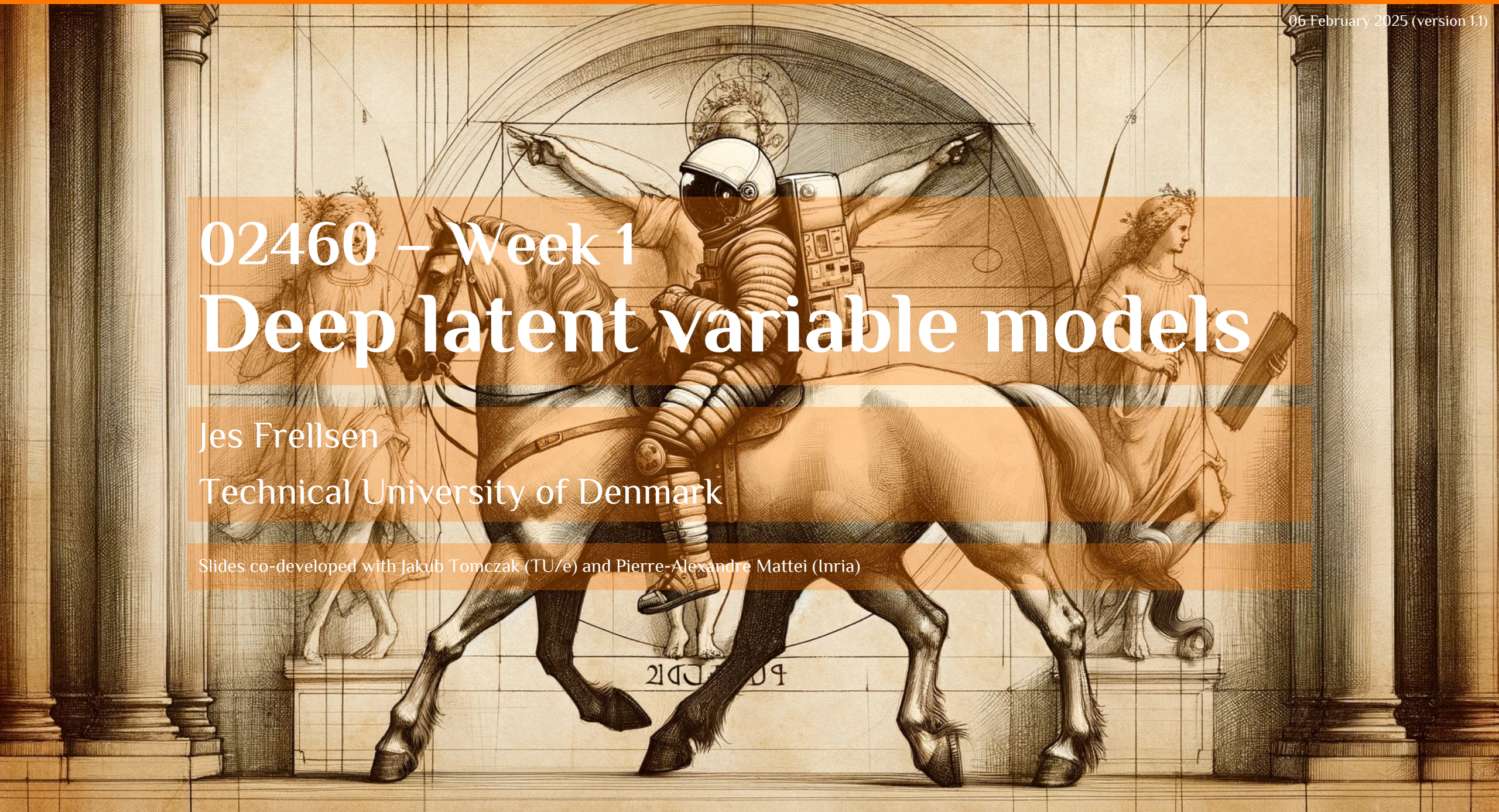
06 February 2025 (version 1.1)



02460 – Week 1
Deep latent variable models

Jes Frellsen

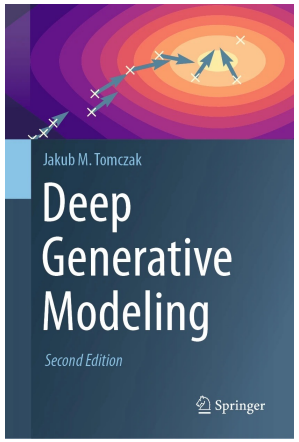
Technical University of Denmark

Slides co-developed with Jakub Tomczak (TU/e) and Pierre-Alexandre Mattei (Inria)

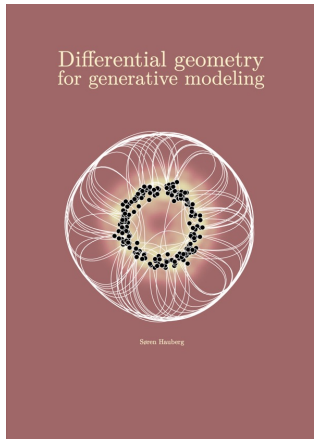
Course overview

Week 13: Q&A

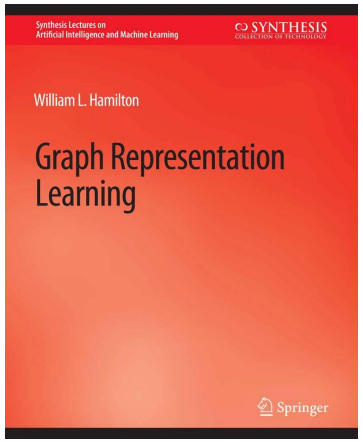
Module 1 **Deep generative models**		Module 2 **Geometric representations**		Module 3 **Graph models**	
Week 1-3: Teaching	**Week 4:** Mini-project 1	**Week 5-7:** Teaching	**Week 8:** Mini-project 2	**Week 9-11:** Teaching	**Week 12:** Mini-project 3



Jes Frellsen



Søren Hauberg



Mikkel N. Schmidt,

Second edition (2024)

Couse activities

- In teaching weeks (Thursdays)
  - Lectures 13:00-15:00 in B306-A033
  - Exercises 15:00-17:00 in B306-A033
- In project weeks
  - Q&A 13:00-17:00 in B306-A033 and B306 group area west
- We have four outstanding TA in the exercise and project Q&As
- Communication:
  DTU Learn and Slack


Alejandro


Marisa


Rasmus

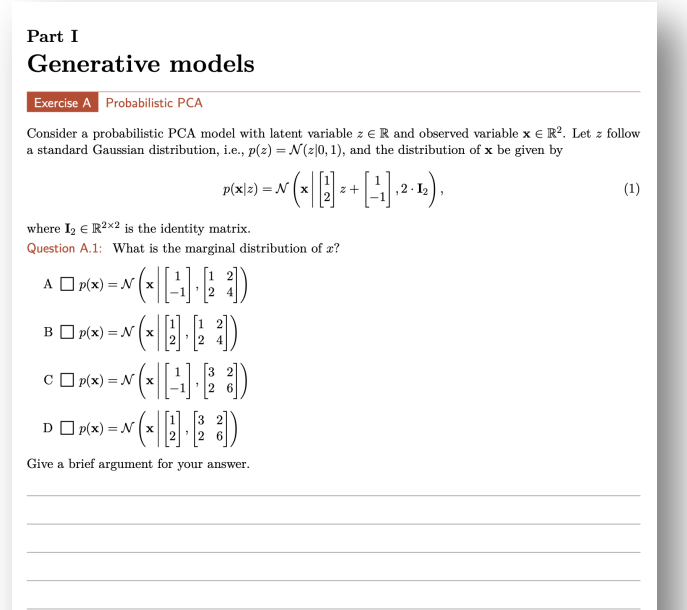

Stas

Course alignment and evaluation

- The weekly exercises consist of
  - **Programming exercises** that will prepare you for the **mini-projects**
  - **Theoretical exercises** that will prepare you for the **written exam**

Mock exam (2024) on Learn

- Your grade is based on the evolution of:
  - The **three mini-projects** (done in groups of 3-4 people)
    - Report: max 1 page + reference and code
  - The **final written exam**
    - 2h on premise
    - No electronic aids
    - Written works of reference are permitted



Part I

Generative models

Exercise A Probabilistic PCA

Consider a probabilistic PCA model with latent variable $z \in \mathbb{R}$ and observed variable $\mathbf{x} \in \mathbb{R}^2$. Let z follow a standard Gaussian distribution, i.e., $p(z) = \mathcal{N}(z|0,1)$, and the distribution of $\mathbf{x}$ be given by

$$p(\mathbf{x}|z) = \mathcal{N}\left(\mathbf{x} \middle| \begin{bmatrix}1\\2\end{bmatrix} z + \begin{bmatrix}1\\-1\end{bmatrix}, 2 \cdot \mathbf{I}_2\right), \quad (1)$$

where $\mathbf{I}_2 \in \mathbb{R}^{2\times 2}$ is the identity matrix.

Question A.1: What is the marginal distribution of x?

A ☐ $p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x} \middle| \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}1 & 2\\2 & 4\end{bmatrix}\right)$

B ☐ $p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x} \middle| \begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}1 & 2\\2 & 4\end{bmatrix}\right)$

C ☐ $p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x} \middle| \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}3 & 2\\2 & 6\end{bmatrix}\right)$

D ☐ $p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x} \middle| \begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}3 & 2\\2 & 6\end{bmatrix}\right)$

Give a brief argument for your answer.

