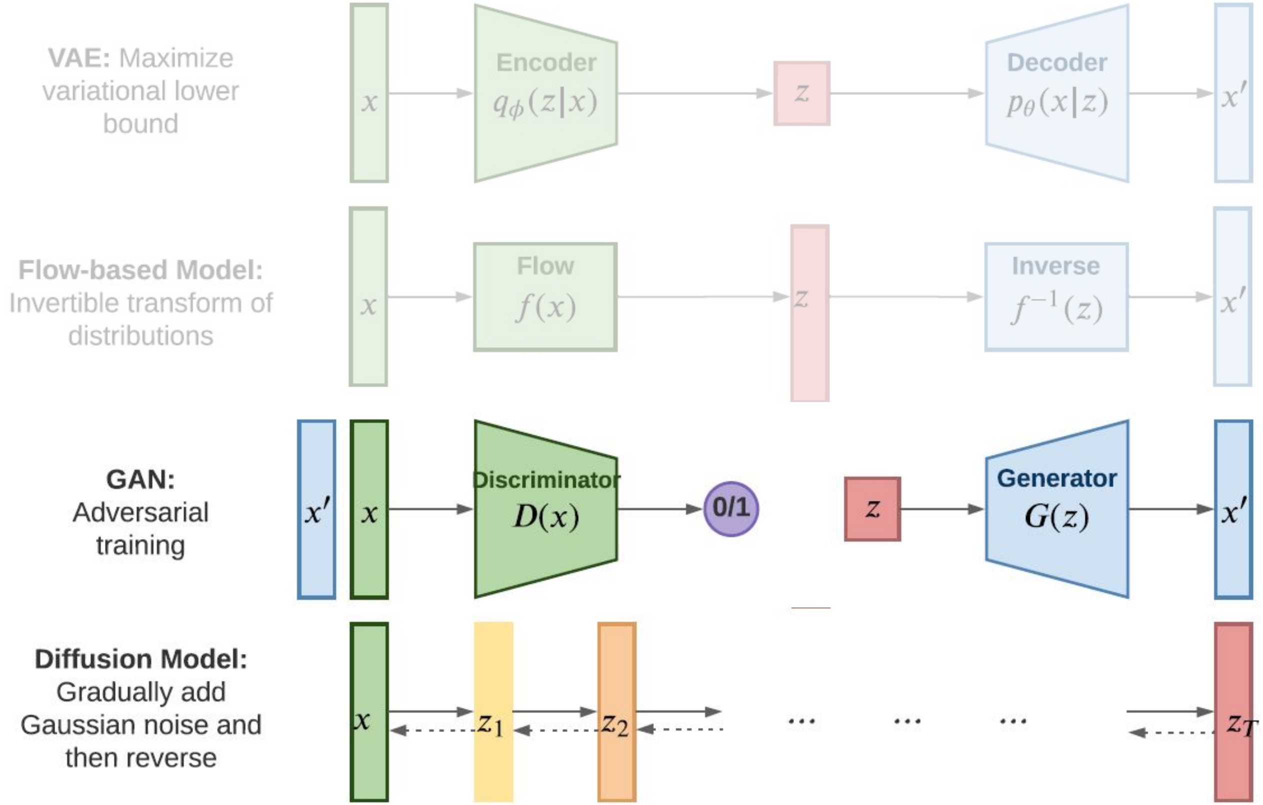
Probabilistic Machine Learning: Section 20.2

Deep Generative Models: Part I



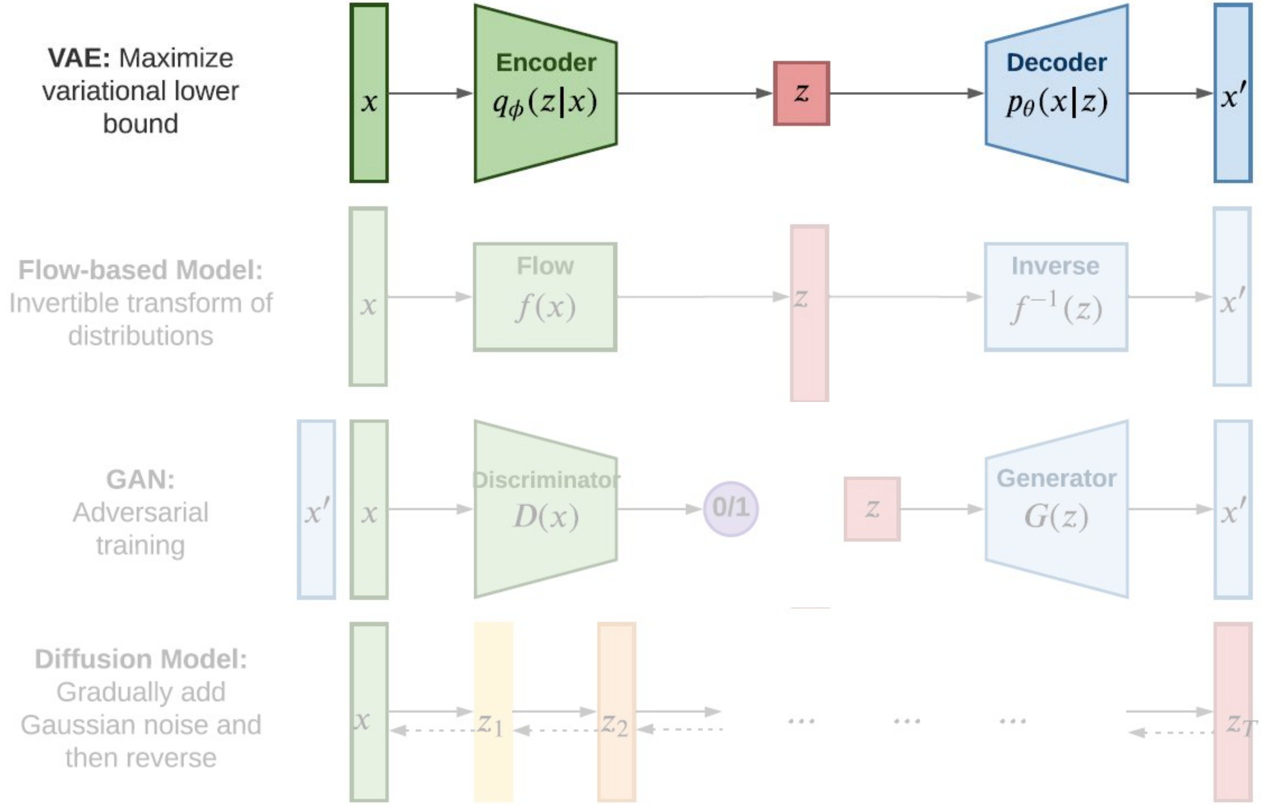
Probabilistic Machine Learning: Section 20.2

