An introduction to Variational Autoencoders

Giovanni Catalani, Joseph Morlier
2024/2025 ISAE-Supaero

Latent Variable Model

A latent variable model defines a joint distribution over observed variables x and unobserved (latent) variables z

Posterior Distribution

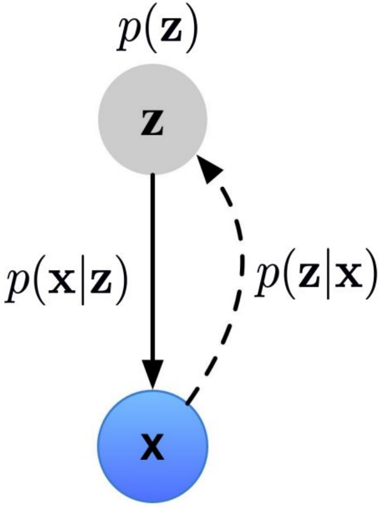
- $p(x,z) = p(x|z)p(z)$
- Marginal Likelihood $p(x) = \int p(x|z)p(z)dz$ is the probability of observing the data that we want to maximize.
- For a dataset of N observations $\log p(X) = \sum_{i=1}^{N} \log p(x^{(i)}) = \sum_{i=1}^{N} \log \int p(x^{(i)}|z)p(z)dz$

Typically the observations comes from a high dimensional distribution (ex. images, text,...) and the latent variables are assumed to be lower dimensional and from a tractable distribution for applications like:

- Dimensionality Reduction.
- Generative modeling (by sampling from the latent distribution).

Challenge:

- Maximizing directly the marginal likelihood is hard because the integral is intractable in general.



https://storage.googleapis.com/deepmind-media/UCLxDeepMind_2020/L11%20-%20UCLxDeepMind%20DL2020.pdf

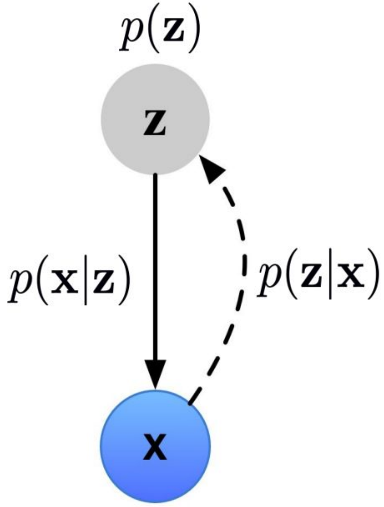
Latent Variable Model

Challenge:

- Maximizing directly the marginal likelihood is hard because the integral is intractable in general.

$$\begin{aligned}
\nabla_\theta \log p_\theta(\mathbf{x}) &= \frac{\nabla_\theta p_\theta(\mathbf{x})}{p_\theta(\mathbf{x})} = \frac{\int \nabla_\theta p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}}{p_\theta(\mathbf{x})} \\
&= \frac{\int p_\theta(\mathbf{x}, \mathbf{z}) \nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}}{p_\theta(\mathbf{x})} \\
&= \int p_\theta(\mathbf{z}|\mathbf{x}) \nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}
\end{aligned}$$

Using the identity $\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}) = \frac{\nabla_\theta p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{x}, \mathbf{z})}$

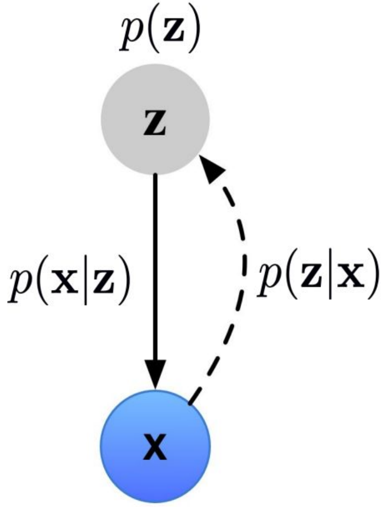


https://storage.googleapis.com/deepmind-media/UCLxDeepMind_2020/L11%20-%20UCLxDeepMind%20DL2020.pdf

Variational Inference

Variational inference (VI) turns the task of finding the posterior distribution into an optimization problem.