Our expectations to you

- This is DTU's most advanced MSc-level ML course
- We expect that you have the right prerequisites
  - 02450 Introduction to ML and Data Mining
  - 02456 Deep learning

  } Expected

  - 02476 Machine Learning Operations
  - 02405 Probability theory
  - 02477 Bayesian machine learning

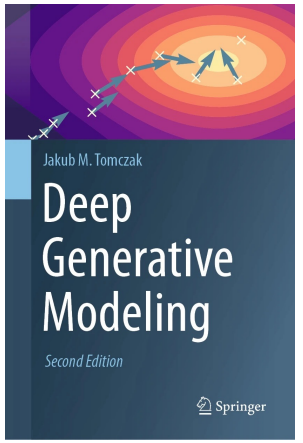
  } Preferably

  } Can be followed in parallel
- We expect that you
  - Read before the lectures
  - Do on the exercises (at least look at them before the sessions)
  - Have installed PyTorch, can use HPC and code using a modern editor

It's the second time we run the course

There may still be some hiccups

Menu of the day



Module 1: Generative models

- Week 1: DLVMs
- Week 2: Flows
- Week 3: DDPMs
- Week 4: Mini-project 1

Lecture

- Intro to DGM and DLVMs
  - Chapter 1
  - Chapter 5-5.4.1.5 and 5.5.1-5.5.2

Exercises

- **Theoretical**: derivations about PPCA, ELBO and VAEs
- **Programming**: Implement a VAE on binary MNIST

Lecture outline

- Why do we want to do generative models?
- KL divergence and maximum likelihood estimation
- Introduction to **deep latent variable models (DLVM)**
- **Inference** in DLVM
- **Monte Carlo gradient estimation**
- How to **improve** VAEs?
  - Improving the prior

Why generative models?

- We **observe** some data $x_1, \ldots, x_N \in \mathcal{X}$, e.g., the CelebA dataset

- **Goal:** train an algorithm that will be able to generate new and plausible images!

Images from: https://www.tensorflow.org/datasets/catalog/celeb_a_hq

Why on earth would we want to generate new and plausible images?

Because it is cool!

Why on earth would we want to generate new and plausible images?

Because it can impact society unexpectedly quickly!

← THE BEST INVENTIONS OF 2022

TIME

Artificial Imagination

OpenAI DALL-E 2

The Economist

Science & technology | Generative AI

Large, creative AI models will transform lives and labour markets

The New York Times

OPINION

GUEST ESSAY

Noam Chomsky: The False Promise of ChatGPT

March 8, 2023

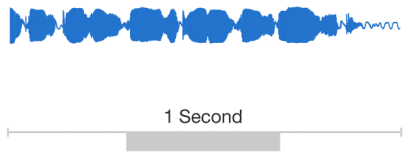
It's not just pretty images and texts!

WaveNet: A Generative Model for Raw Audio

Aäron van den Oord **Sander Dieleman** **Heiga Zen**[†]

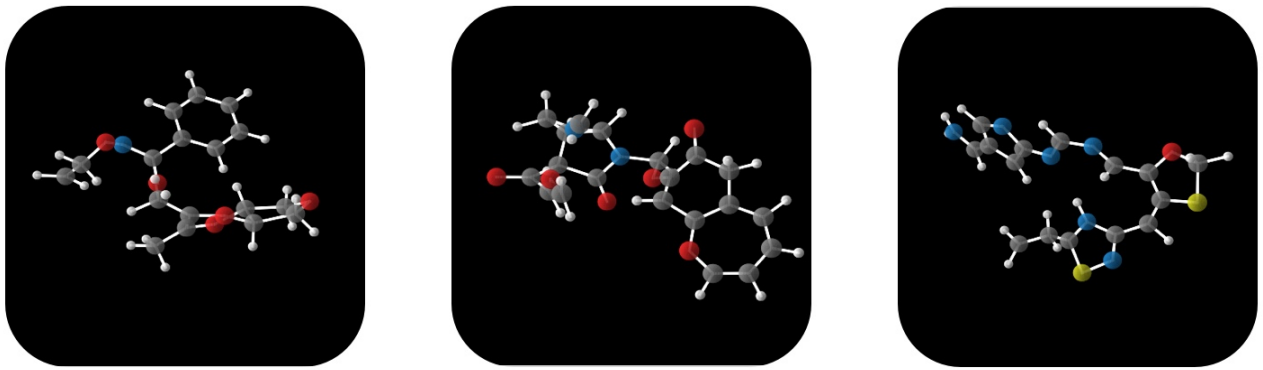
Karen Simonyan **Oriol Vinyals** **Alex Graves**

Nal Kalchbrenner **Andrew Senior** **Koray Kavukcuoglu**



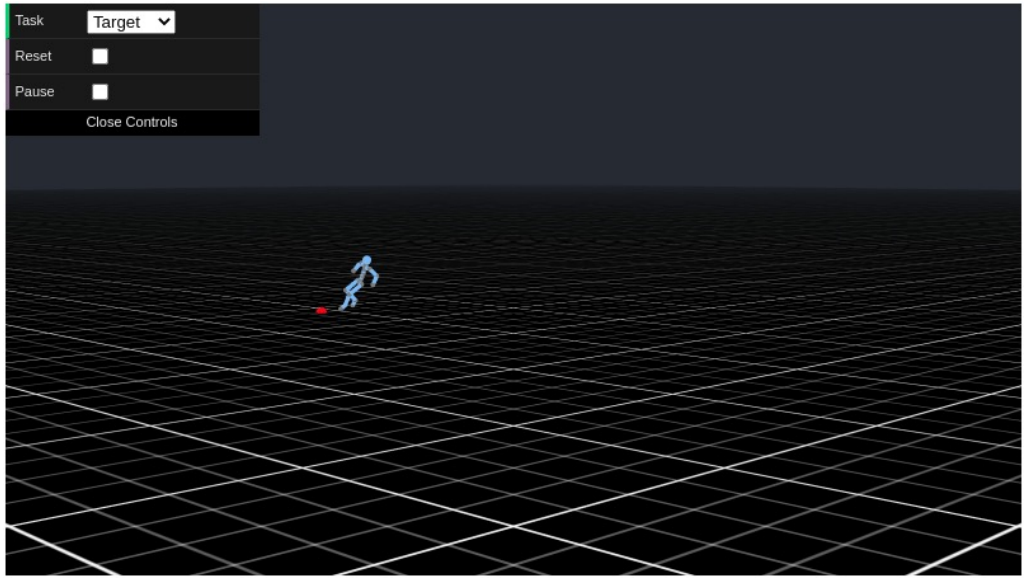
Equivariant Diffusion for Molecule Generation in 3D

Emiel Hoogeboom [*1] **Victor Garcia Satorras** [*1] **Clément Vignac** [*2] **Max Welling** [1]



Character Controllers Using Motion VAEs

HUNG YU LING, University of British Columbia, Canada
FABIO ZINNO, Electronic Arts Vancouver, Canada
GEORGE CHENG, Electronic Arts Vancouver, Canada
MICHIEL VAN DE PANNE, University of British Columbia, Canada

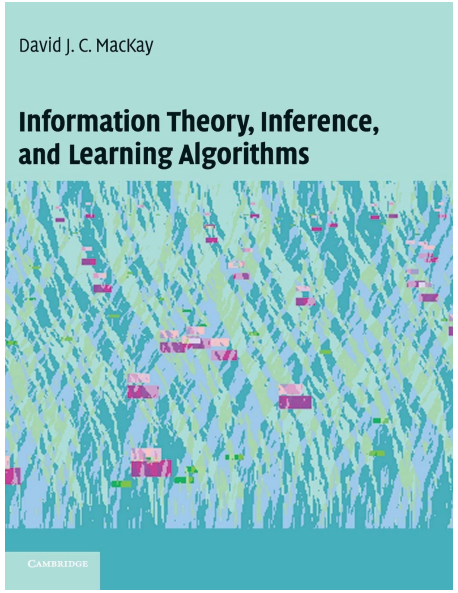


Towards formalising the problem

Our **goal is to generate new data points** that are plausible

- The **mathematical object** that does that is a **probability distribution**
- Our goal will thus be:
  to learn a probability distribution that could have generated the data

> "*a generative model [...] describes a process that is assumed to give rise to some data*" -DJC MacKay

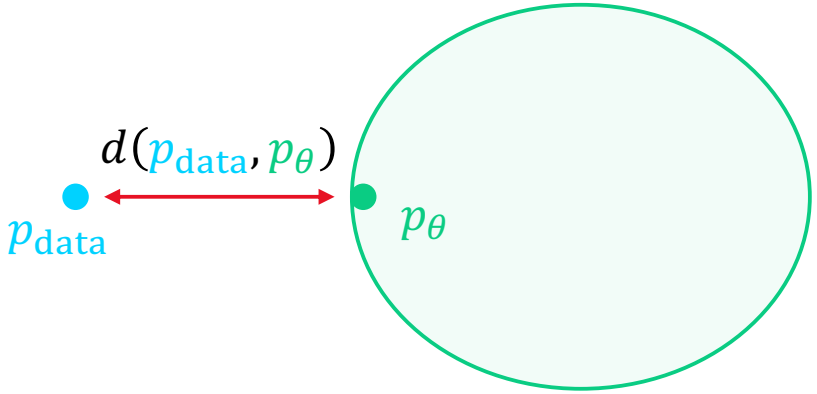


Towards maximum likelihood

- Consider some model family $\{p_\theta\}_{\theta \in \Theta}$
- We want to **find a good** $\hat{\theta} \in \Theta$ such that $p_{\hat{\theta}} \approx p_{\text{data}}$
- Given a distance measure d between distribuions, we want to find

$$\hat{\theta} \in \arg\min_{\theta \in \Theta} d(p_{\text{data}}, p_\theta)$$

$\mathbf{x}_i \sim p_{\text{data}}$



$\theta \in \Theta$
Model family

Illustration based on: Ermon S, Song Y. CS236 , lecture 11.