Deep Generative Models: Part II



Probabilistic Machine Learning: Section 20.2

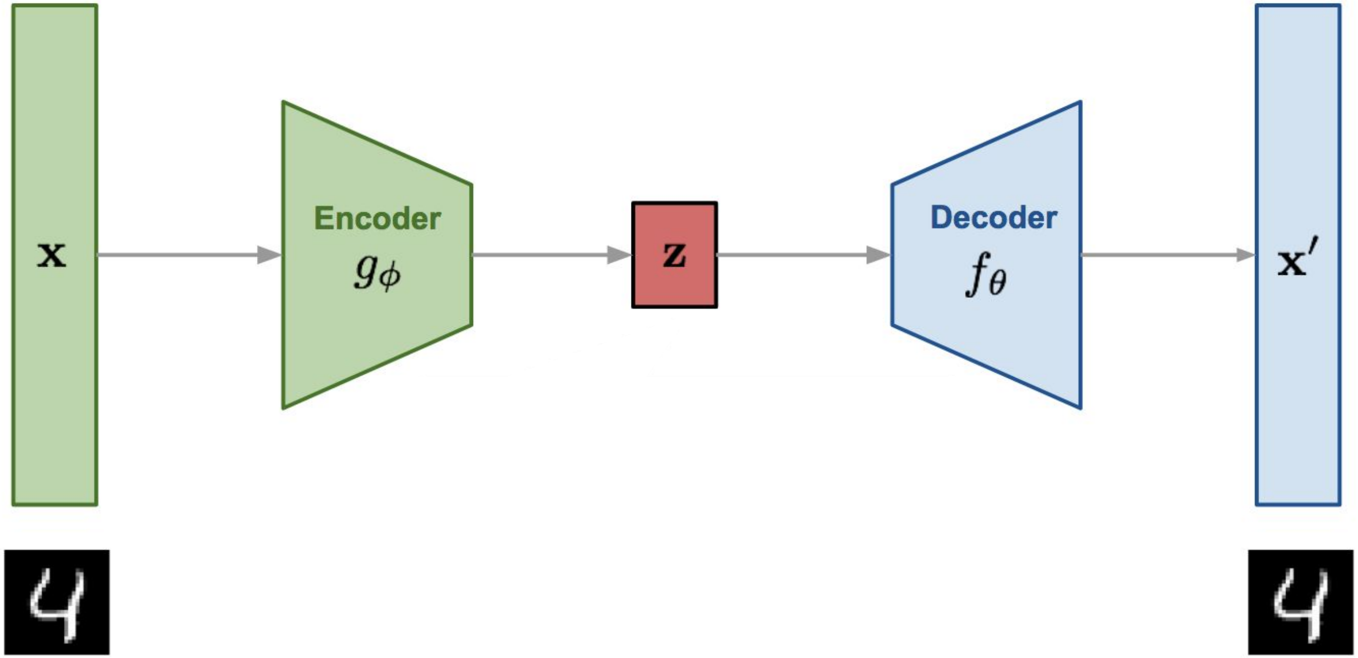
Variational Autoencoders



Probabilistic Machine Learning: Section 20.2

An autoencoder is a feedforward neural network trained to learn the identity function

$$\mathbf{z} = g_\phi(\mathbf{x}) \qquad \mathbf{x}' = f_\theta(g_\phi(\mathbf{x}))$$



https://lilianweng.github.io/posts/2018-08-12-vae/

Autoencoders minimize reconstruction loss

Linear:

$$W_f, W_g = \operatorname{argmin} \frac{1}{2} \sum_{i=1}^{n} \|W_g W_f x_i - x_i\|_2^2$$

Nonlinear:

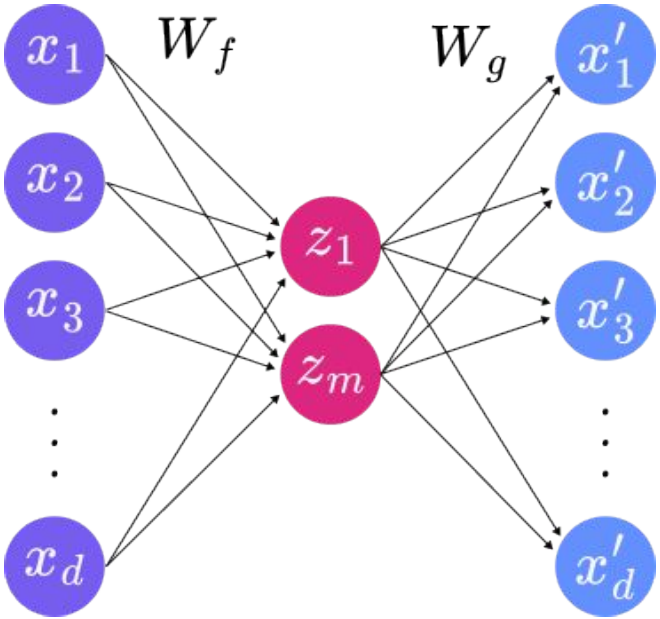
$$W_f, W_g = \operatorname{argmin} \sum_{i=1}^{n} \frac{1}{2} \|g(f(x_i)) - x_i\|_2^2$$

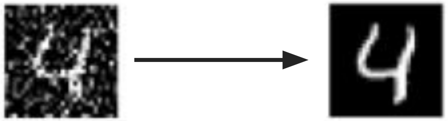
Sparse:

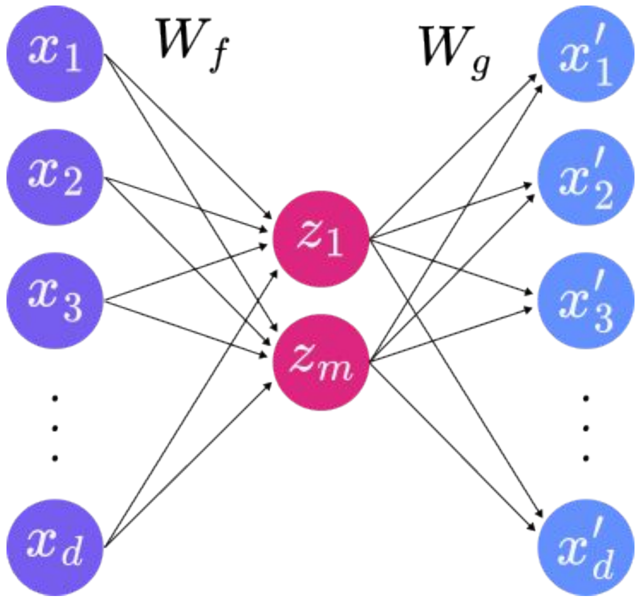
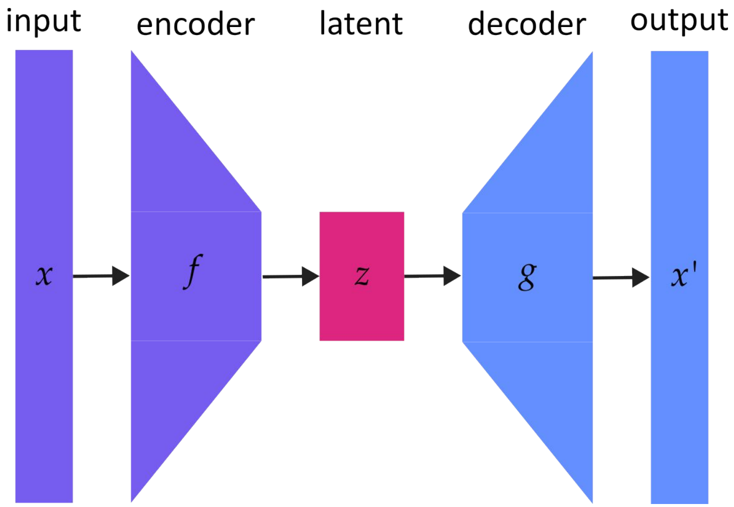
$$W_f, W_g = \operatorname{argmin} \sum_{i=1}^{n} \frac{1}{2} \|g(f(x_i)) - x_i\|_2^2 + \lambda \|f(x_i)\|_1$$

Denoising:

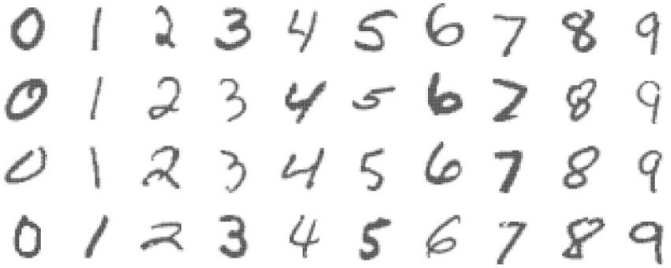
$$W_f, W_g = \operatorname{argmin} \sum_{i=1}^{n} \frac{1}{2} \|g(f(\tilde{x}_i)) - x_i\|_2^2$$