- The idea is to approximate the exact posterior with an approximate posterior $q_\phi(z|x) \approx p_\theta(z|x)$
- It should be easy to sample from the approx. posterior and to optimize wrt the parameters ϕ
- What is the **optimization objective**? It should be related to the original marginal likelihood



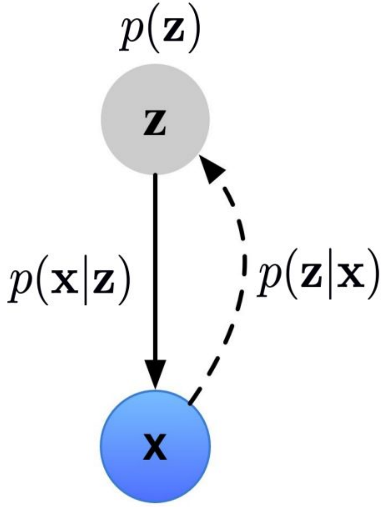
https://storage.googleapis.com/deepmind-media/UCLxDeepMind_2020/L11%20-%20UCLxDeepMind%20DL2020.pdf

Variational Inference: the Evidence Lower Bound

For **any** $q_\phi(z)$ it holds the following inequality :

$$
\begin{aligned}
\log p_\theta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\
&= \log \int q_\phi(\mathbf{z}) \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z})} d\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z})} d\mathbf{z} \\
&= \mathbb{E}_{q_\phi(\mathbf{z})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z})} \right]
\end{aligned}
$$

Using the Jensen's inequality:
$\log \mathbb{E}_{q_\phi(\mathbf{z})} [f(\mathbf{z})] \geq \mathbb{E}_{q_\phi(\mathbf{z})} [\log f(\mathbf{z})]$



Using the variational posterior as $q_\phi(z)$, we obtain the **ELBO:**

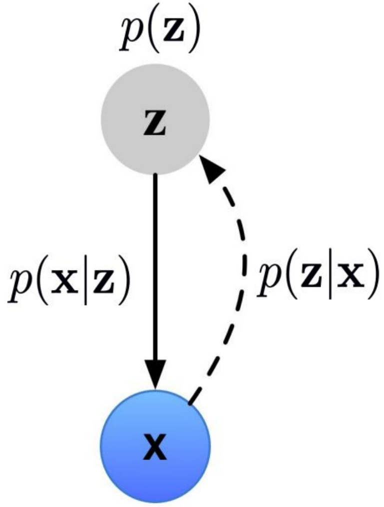
$$
\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right]
$$

https://storage.googleapis.com/deepmind-media/UCLxDeepMind_2020/L11%20-%20UCLxDeepMind%20DL2020.pdf

Rewriting the ELBO

The ELBO can be rewritten as the difference of two intractable terms:

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



The **Kullback–Leibler divergence** between two distributions is defined as:

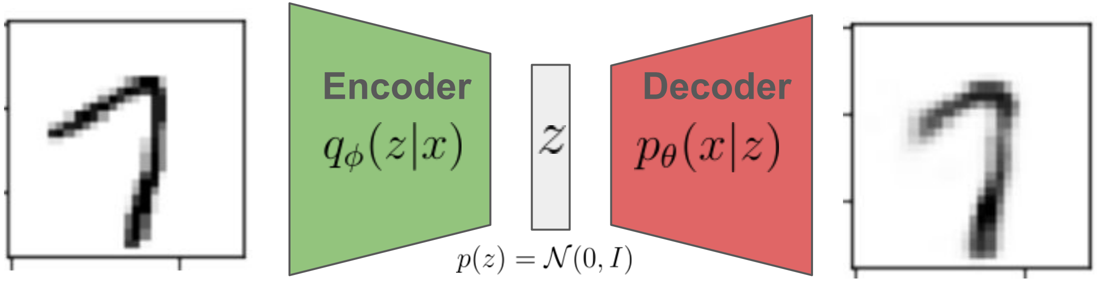
$$\mathbb{KL}(q(\mathbf{z})||p(\mathbf{z})) = \mathbb{E}_{q(\mathbf{z})}\left[\log \frac{q(\mathbf{z})}{p(\mathbf{z})}\right]$$

Maximizing the ELBO wrt to the variational parameters is equivalent to fitting the variational posterior to the true posterior

https://storage.googleapis.com/deepmind-media/UCLxDeepMind_2020/L11%20-%20UCLxDeepMind%20DL2020.pdf

Variational Auto-Encoders

A **Variational Autoencoder** parametrizes the approximate posterior with an encoder neural network and the likelihood with a decoder neural network