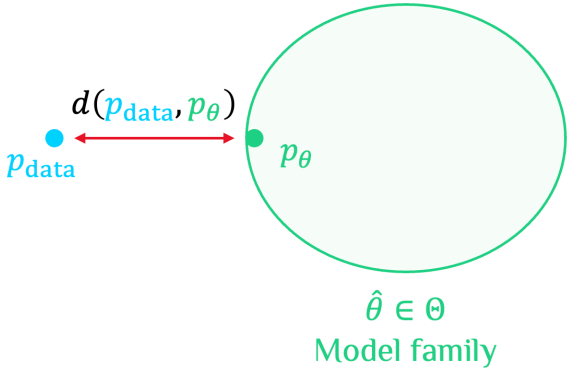
The Kullback-Leibler divergence

- A commonly used measure of how similar two distributions p and q are

$$\mathrm{KL}[p \parallel q] = \int p(x) \log \frac{p(x)}{q(x)} \mathrm{d}x = \mathbb{E}_{x \sim p(x)} \left[\log \frac{p(x)}{q(x)} \right]$$

- Measure the average difference of the log-densities
- **Asymmetrical**, but with some nice "**distance-like**" properties:
  1. $\mathrm{KL}[p \parallel q] \geq 0$
  2. $\mathrm{KL}[p \parallel q] = 0 \Leftrightarrow p = q$

Finding the best model according to KL



We want to **find a good** $\hat{\theta} \in \Theta$ such that $p_{\hat{\theta}} \approx p_{\text{data}}$

$$\hat{\theta} \in \arg\min_{\theta \in \Theta} \mathrm{KL}[p_{\text{data}} \,\|\, p_\theta]$$

where

- $\mathrm{KL}[p_{\text{data}} \,\|\, p_\theta] = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[\log p_{\text{data}}(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[\log p_\theta(\mathbf{x})]$ (Does not depend on θ)
- $\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[\log p_\theta(\mathbf{x})] \approx \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i)$ for samples $\mathbf{x}_1, \ldots, \mathbf{x}_N \sim p_{\text{data}}(\mathbf{x})$

So, we want to find

$$\hat{\theta} \in \arg\max_{\theta \in \Theta} \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i) = \arg\max_{\theta \in \Theta} \ell(\theta)$$

Log-likelihood function

Maximum likelihood estimation

- Given
  - a model family $\{p_\theta\}_{\theta \in \Theta}$
  - observations $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$,
- Our objective for learning θ is **log-likelihood function**

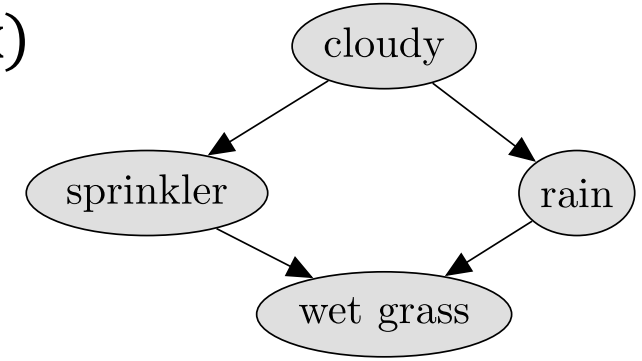
$$\ell(\theta) = \sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i)$$

- We would like to find the **ML estimate**: $\hat{\theta} = \arg\max_{\theta \in \Theta} \ell(\theta)$.
- Today we will consider the **latent variable model** class

What are probabilistic generative models and latent variable models?

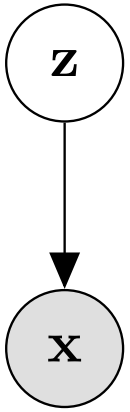
A forward **generative model** "describes the process assumed to rise to data"[1]

- In **unsupervised** learning, we model the (joint) **density** $p(\mathbf{x})$
- The model allows for **generating data** $\mathbf{x} \sim p(\mathbf{x})$
- We assume some **factorisation** of $p(\mathbf{x})$



In a **latent variable** model, we assume an **unobserved random variable** $\mathbf{z}$

- The **joint density** $p(\mathbf{x}, \mathbf{z})$ is modelled
- The model allows for **generating** $\mathbf{x}, \mathbf{z} \sim p(\mathbf{x}, \mathbf{z})$
- We assume some **factorisation** of $p(\mathbf{x}, \mathbf{z})$
- We can think of $\mathbf{z}$ as the **factors** in data or a **code** summarising $\mathbf{x}$

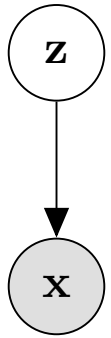


[1] MacKay DJ. Information theory, inference and learning algorithms. Cambridge university press, 2003.

Latent variable models

Generative process

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



Manifold hypothesis: $\mathbf{z} \in \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^D$ where $M < D$

Latent factors, e.g.:

- Colour, angle, background, etc.

We only observe $\mathbf{x}$, and the marginal density of $\mathbf{x}$ is

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

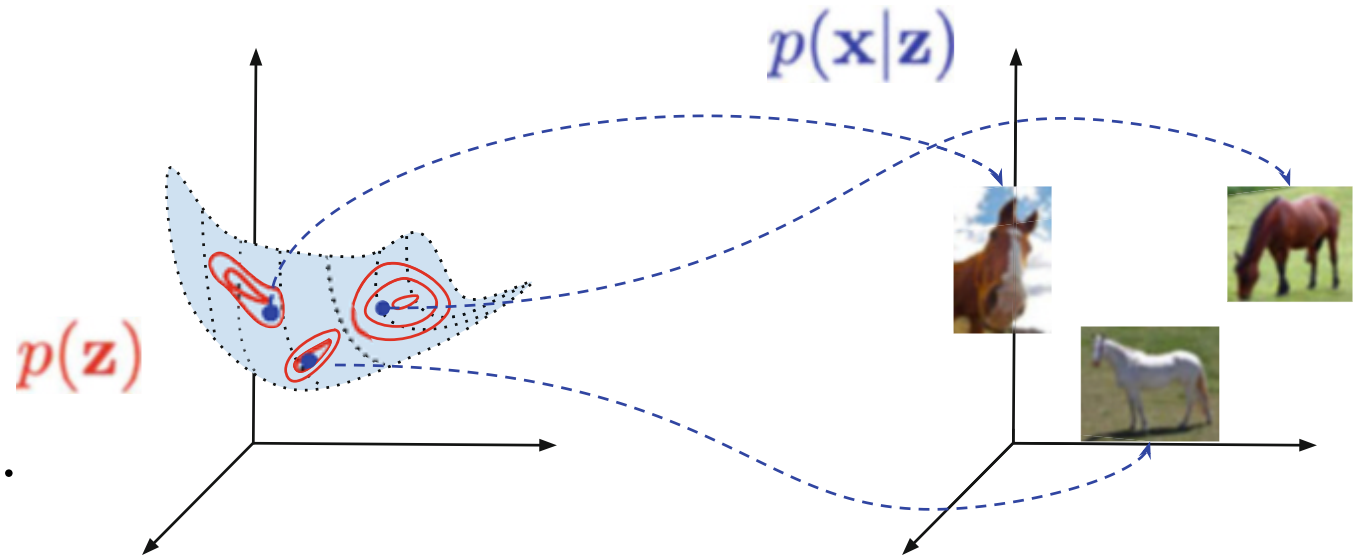


Figure from: Tomczak JM. Deep generative modeling. Springer, 2022.

Deep latent variable models (DLVMs)

- "Old school probabilistic" generative models:
  - + Well-founded framework for model building, inference and prediction
  - - Limited complexity: Mainly conjugate and linear models
  - - Learning is computational expensive
- Deep neural networks:
  - + Rich non-linear models
  - + Scalable learning using stochastic gradient decent
  - - Only give point estimates
  - - Typically, discriminative and non-probabilistic

Deep latent variable models combine the approximation abilities of deep neural networks and the statistical foundations of generative models.

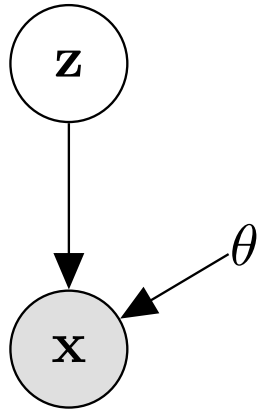
Inspired by: Mohamed, S. and D. Rezende (2017) Tutorial on Deep Generative Models at UAI.

Deep latent variable models (DLVMs)

Assume $\mathbf{x}, \mathbf{z}$ are random variables driven by the model:

$$\mathbf{z} \sim p(\mathbf{z}) \qquad \text{(prior)}$$

$$\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z}) = \Phi(\mathbf{x}|f_\theta(\mathbf{z})) \qquad \text{(observation model)}$$



where

- $\mathbf{z} \in \mathcal{Z}^M$ is the continuous latent variable,
- $\mathbf{x} \in \mathcal{X}$ is the observed variable,
- the function $f_\theta : \mathcal{Z}^M \to H$ (deep) neural network called the decoder,
- $\{\Phi(\cdot\,|\eta)\}_{\eta \in H}$ is a parametric family called the output density.