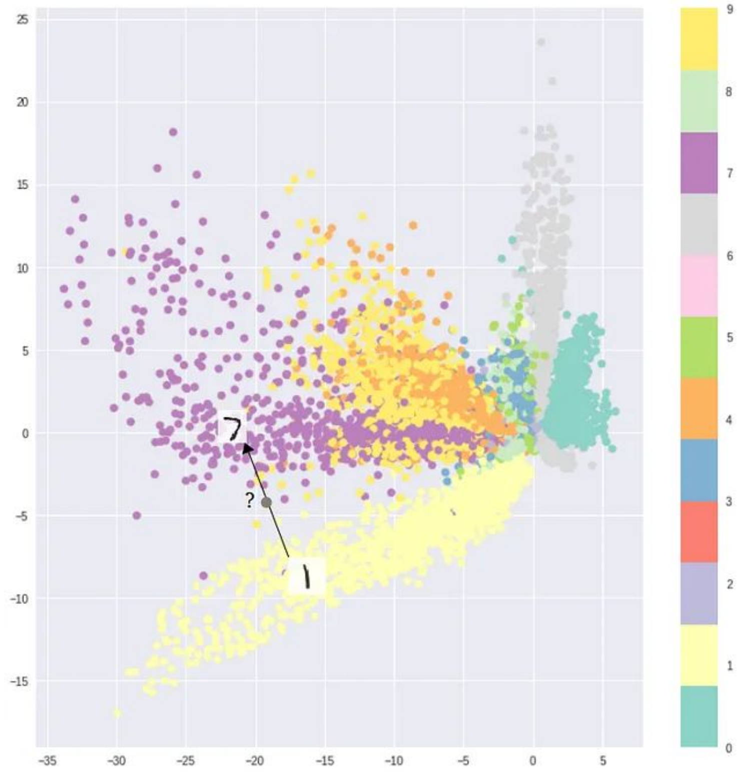





W_f
W_g
x_1
x_2
x_3
x_d
z_1
z_m
x'_1
x'_2
x'_3
x'_d

input
encoder
latent
decoder
output
x
f
z
g
x'

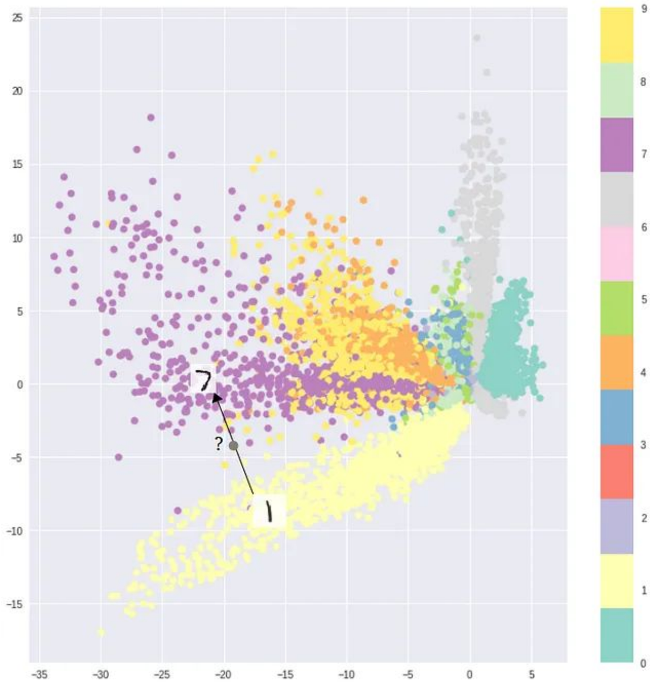
The autoencoder has discontinuities in its latent space

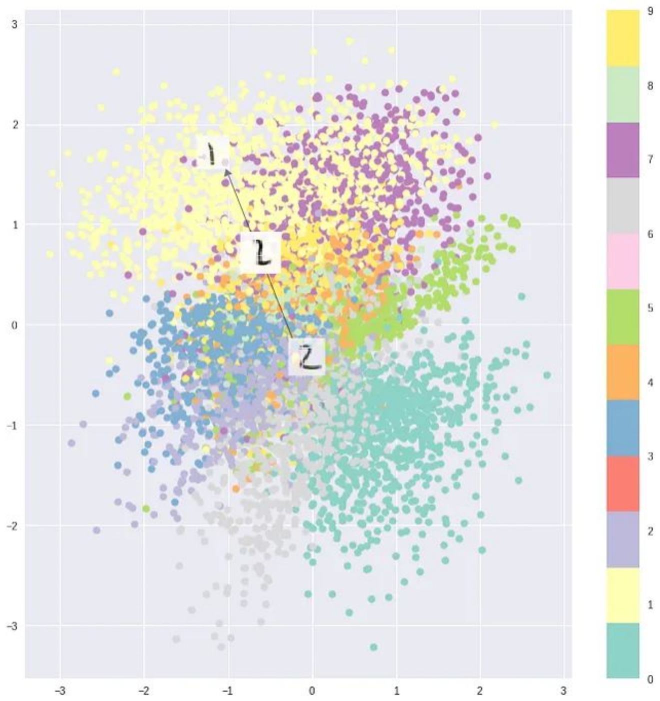




https://www.jeremyjordan.me/variational-autoencoders/

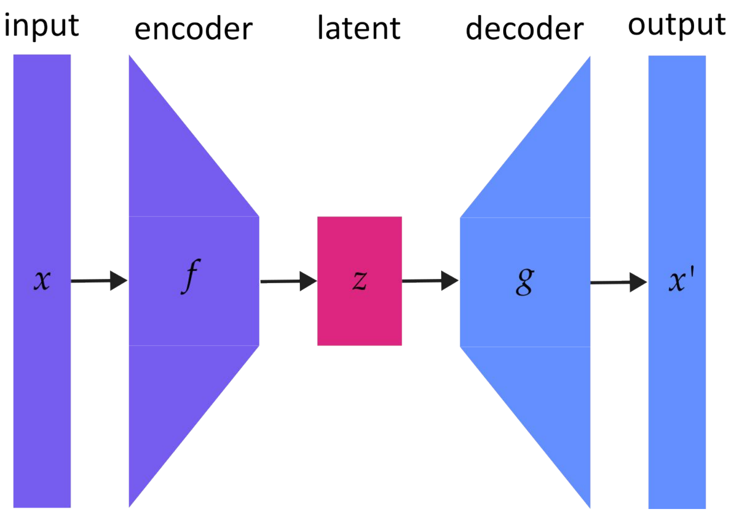
Variational autoencoders encourage continuity in the latent space