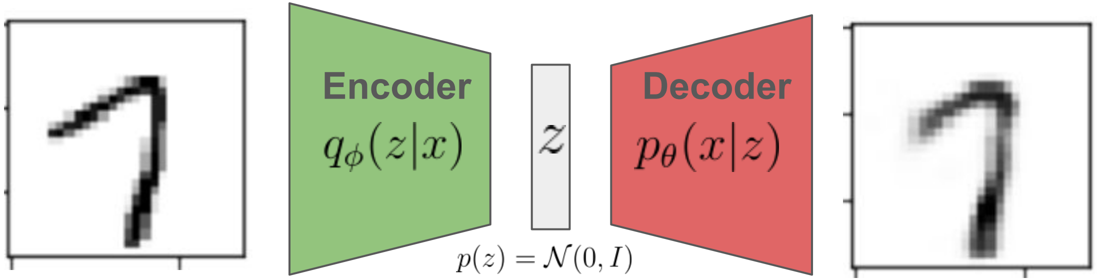
The following formulation is used to train a VAE by maximizing the ELBO:

$$\mathcal{L}_{\theta,\phi} = \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - \mathrm{KL}(q_\phi(z|x)||p(z))$$

For a dataset of N observations the loss becomes:

$$\sum_{i=1}^{N} \mathbb{E}_{z\sim q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)] - \mathrm{KL}(q_\phi(z|x^{(i)})||p(z))$$

Variational Auto-Encoders



In practice:

$$q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \mathrm{diag}(\sigma^2_\phi(x)))$$

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

Full Derivation:

https://stats.stackexchange.com/questions/318748/deriving-the-kl-divergence-loss-for-vaes/370048#370048

$$\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] = \mathbb{E}_{z\sim q_\phi(z|x)}\left[-\frac{1}{2\sigma^2}\|x - \mu_\theta(z)\|^2 - \frac{D}{2}\log(2\pi\sigma^2)\right]$$

It is not possible to sample directly from $q_\phi(z|x)$ so we use the reparametrization trick:

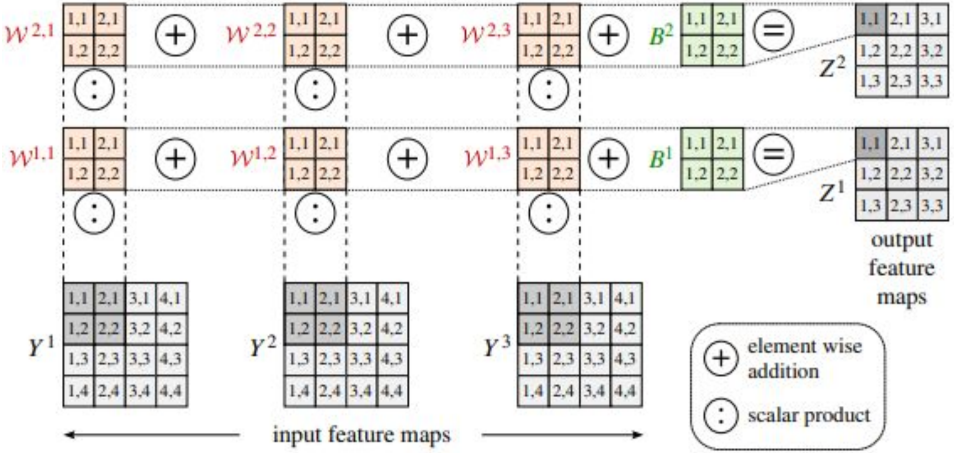
$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

Convolutional Neural Networks (CNNs)

Convolution Operation

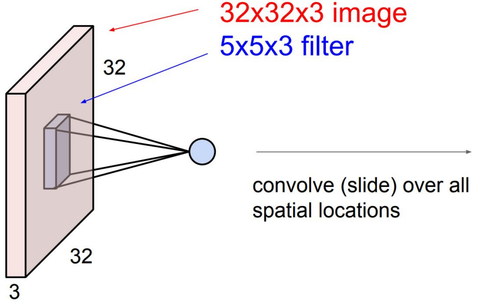
$$Z^m_{i,j} = \sigma(\sum_{l=1}^{C_{in}} \sum_{\alpha=1}^{k_x} \sum_{\beta=1}^{k_y} W^{m,l}_{\alpha,\beta} Y^c_{i+\alpha,j+\beta} + B^m_{\alpha,\beta}), \quad for \quad 1 \leq m \leq C_{out}, \quad 1 \leq i \leq W_{out}, \quad 1 \leq j \leq H_{out}$$