You can think of a DLVM as non-linear PPCA

You can think of a DLVM as a MM, but with continuous $\mathbf{z}$

The role of the decoder and output density

- Role of **decoder** $f_\theta : \mathcal{Z}^M \to H$ is to parametrize the **output density**
  - I.e. transform $\mathbf{z}$ into **parameters** $\eta = f_\theta(\mathbf{z})$ of the output density $\Phi(\cdot\,|\eta)$
- The **output density** is a model assumption (free to choose)
  - It must make sense for the considered problem and data type, e.g.,

$$p_\theta(\mathbf{x}|\mathbf{z}) = \text{Categorical}(\mathbf{x}|\boldsymbol{\pi} = f_\theta(\mathbf{z})) \qquad \text{for } \mathbf{x} \in \{0, \dots, 255\}^D$$

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = f_\theta(\mathbf{z})) \qquad \text{for } \mathbf{x} \in \mathbb{R}^D$$

  - Or combinations (can be difficult learn)

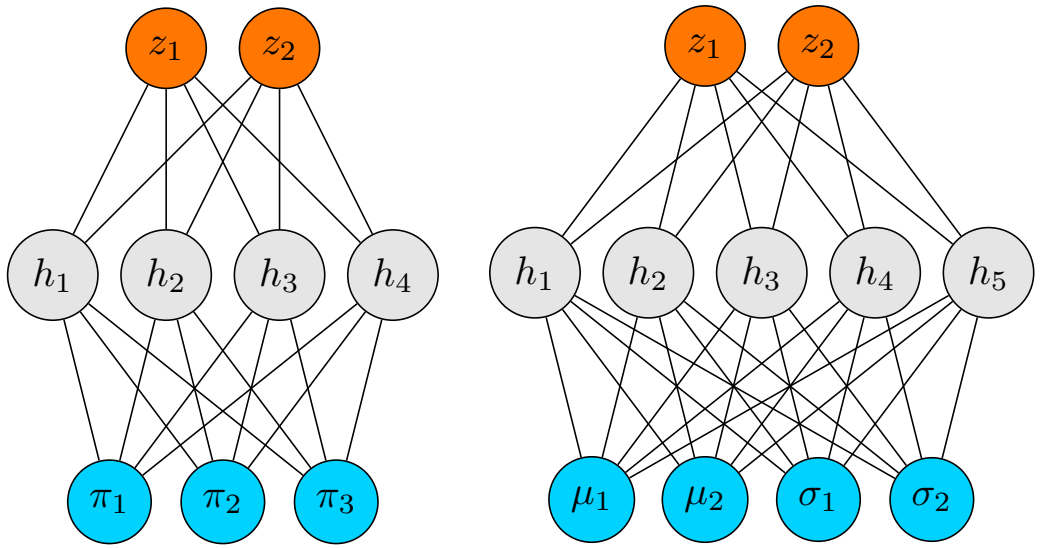
$$p_\theta(\mathbf{x}|\mathbf{z}) = \text{Categorical}(\mathbf{x}_{1:\text{i}}|\boldsymbol{\pi} = f_\theta(\mathbf{z})_{1:\text{i}}) \cdot \mathcal{N}(\mathbf{x}_{i+1:D}|\boldsymbol{\mu}, \boldsymbol{\Sigma} = f_\theta(\mathbf{z})_{i+1:\dots})$$

The role of the decoder and output density

$$p_\theta(\mathbf{x}|\mathbf{z}) = \text{Categorical}(\mathbf{x}|\boldsymbol{\pi} = f_\theta(\mathbf{z})) \qquad \text{for } \mathbf{x} \in \{0, \dots, 255\}^D$$

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = f_\theta(\mathbf{z})) \qquad \text{for } \mathbf{x} \in \mathbb{R}^D$$

- Typically, we just have one **decoder with multiple output heads**.
- The **output of the decoder** must lie in the **parameter space**
  - For the **categorical**, we apply a softmax at the output.
  - For the **Gaussian**, we typically use a **diaoginal covariance** $\boldsymbol{\Sigma} = \text{diag}\left(\sigma_\theta^2(\mathbf{z})\right)$ or $\boldsymbol{\Sigma} = \text{diag}(\exp(f_\theta(\mathbf{z})))$



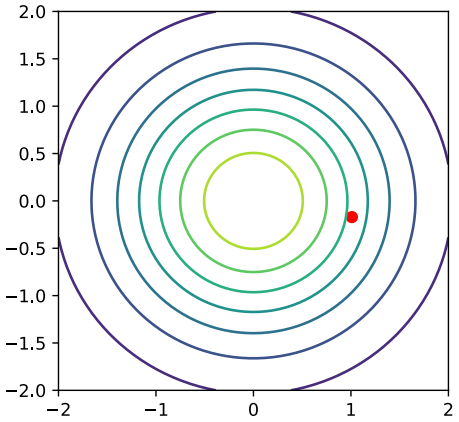
Illustrative example of a DLVM

Training data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ binary MNIST

Generation

$$\mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)$$

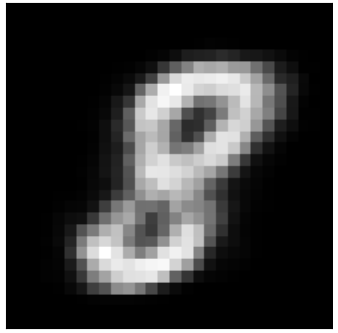
$$\mathbf{z} = (1.0097, -0.1711)$$



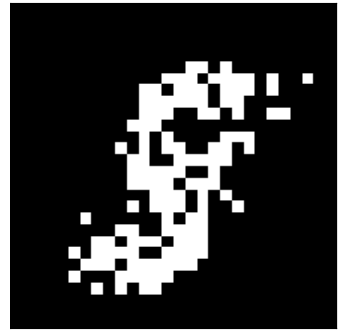
Generative model for $\mathbf{z} \in \mathbb{R}^2$ and $\mathbf{x} \in \{0,1\}^{28\times 28}$

$$\begin{cases} \mathbf{z} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \\ x^{j,k} \sim \text{Bernoulli}(p = f^{j,k}(\mathbf{z})) \end{cases}$$

$f(z)$



$x^{j,k} \sim \text{Bern}(f^{j,k}(z))$



Decoder network

$$f(\mathbf{z}) = \text{Sigmoid}(\mathbf{V}\tanh(\mathbf{W}\mathbf{z} + \mathbf{b}) + \boldsymbol{\beta})$$

"You haven't mentioned an encoder yet?"

"In fact, you haven't even said variational autoencoder"

Spend 3 minutes **discussing with your neighbour**:

- Q: How has the presentation, so far, differed from the "usual" presentation of VAEs?
  - A: I have only focused on the (generative) model description
- Q: Why haven't I mentioned an encoder?
  - A: That is because the encoder is "only" used for learning the model
  - In fact, you can learn in DLVM without an encoder[1,2]
- Q: What insights does it give not mentioning an encoder yet?
  - A: It decouples model and learning (inference) assumptions

[1]Schuster V, Krogh A. The Deep Generative Decode. arXiv:2110.06672, 2021.
[2]Hoffman MD. Learning Deep Latent Gaussian Models with Markov Chain Monte Carlo. ICML, 2018

Learning objective

- We want to **learn the parameters θ of the decoder**
- Given observations $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, a natural objective is **log-likelihood function**

$$\ell(\theta) = \sum_{n=1}^{N} \log p(\mathbf{x}_n) \quad \text{where} \quad p(\mathbf{x}_n) = \int p_\theta(\mathbf{x}_n|\mathbf{z}) p(\mathbf{z}) \, \mathrm{d}\mathbf{z}$$

- We would like to find the **ML estimate**: $\hat{\theta} = \arg\max_\theta \ell(\theta)$.
- However, for non-linear decoders (generally)
  - $p(\mathbf{z}|\mathbf{x}_n)$ is **intractable** rendering **expectation–maximization (EM) intractable**
  - $p(\mathbf{x}_n)$ is **intractable** rendering ***direct*** **MLE intractable**
  - The **MC estimate** $p(\mathbf{x}_n) = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[p_\theta(\mathbf{x}_n|\mathbf{z})] \approx \frac{1}{K}\sum_k p(\mathbf{x}_n|\mathbf{z_k})$ has high variance

Variational inference for DLVMs

- Consider a family distributions $\{q_\phi(\mathbf{z})\}_{\phi \in \Phi}$ which we call the variational family
- Assuming $q_\phi(\mathbf{z}) > 0$ for all $\mathbf{z}$, we can write

$$
\begin{aligned}
\log p(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) \, \mathrm{d}\mathbf{z} \\
&= \log \int \underbrace{\frac{q_\phi(\mathbf{z})}{q_\phi(\mathbf{z})}}_{=1} p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) \, \mathrm{d}\mathbf{z} \\
&= \log \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z})} \left[\frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q_\phi(\mathbf{z})} \right] \\
&\geq \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z})} \left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q_\phi(\mathbf{z})} \right] \\
&= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z})} [\log p_\theta(\mathbf{x}|\mathbf{z})] + \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z})} [\log p(\mathbf{z}) - \log q_\phi(\mathbf{z})] \\
&= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z})} [\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}) \,\|\, p(\mathbf{z})]
\end{aligned}
$$

Jensen's inequality
$g(\mathbb{E}[X]) \geq \mathbb{E}[g(X)]$
for concave g

- This objective is called the **evidence lower bound (ELBO)**, since it a lower bound on $\log p(\mathbf{x})$

Jensen J (1905) Om konvekse Funktioner og Uligheder imellem Middelværdier. Nyt tidsskrift for matematik, 16:49–68.

Amortized variational inference

- Estimating $q_{\phi_n}(\mathbf{z})$ for each datum $\mathbf{x}_n$ is expensive for large datasets
  - Even if q_{ϕ_n} are diagonal Gaussians, we have $N \times 2M$ parameters
- We consider an **amortized variational posterior** (i.e., conditional)

$$q_\phi(\mathbf{z}|\mathbf{x}) = \Psi\big(\mathbf{z}\big|g_\phi(\mathbf{x})\big)$$

  where
  - $\{\Psi(\cdot\,|\kappa)\}_{\kappa \in K}$ is the **variational family** over $\mathcal{Z}^M$ (for $\mathbb{R}^M$ usually Gaussians)
  - $g_\phi : \mathcal{X} \to K$ is a neural net called the **encoder** or **inference network**
  - g_ϕ transforms a datapoint $\mathbf{x}$ into parameters of the variational distribution
- We can consider the variational distribution to be a **tractable approximation** of the posterior $q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z}|\mathbf{x})$

Why is the variational distribution a tractable approximation of the posterior?