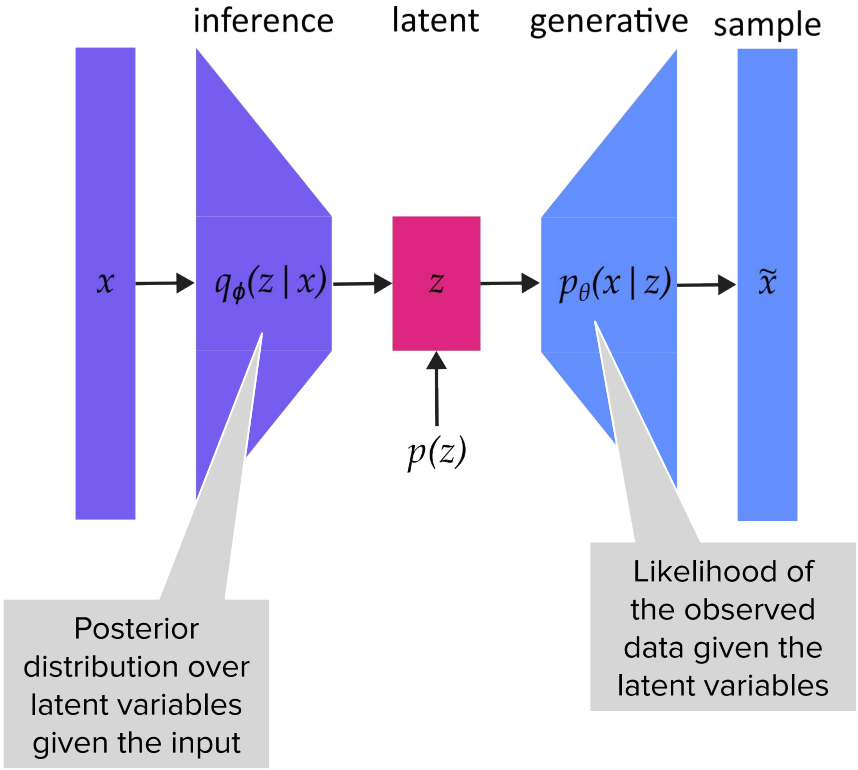




https://www.jeremyjordan.me/variational-autoencoders/

VAEs map an input to a distribution





The high-level architecture of a VAE

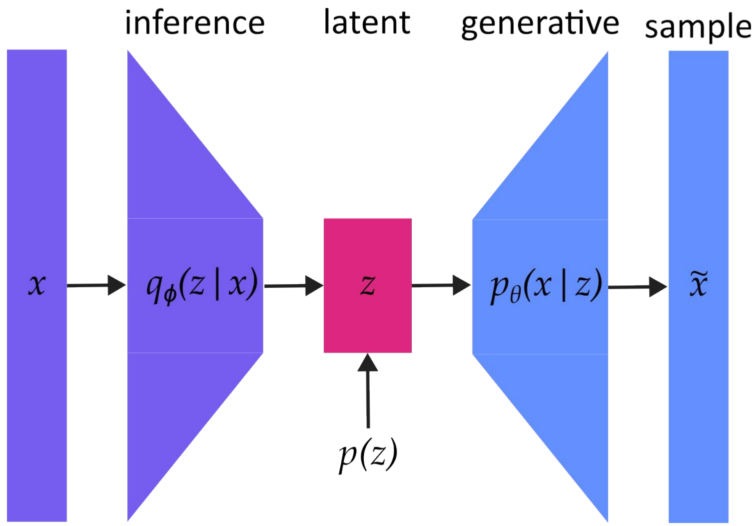
A VAE defines a generative model of the form

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

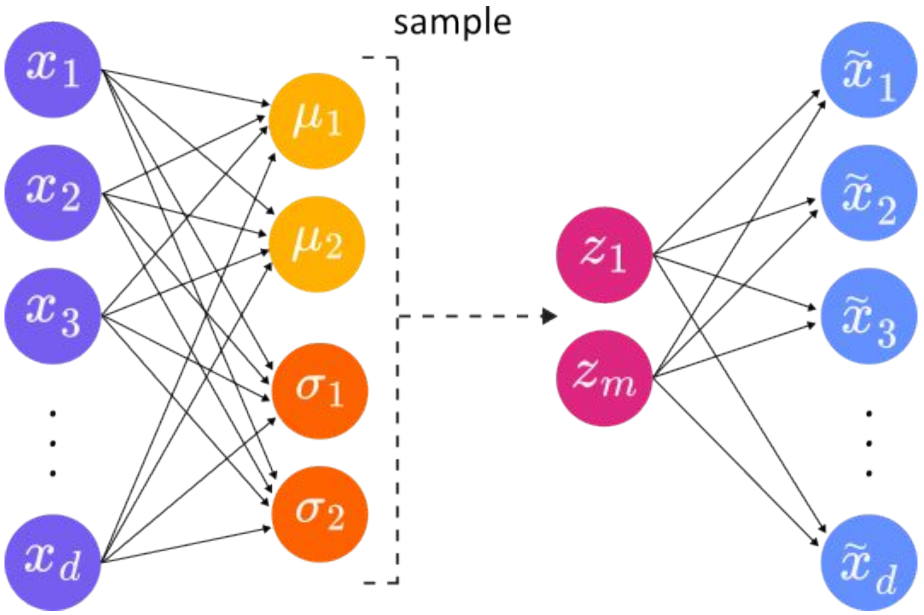
Where: $p_\theta(\mathbf{z})$: is the prior distribution over the latent variable $z \in \mathbb{R}^{\mathrm{m}}$, usually a Gaussian, and $p_\theta(\mathbf{x}|\mathbf{z})$: is the density of the decoder network's outputs, conditioned on latent vector.

A VAE approximate the posterior by fitting a "recognition" model:

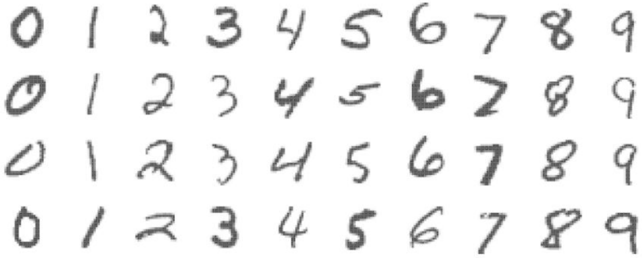
$$\begin{aligned} q_\phi(\mathbf{z}|\mathbf{x}) &= q(\mathbf{z}|e_\phi(\mathbf{x})) \\ &\approx p_\theta(\mathbf{z}|\mathbf{x}) \end{aligned}$$

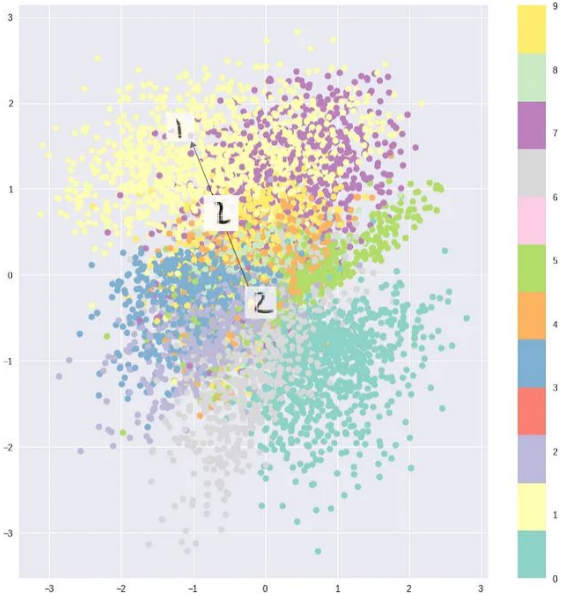


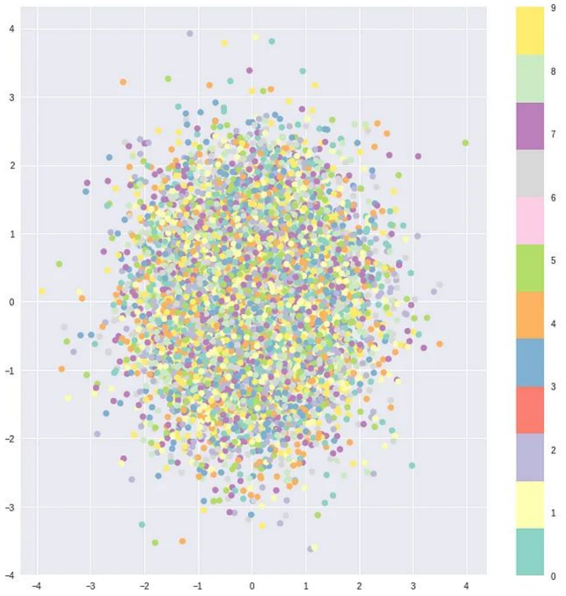
VAEs have both a probabilistic encoder and a probabilistic decoder