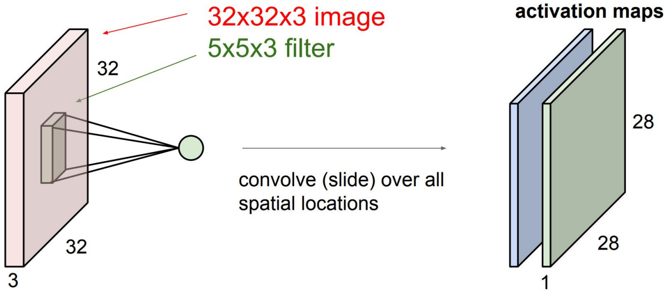
Filter Kernel



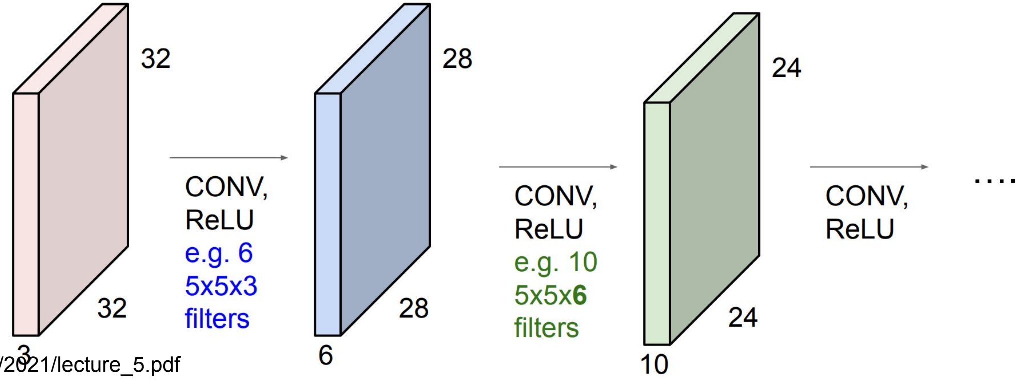
https://cs231n.stanford.edu/2021/slides/2021/lecture_5.pdf

Convolutional Neural Networks (CNNs)





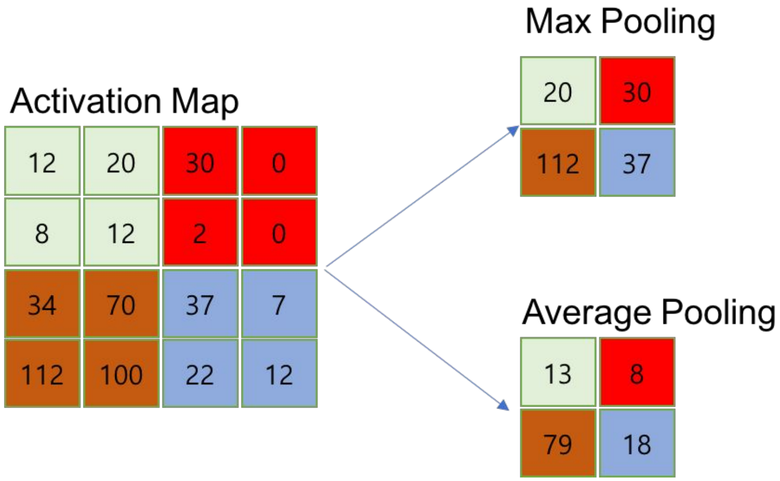
A CNN is a sequence of convolutional layers and non linear activation functions



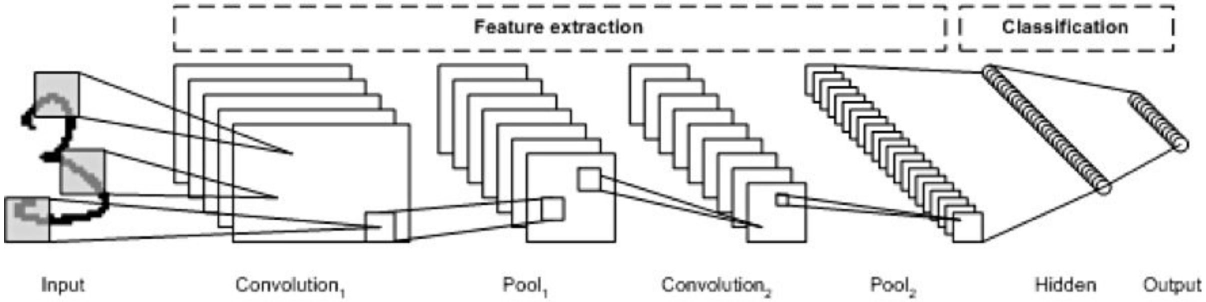
https://cs231n.stanford.edu/2021/slides/2021/lecture_5.pdf

Pooling in CNNs

Pooling is used in CNN to reduce the feature map dimension.



A Convolutional **Encoder** Network can be designed using Convolution and Pooling operators (and flattening).



Tutorial in pytorch: VAE for topology optimization

In this tutorial you will learn:

- How to implement a simple VAE in Pytorch to process images and reconstruct different topologies.
- How to implement a Convolutional Neural Net in Pytorch in the form of a VAE.
- How to use VAE for generative modeling and data exploration.

You can find the Notebooks in the Github repository: https://github.com/jomorlier/IA_CNRS_ICA