- Let's rewrite the ELBO again

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log \left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right]$$

maximise

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

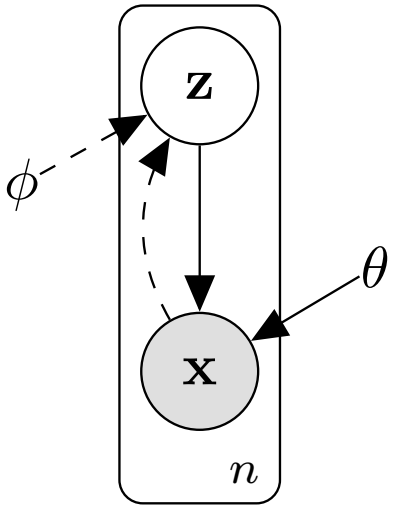
$$= \log p_\theta(\mathbf{x}) + \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log \left[\frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} \right]$$

$$= \log p_\theta(\mathbf{x}) - \mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p_\theta(\mathbf{z}|\mathbf{x}) \right)$$

maximise minimize

The variational autoencoder (VAEs)[1,2]

- A DLVM combined with AVI is called a **variational autoencoder**
  - The **prior** or **marginal distribution** $p(\mathbf{z})$
  - The **likelihood, observation model**, or **stochastic decoder** $p_\theta(\mathbf{x}|\mathbf{z})$
  - The **decoder** $f_\theta: \mathcal{Z}^M \to H$
  - The **variational distribution/posterior** or **stochastic encoder** $q_\phi(\mathbf{z}|\mathbf{x})$
  - The **encoder** $g_\phi: \mathcal{X} \to K$
  - The **ELBO** objective
$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}\big[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\big]$$
  - Optimisation, $\max_{\theta,\phi} \mathcal{L}(\theta, \phi)$, using stochastic gradient descent, where gradient are found using the **reparameterization trick**



[1]Kingma DP, Welling M. Auto-encoding variational Bayes. ICLR, 2014.
[2]Rezende DJ, Mohamed S, Wierstra D. Stochastic backpropagation and approximate inference in deep generative models. ICML 2014.

Monte Carlo gradient estimation

- The ELBO still contains an integral: how to calculate gradients?

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

- For Gaussians, the KL has closed form (exercise today)
- The naïve Monte Carlo (MC) estimator can be written as

$$\nabla_{\theta,\phi}\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[f_{\theta,\phi}(\mathbf{z})] \approx \nabla_{\theta,\phi}\mathcal{F}_L(\theta,\phi) \quad \text{where} \quad \mathcal{F}_L(\theta,\phi) = \frac{1}{L}\sum_{l=1}^{L} f_{\theta,\phi}(\mathbf{z}_l) \quad \text{and} \quad \mathbf{z}_l \sim q_\phi(\mathbf{z}|\mathbf{x})$$

- The gradient wrt. θ are easy, but what about ϕ?

$$\nabla_\theta \mathcal{F}_L(\theta,\phi) = \frac{1}{L}\sum_{l=1}^{L}\nabla_\theta f_{\phi,\psi}(\mathbf{z}_l)$$

$$\nabla_\phi \mathcal{F}_L(\theta,\phi) = ???$$

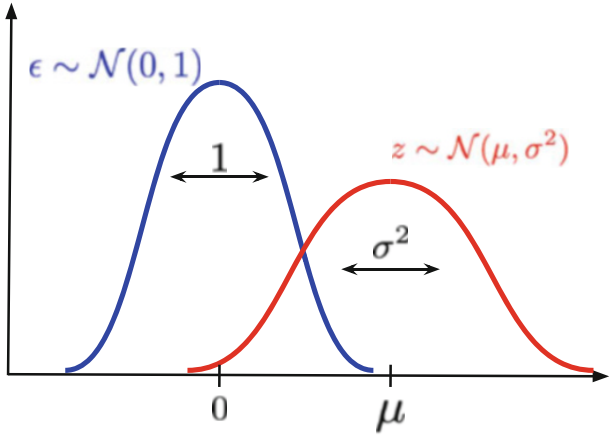
$\mathbf{z}_l$ is a random variable that depends on ϕ, so we would need to backprop sampling process

Reparameterization trick

- We can "**reparameterize**" the random variable $z \sim \mathcal{N}(z|\mu, \sigma^2)$ using

$$z = \mu + \sigma\epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(\epsilon|0,1)$$

  - $g(\epsilon, \mu, \sigma) = \mu + \sigma\epsilon$ is differentiable wrt. (μ, σ)
  - ϵ does not depend on (μ, σ)



- Using this trick, for a Gaussian $q_\phi(\mathbf{z}|\mathbf{x})$, we can write the ∇ELBO as

$$\begin{aligned}
\nabla_{\theta,\phi}\mathcal{L}(\theta,\phi) &= \nabla_{\theta,\phi}\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \nabla_{\theta,\phi}\mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})\right] \\
&= \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0},\mathbf{I}_M)}\left[\nabla_{\theta,\phi}\log p_\theta\left(\mathbf{x}|\boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x}) \odot \boldsymbol{\epsilon}\right)\right] - \nabla_{\theta,\phi}\mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})\right] \\
&\approx \sum_{l=1}^{L} \nabla_{\theta,\phi}\log p_\theta\left(\mathbf{x}|\boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x}) \odot \boldsymbol{\epsilon}_l\right) - \nabla_{\theta,\phi}\mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})\right]
\end{aligned}$$

  where $\boldsymbol{\epsilon}_l \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I}_M)$ and $\left(\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}_\phi(\mathbf{x})\right)$ is the output of the encoder

- This is a low **variance gradient estimator**

Figure from: Tomczak JM. Deep generative modeling. Springer, 2022.

Reparameterization trick

- Also known as the **pathwise gradient estimator**, split into
  1. **Stochastic** parameter free part ($\epsilon \sim \mathcal{N}(\epsilon|0,1)$)
  2. A **deterministic** parametric transformation ($z = \mu + \sigma\epsilon$)
- **Many distributions** can be reparameterized
  - Including: Gaussian, Beta, Gamma, StudentT, and Dirichlet
  - Implemented as `rsample()` in `torch.distributions`
  - **Does not work for discrete variable** (only permutations possible)
    - Solved by relaxed discrete distribution, e.g., the Gumbel-softmax trick[1,2]
- Other **Monte Carlo gradient estimation** exist
  - Including score function estimators (**REINFORCE**) and measure-valued gradient estimators, see Mohamed et al. (2020)[3] for more details

[1]Jang E, Gu S, Poole B. Categorical Reparameterization with Gumbel-Softmax. ICLR 2017.
[2]Maddison C, Mnih A, Teh Y. The concrete distribution: A continuous relaxation of discrete random variables. ICLR, 2017.
[3]Mohamed S, Rosca M, Figurnov M, Mnih A. Monte Carlo Gradient Estimation in Machine Learning. JMLR, 2020.

Why is it called a VAE?

$$-\frac{1}{2}\left\|\mathrm{x}-f_\theta\left(g_\phi(\mathbf{x})+\boldsymbol{\epsilon}\right)\right\|^2-\log Z$$

- Let's revisit the ELBO for Gaussian output

$$\begin{aligned}\mathcal{L}(\theta,\phi) &= \overbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}^{\text{reconstruction term}} - \overbrace{\mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\right]}^{\text{regularisation term}} \\ &= \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0},\mathbf{I}_M)}\left[\log \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}=f_\theta\left(g_\phi(\mathbf{x})+\boldsymbol{\epsilon}\right), I_D\right)\right] - \mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel \mathcal{N}(\mathbf{z})\right]\end{aligned}$$

  - Here, we only encode the mean of a Gaussian variational distribution
- And the loss of an autoencoder (AE), i.e.,

$$L(\theta,\phi) = \left\|\mathbf{x}-\mathrm{Dec}_\theta\left(\mathrm{Enc}_\phi(\mathbf{x})\right)\right\|^2$$

- The VAE can be viewed as an AE with
  - a noisy encoder
  - a Gaussian regularisation of the latent space

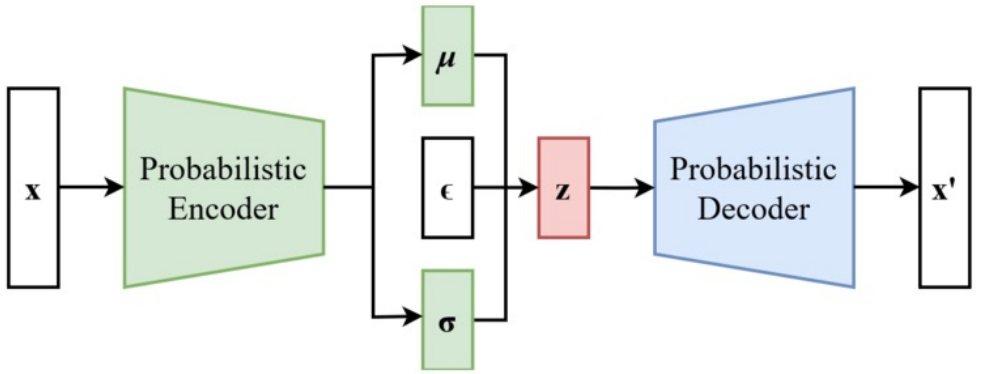


Figure from: https://commons.wikimedia.org/wiki/File:Reparameterized_Variational_Autoencoder.png

Latent representation

$p(\mathbf{z}|\mathbf{x}) \approx q(\mathbf{z}|\mathbf{x})$ gives a **stochastic latent representation** of data, e.g., for MNIST:

Aggregated posterior

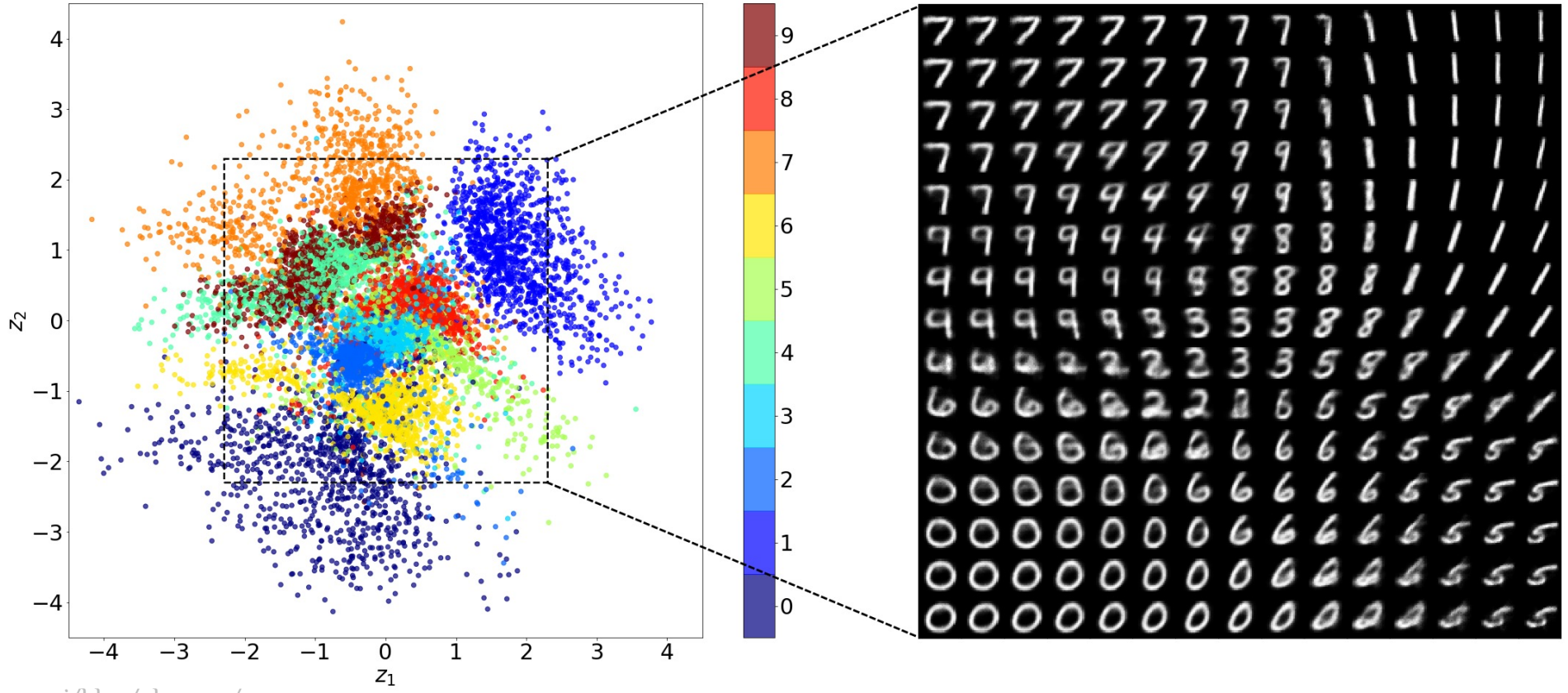
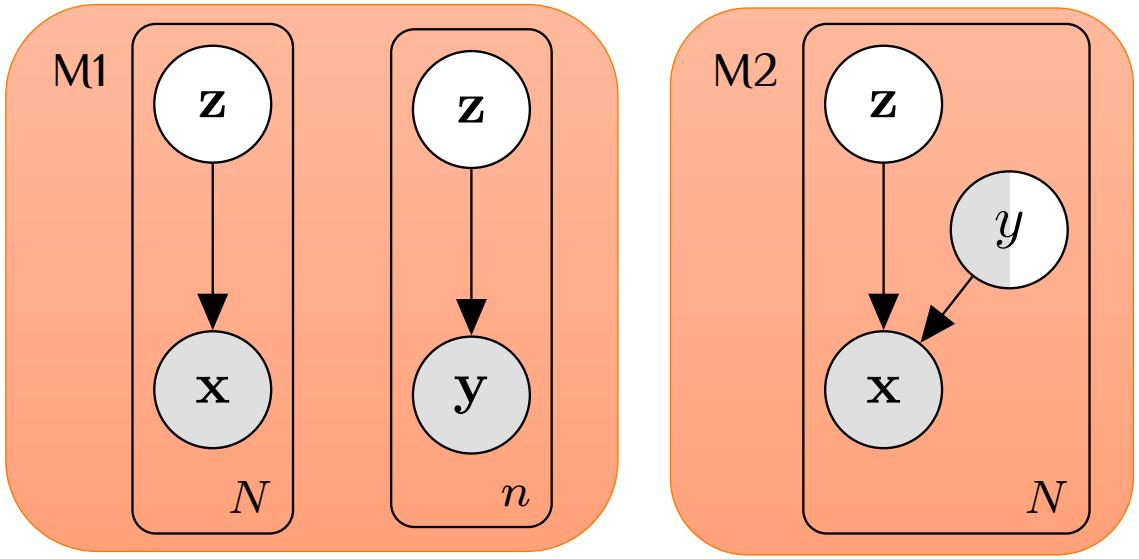


Figure from: https://argmax.ai/blog/vhp-vae/

What can we use VAEs for?



- As data **generators**, e.g., for
  - Generating pretty images 😀
  - Data augmentation
- As a **density model**, e.g., for
  - Missing data imputation[1,2,3]
  - Information acquisition /active sequential variable selection[3]
- For dimensionality reduction or **representation learning**, e.g., for
  - Data visualisation
  - Semi-supervised learning[4]

[1]Nazabal A, Olmos PM, Ghahramani Z, Valera I. Handling incomplete heterogeneous data using VAEs. arXiv:1807.03653, 2018.
[2]Mattei P-A, Frellsen J. MIWAE: Deep Generative Modelling and Imputation of Incomplete Data Sets. ICML, 2019.
[3]Ma C, Tschiatschek S, Palla K, Hernandez Lobato JM, Nowozin S, Zhang C. EDDI: Efficient Dynamic Discovery of High-Value Information with Partial VAE. ICLR, 2019.
[4]Kingma DP, Mohamed S, Rezende DJ, Welling M. Semi-supervised Learning with Deep Generative Models. NeurIPS, 2014.

Are VAEs Bayesian?

- NO! They just borrow a lot of **thermology** from Bayesian inference
- **We learn the model by approximate MLE**
- The prior $p(\mathbf{z})$ is not a Bayesian prior: we don't update it
  - It does not change after we have seen data
  - We may learn it by MLE (as we will see soon)
- The **posterior $p(\mathbf{z}|\mathbf{x}) \approx q(\mathbf{z}|\mathbf{x})$** corresponds to the **component responsibility** in a mixture model
- **You can make Bayesian VAEs** by putting Bayesian priors on, e.g., decoder parameters θ[1]

[1]Daxberger E, Hernández-Lobato JM. Bayesian Variational Autoencoders for Unsupervised Out-of-Distribution Detection. arXiv:1912.05651, 2019.

Issues with VAEs (1/3)

- Posterior collapse[1] (e.g., 1)
  - For a non-trainable prior, e.g., standard Gaussian, the **regularisation term** will be minimized if $\forall \mathbf{x} : q_\phi(\mathbf{z}|\mathbf{x}) = p(\mathbf{z})$
  - A powerful decoder can learn to **ignore** $\mathbf{z}$ (e.g. ARM like a LSTM)
  - Can partially be addressed by warm-up of the KL-term[2,3] (e.g., 2,3)

$$\mathcal{L}(\theta, \phi) = \overbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}^{\text{reconstruction term}} - \overbrace{\mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\right]}^{\text{regularisation term}}$$

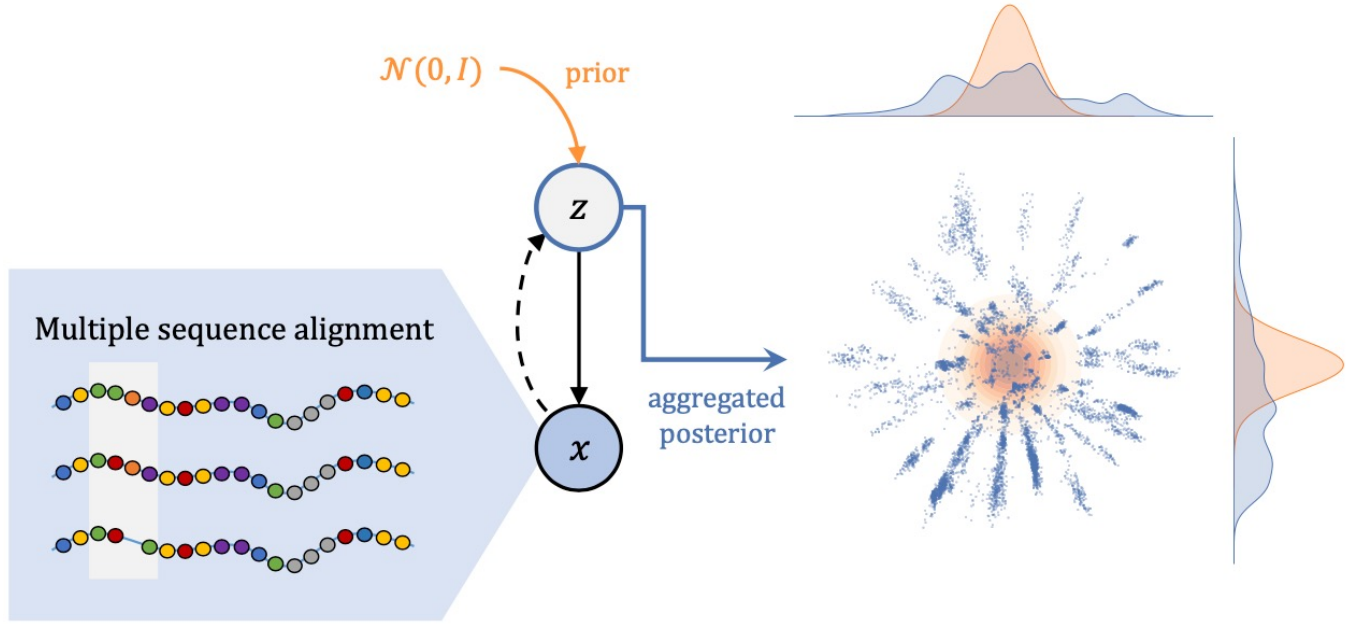
$$\mathcal{L}(\theta, \phi) = \overbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}^{\text{reconstruction term}} - \beta \cdot \overbrace{\mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\right]}^{\text{regularisation term}}$$