The VAE objective balances regularization and reconstruction







https://www.jeremyjordan.me/variational-autoencoders/

The VAE objective balances regularization and reconstruction

Encourage the input distribution to correspond to the latent distribution (regularization):

$$\min \mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})\right)$$
$$\Longrightarrow \min \mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) \| \mathcal{N}(0, I)\right)$$

Encourage accurate decoding (reconstruction):

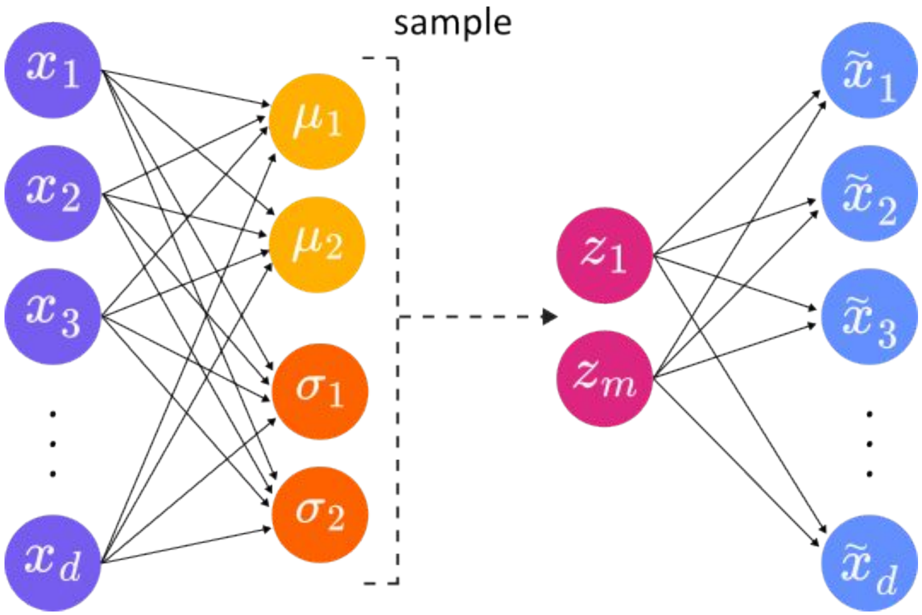
$$\max \mathbb{E}_{\mathbf{z}\sim q_\phi(\cdot|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]$$

It can be shown that these two quantities define the lower bound on the evidence $p_\theta(\mathbf{x})$:

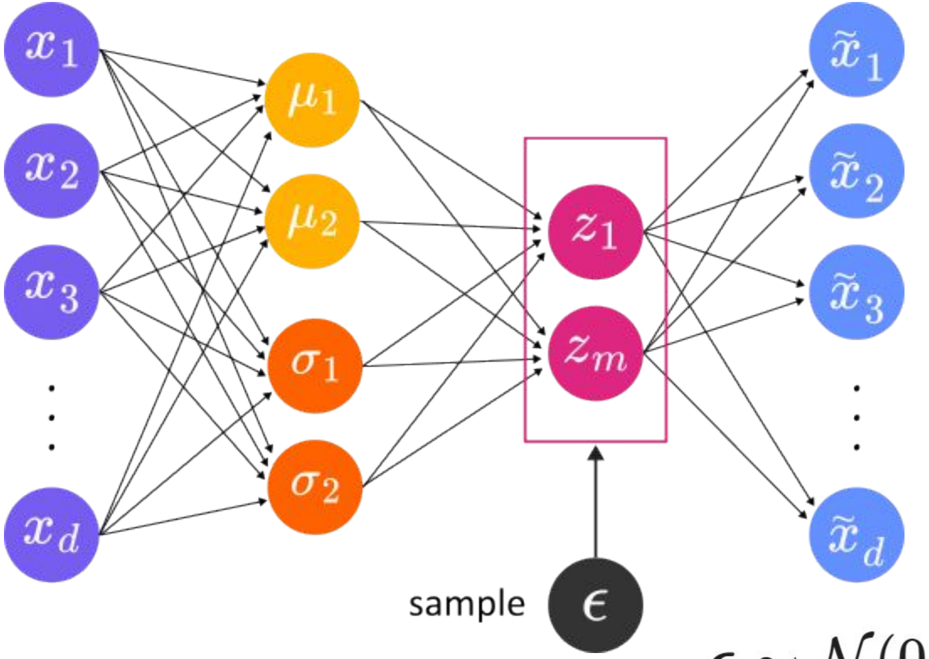
$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - KL\left(q_\phi(\mathbf{z}|\mathbf{x}) \| N(0, I)\right)$$

Derivation: Probabilistic Machine Learning: Advanced Topics Section 21.2

How to compute gradients across random operations?



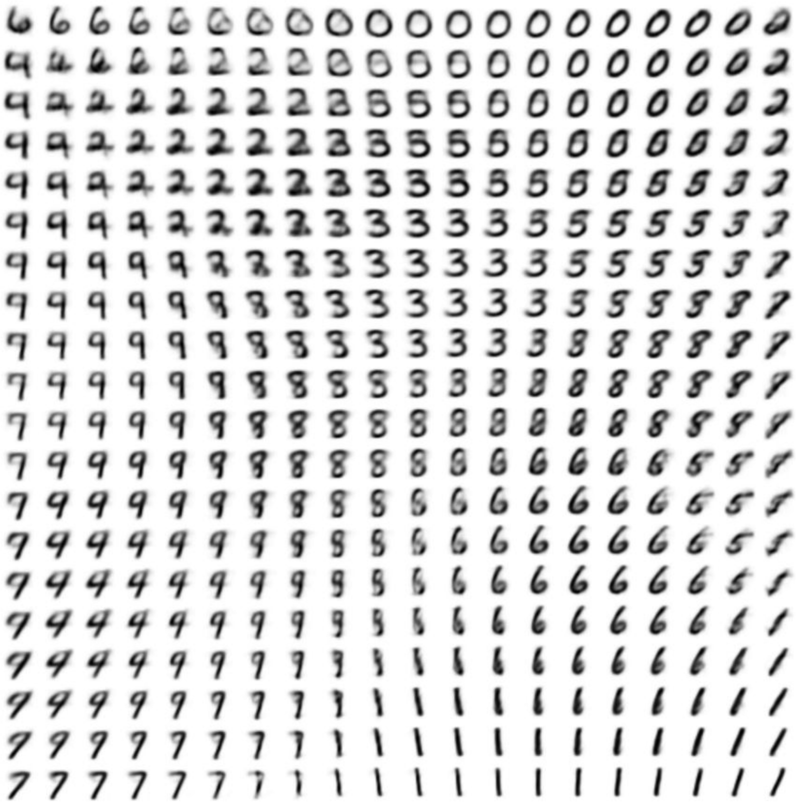
The "Reparametrization Trick"