$\beta \in [0; 1]$

[1] Lucas J, Tucker G, Grosse R, Norouzi M. Understanding Posterior Collapse in Generative Latent Variable Models. 2019.
[2] Higgins I, Matthey L, Pal A, Burgess C, Glorot X, Botvinick M, Mohamed S, Lerchner A. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. ICLR, 2017.
[3] Sønderby CK, Raiko T, Maaløe L, Sønderby SK, Winther O. Ladder Variational Autoencoders. NeurIPS, 2016.

Issues with VAEs (2/3)

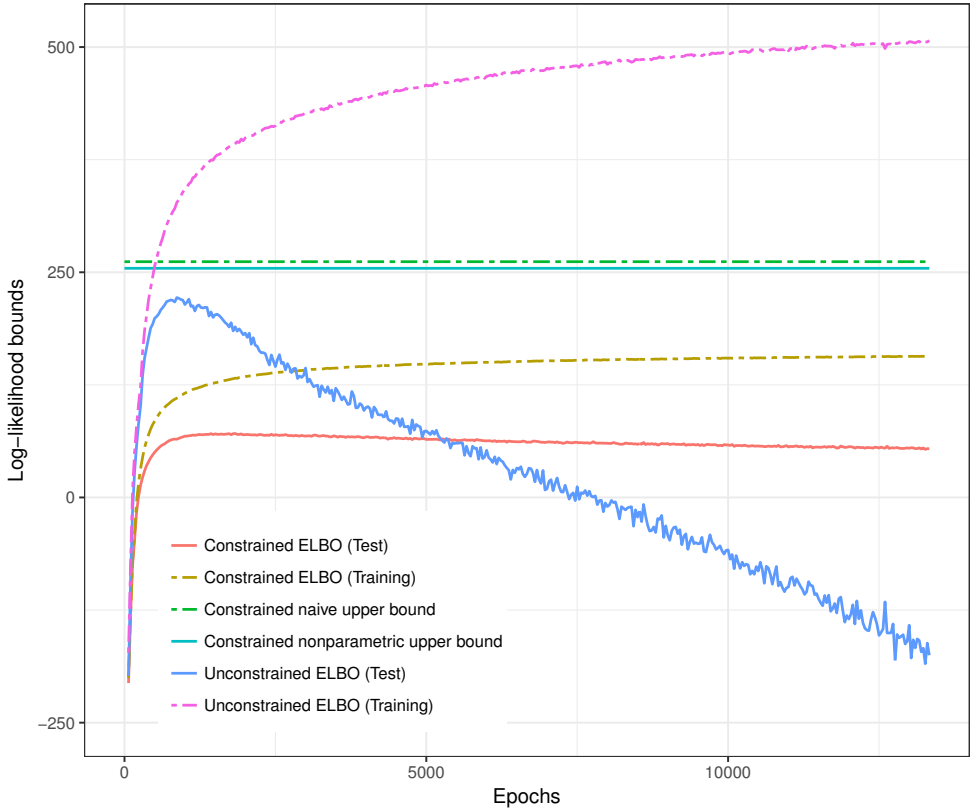
- The **hole problem**[1]
  - Mismatched between the prior $p(\mathbf{z})$ and the aggregated posterior
    $$q_\phi(\mathbf{z}) = \frac{1}{N}\sum_n q_\phi(z|x_n)$$
  - Sampling from these holes gives unrealistic $\mathbf{z}$'s ⇒ $p(\mathbf{x}|\mathbf{z})$ is low quality



[1]Rezende DJ, Viola F. Taming VAEs. arXiv:1810.00597, 2018.
Figure from: Arts ME, Bergamin F, Boomsma W, Frellsen J. Sampling quality in deep generative models of protein sequences. Unpublised, 2023.

Issues with VAEs (3/3)

- **Unbounded likelihood**[1]
  - As for GMMs, **MLE is ill-posed** for VAEs with Gaussian output density
  - ***The likelihood is unbounded***
  - ***Detrimental to generalisation***
  - **Solution**: regularise, clamp or Bayesian about co-variance



[1]Mattei PA, Frellsen J. Leveraging the Exact Likelihood of Deep Latent Variable Models. NeurIPS, 2018.

Improving VAEs

- We can improve all the components of a VAE
- Here, we focus on
  - the **prior**

... but there is much more, we can also improve

  - the **inference method**
  - the **variational posterior** (e.g. with flows)

Improving the prior

$$\mathcal{L}(\theta,\phi) = \overbrace{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}^{\text{reconstruction term}} - \overbrace{\mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\lambda(\mathbf{z})\right]}^{\text{regularisation term}}$$

$$\mathbb{CE}\left[q_\phi(\mathbf{z}) \parallel p_\lambda(\mathbf{z})\right] = -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z})}[\log p_\lambda(\mathbf{z})]$$

Maximize

Minimize

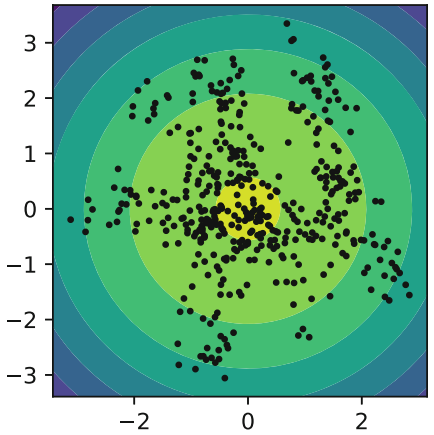
- We can rewrite the **regularization** term as

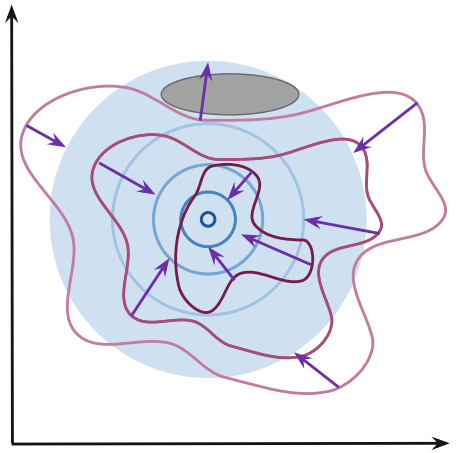
$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[-\mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\lambda(\mathbf{z})\right]\right] = -\mathbb{CE}\left[q_\phi(\mathbf{z}) \parallel p_\lambda(\mathbf{z})\right] + \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\mathbb{H}\left[q_\phi(\mathbf{z}|\mathbf{x})\right]\right]$$

  where $q_\phi(\mathbf{z}) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[q_\phi(\mathbf{z}|\mathbf{x})] \approx \frac{1}{N}\sum_n q_\phi(\mathbf{z}|\mathbf{x}_n)$ is the **aggregated posterior**

- **Difficulties in optimising the CE** wrt. q_ϕ for a fixed prior gives rise **holes**
- The cross entropy, is minimised wrt. $p_\lambda(\mathbf{z})$ when $p_\lambda(\mathbf{z}) = q_\phi(\mathbf{z})$
- This can be addressed by learning the prior $p_\lambda(\mathbf{z})$

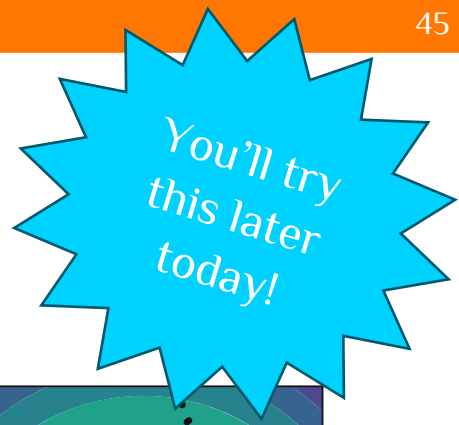
Maximize, i.e. towards high variance





Figures from: Tomczak JM. Deep generative modeling. Springer, 2022.