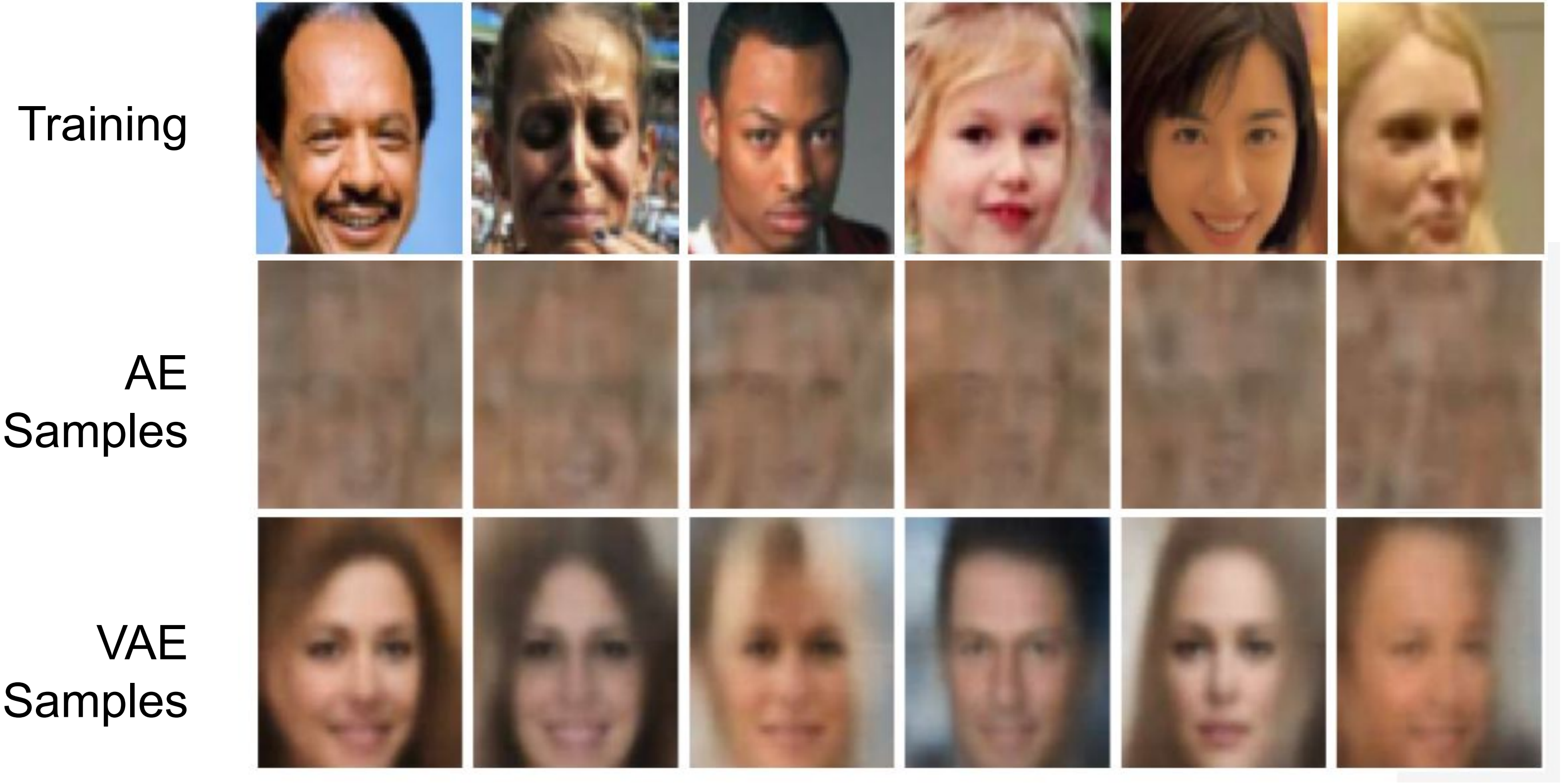
$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$$

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$$

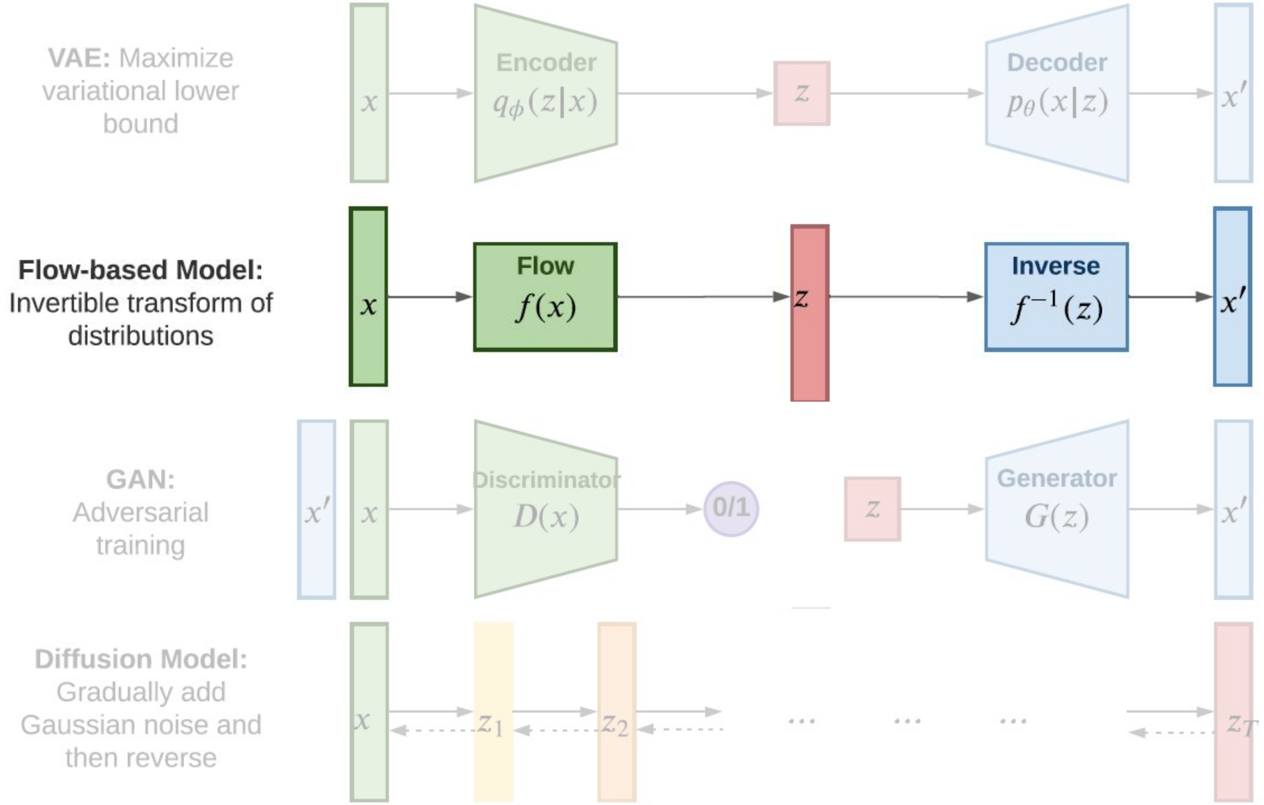
Interpolation with VAEs



AE and VAE on an unconditioned generation task

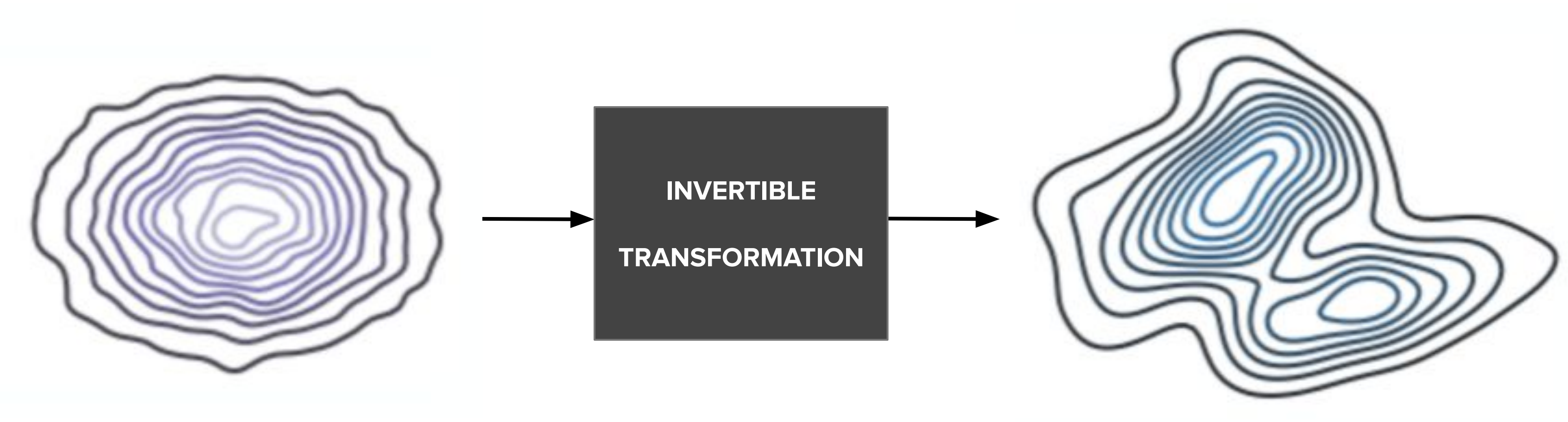


Normalizing Flows



Probabilistic Machine Learning: Section 20.2

Normalizing Flows



Transformations of random variables must preserve probability mass

Uniform random variable:

$$X \sim \text{Uniform}(0,1)$$

Transformation of X:

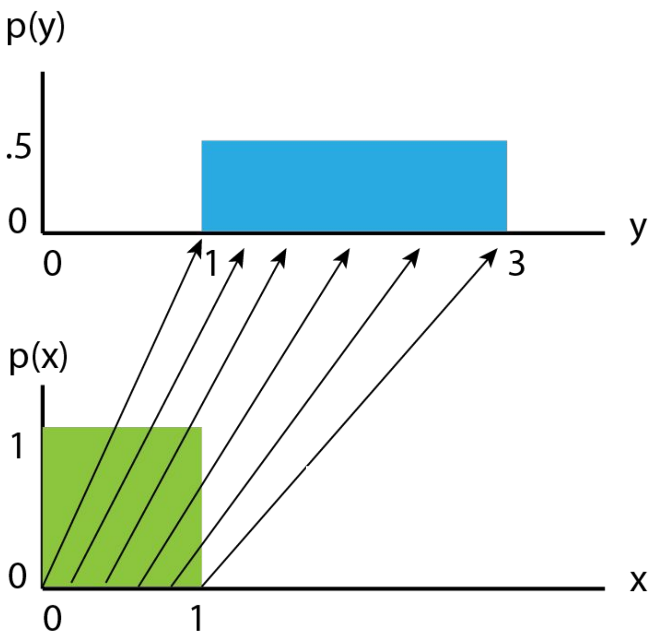
$$Y = f(X) = 2X + 1$$

Differentials:

$$(x, x+dx) \to (y, y+dy).$$

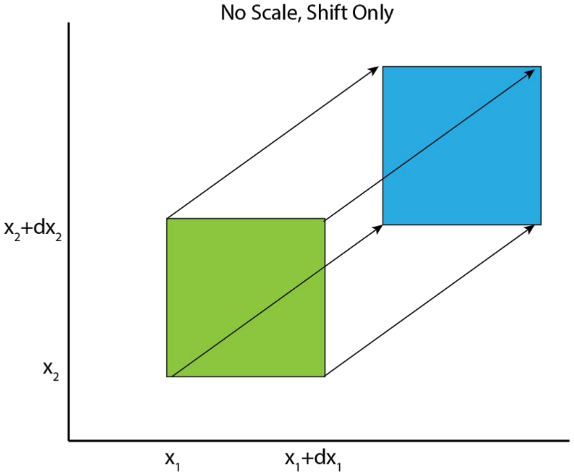
Relationship between probability densities:

$$p(x)dx = p(y)dy$$
$$p(y) = p(x)\left|\frac{dx}{dy}\right|$$



https://blog.evjang.com/2018/01/nf1.html

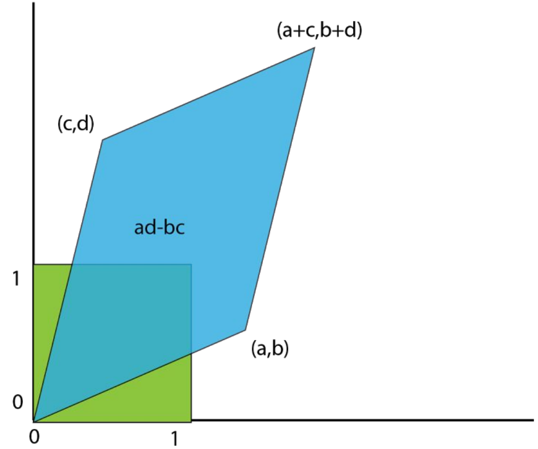
Linear transformations of multivariate distributions must be scaled by the determinant of the projection matrix



$$\boldsymbol{X} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \quad T = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

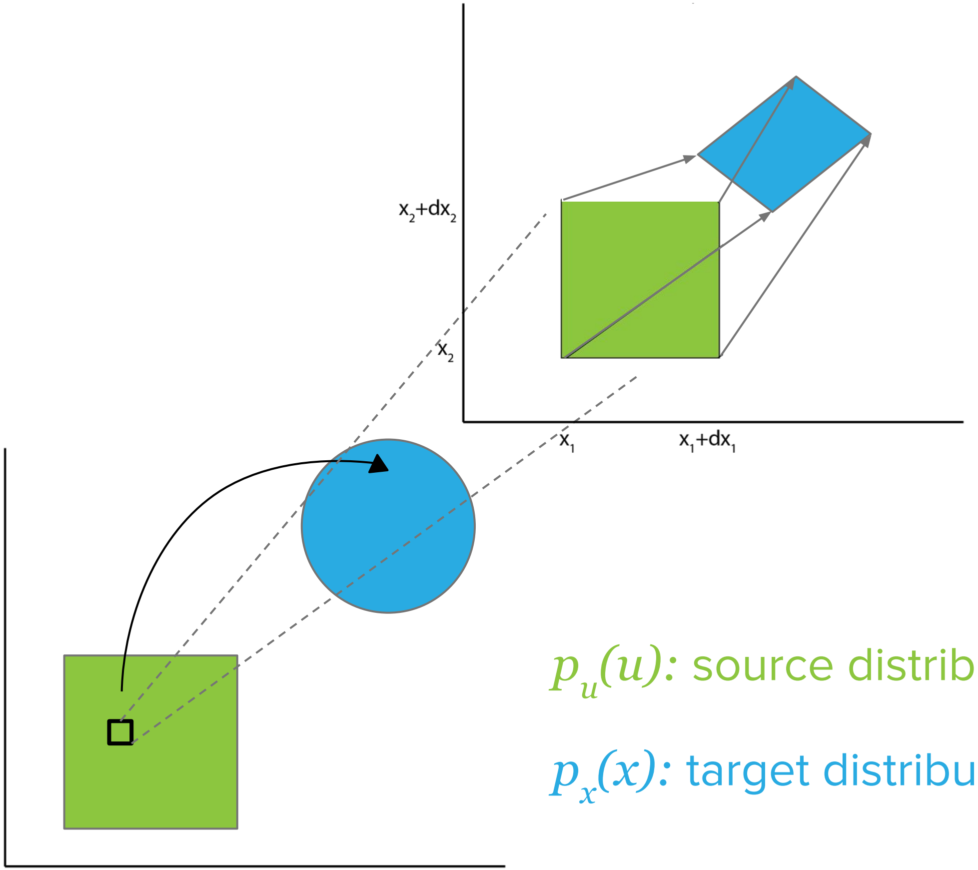
$$\begin{aligned} \boldsymbol{Y} &= T\boldsymbol{X} \\ &= \begin{bmatrix} 0 & 0 \\ a & b \\ c & d \\ a+c & b+d \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \int p(y)dy &= ad - bc \\ &= |\det(T)| \end{aligned}$$