Mixture of Gaussians (MoG) prior



- What is then the best prior? The **aggregated posterior**:

$$p_\lambda(\mathbf{z}) = \frac{1}{N}\sum_{n=1}^{N} q_\phi(\mathbf{z}|\mathbf{x}_n)$$

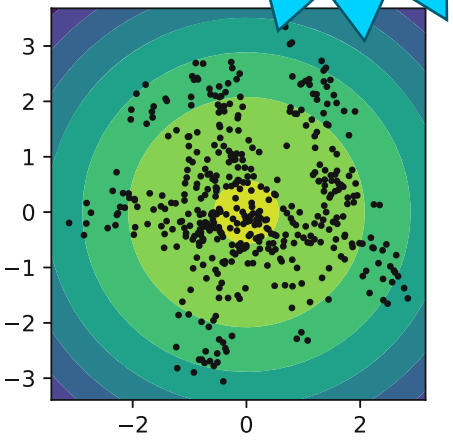
  It minimeres CE, but is not feasible for large N

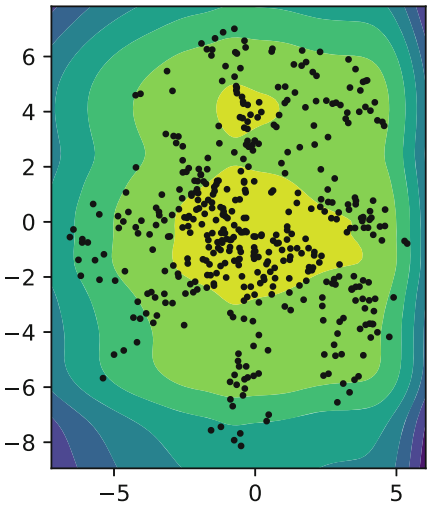
- Notice that this is a mixture distribution!
- So, we could use a **MoG prior** with $K < N$ components

$$p_\lambda(\mathbf{z}) = \sum_{k=1}^{K} w_k \mathcal{N}\left(\mathbf{z}|\boldsymbol{\mu}_k, \mathrm{diag}(\boldsymbol{\sigma}_k^2)\right)$$

  were $\lambda = \left\{\{w_k\}, \{\boldsymbol{\mu}_k\}, \{\boldsymbol{\sigma}_k^2\}\right\}$ are learnable parameters

- ***Doesn't this introduce a new latent variable?***
- **This also allows for end-to-end clustering with VAEs**





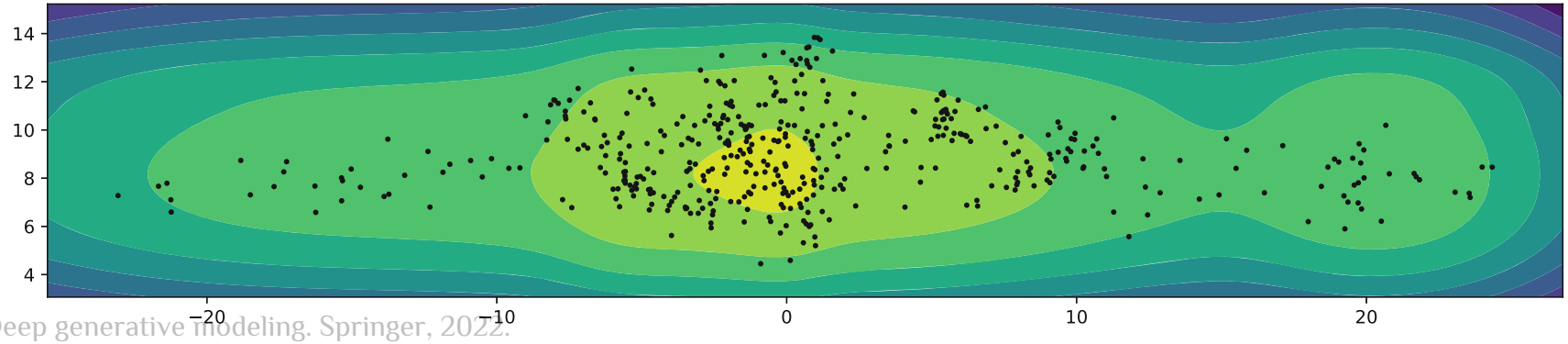
Figures from: Tomczak JM. Deep generative modeling. Springer, 2022.

Variational mixture of posterior prior (VampPrior)[1]

- We can improve on the MoG prior by using a **mixture of the approximate posterior** with $K < N$ pseudo-inputs

$$p_\lambda(\mathbf{z}) = \frac{1}{K}\sum_{k=1}^{K} q_\phi(\mathbf{z}|\mathbf{u}_k)$$

  where $\lambda = \{\{\mathbf{u}_k\}, \phi\}$ are learnable parameters
- Improves the quality of the model (**fewer holes**)
- Challenging initializing pseudo-inputs



Figures from: Tomczak JM. Deep generative modeling. Springer, 2022.

[1]Tomczak J, Welling M. VAE with a VampPrior. AISTATS, 2018.

Hierarchical latent variable models[1,2]

- Can we build a **flexible** prior using an **inductive bias**?
  - Common (ML) **hypothesis**: our world can be organized **hierarchically**
- We can make a VAE, with multiple hierarchical latent variable, e.g.

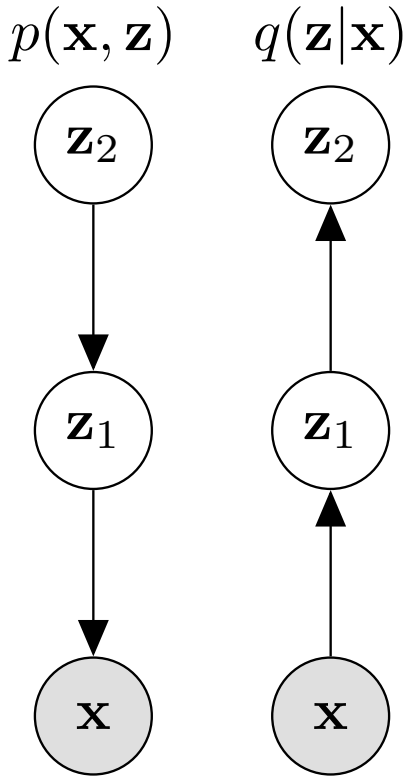
$$p(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2) = p(\mathbf{x}|\mathbf{z}_1)p(\mathbf{z}_1|\mathbf{z}_2)p(\mathbf{z}_2)$$

- A natural variational posterior is **bottom-up**

$$q(\mathbf{z}_1, \mathbf{z}_2|\mathbf{x}) = q(\mathbf{z}_2|\mathbf{z}_1)q(\mathbf{z}_1|\mathbf{x})$$

- However, we easily get posterior collapse for $\mathbf{z}_1$

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{z}_1,\mathbf{z}_2|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z}_1) + \log \frac{p(\mathbf{z}_1|\mathbf{z}_2)}{q(\mathbf{z}_1|\mathbf{x})} - \mathrm{KL}[q(\mathbf{z}_2|\mathbf{z}_1)||p(\mathbf{z}_2)]\right]$$



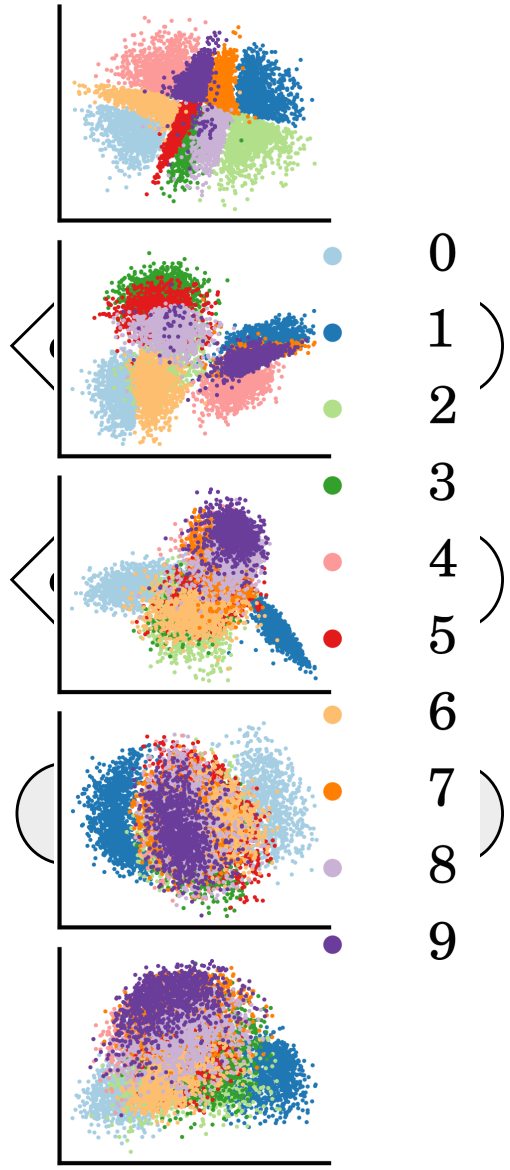
[1]Rezende DJ, Mohamed S, Wierstra D. Stochastic backpropagation and approximate inference in deep generative models. ICML 2014.
[2]Sønderby CK, Raiko T, Maaløe L, Sønderby SK, Winther O. Ladder Variational Autoencoders. NeurIPS, 2016.

Hierarchical latent variable models

- Instead, we can write the variational posteriors **top-down**

$$q(\mathbf{z}_1, \mathbf{z}_2|\mathbf{x}) = q(\mathbf{z}_1|\mathbf{z}_2, \mathbf{x})q(\mathbf{z}_2|\mathbf{x})$$

- The decoder is **shared** between the variational posterior and the prior
- This helps **preventing posterior collapse**
- Warm-up (gradually turning on the KL-term) is often still required



Figures from: Sønderby CK, Raiko T, Maaløe L, Sønderby SK, Winther O. Ladder Variational Autoencoders. NeurIPS, 2016.

Very deep VAEs (48 layers) on FFHQ-256[1]

[1]Child R. Very deep VAEs generalize autoregressive models and can outperform them on images. arXiv:2011.10650, 2020.

Take-away!

- DLVM: a power framework for building generative model
- VAE: a DLVM learned with amortized variational inference
- Allows for **representation learning** and **semi-supervised** learning
- Key components we can **improve**
  - Prior
  - Inference
  - Variational distribution

Thank you!