https://blog.evjang.com/2018/01/nf1.html

Smooth nonlinear transformations are locally linear



$p_u(u)$: source distribution

$p_x(x)$: target distribution

Given a mapping $f : \mathbb{R}^d \mapsto \mathbb{R}^m$, the Jacobian matrix, defines all first-order partial derivatives:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_d} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_d} \end{bmatrix}$$

$$x = f(u), u = f^{-1}(x)$$

$$\int_{\mathcal{X}} p_x(x)dx = \int_{\mathcal{U}} p_u(u)du = 1$$

$$\implies p_x(x) = p_u(u)|\det \mathbf{J}(f)(u)|^{-1}$$

$$= p_u(f^{-1}(x))|\det \mathbf{J}(f^{-1})(x)|$$

Density estimation with an invertible transformation

Given a dataset $\mathcal{D} = \{x_1, x_2, \ldots, x_n\} \sim p_x(x)$, where $p_x(x)$ is some unknown distribution, we wish to learn the density $p_x(x)$.

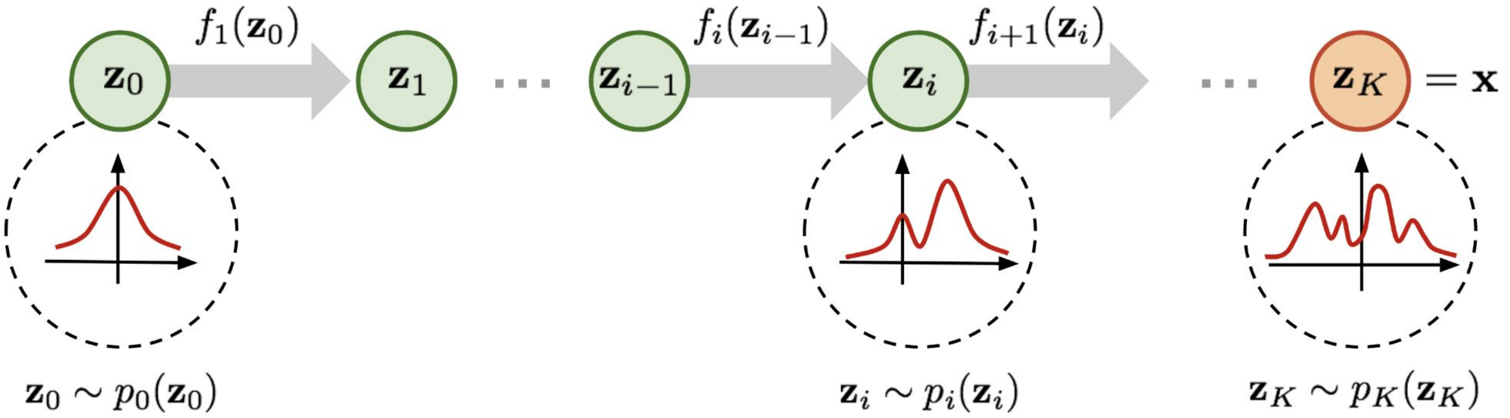
$$\mathcal{L} = \prod_{i=1}^{n} p_x(x_i)$$

$$= \prod_{i=1}^{n} p_u(u_i)\,|\det \mathbf{J}(f)(u_i)|^{-1}$$

$$\implies \hat{f} = \arg\max_f \prod_{i=1}^{n} p_u(u_i)\,|\det \mathbf{J}(f)(u_i)|^{-1}$$

$$\hat{f} = \arg\max_f \sum_{i=1}^{n} \log p_u(u_i) - \log|\det \mathbf{J}(f)(u_i)|$$

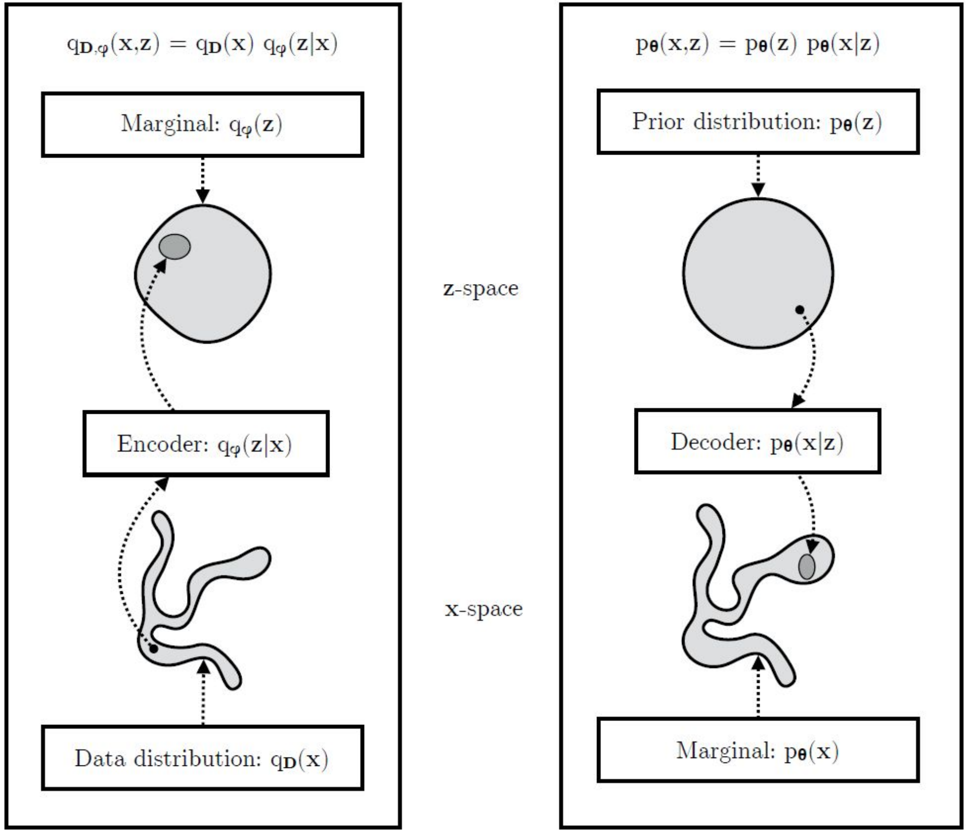
A normalizing flow defines a sequence of *bijectors*



https://lilianweng.github.io/posts/2018-10-13-flow-models/

Now that we're at the end of the lecture, you should be able to...

- ★ Distinguish generative from discriminative models, and recall two models that focus on learning an **explicit representation of the distribution: VAEs and normalizing flows**.
- ★ Differentiate autoencoders and variational autoencoders on the basis of the **stochasticity of their outputs** and the **structure of their latent spaces**, and recommend one or the other for particular use-cases.
- ★ Describe how VAEs generate new samples using the **prior distribution in latent space** and the **decoder.**
- ★ Interpret the **loss function of a VAE** with reference to **KL divergence**, **prior distribution in latent space**, **evidence lower bound**.
- ★ Defend the use of a **Gaussian prior** in VAE and the **reparametrization trick** for enabling **backpropagation through random operations**.
- ★ List limitations of VAEs and recommend approaches to **improve performance**.
- ★ Use the latent space of a VAE to **interpolate between points in data space**, given the parameters of a small model.
- ★ Differentiate VAEs and normalizing flows on the basis of offering **explicit density evaluation.**
- ★ Use **invertible transformations** to **compute probability density** of a **target distribution** from a **source distribution**.



Probabilistic Machine Learning: Advanced Topics Section 21.3