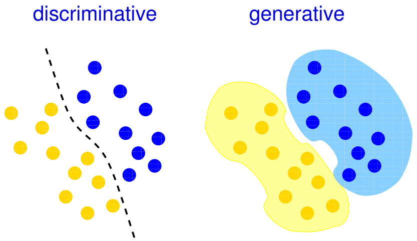
Generative Models for Imaging



Generative Models

Generative Model: a model that tries to learn the actual distribution p_{data} of real data from available samples (training set).

Goal: build probability distribution p_{model} close to p_{data}.

We can either try to

- explicitly estimate the distribution
- build a generator able to sample according to p_{model}

Generative models mostly focus on the second approach.

Why studying Generative Models

In many interesting cases there is **no unique intended solution** to a given problem:

- add colors to a gray-scale image
- guess the next word in a sentence
- fill a missing information
- increase the image resolution
- predict the next state/position in a game
- . . .

When the output is intrinsically **multi-modal** (and we do not want to give up to the possibility to produce multiple outputs) we need to rely on generative modeling.

Latent-variable Models

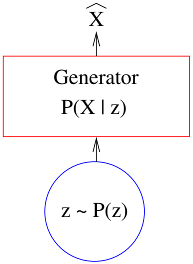
Neural Networks are **deterministic**, but generative tasks require producing diverse output.

In a latent-variabl model the generative process is modelled as:

$$P(X) = \int P(X|z)P(z)dz$$

where

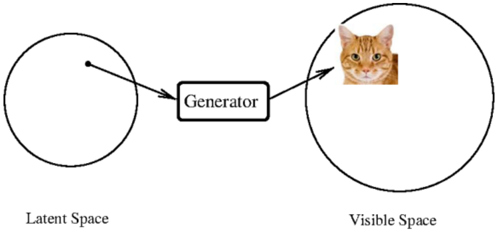
- $P(z)$ is the **Prior Distribution** over latent variables, typically a Gaussian distribution
- $P(X|z)$ is the likelyhood function, approximated by the deterministic Generator G(z).



Ancestral sampling

Sampling is typically a two stage process:

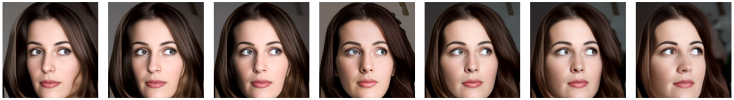
- Step 1: sample z from $P(z)$ (a **known** distribution)
- Step 2: pass z through the deterministic generator to produce X.



The latent space

- Points in the latent space are a compressed representation of the data.
- Each latent point contains all the information required to generate a different image (similar to dna code)
- Generators decode points in the latent space to produce output
- Generators are continuous: small modificcations of the latent representation results in small modifications of the output

Exploring the latent space



Picture from Illumination and Shadows in Head Rotation

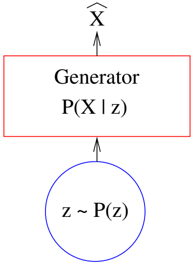
- The continuity of the generation process allows for the identification of **trajectories** for **editing** and **manipulation**
- any sample can potentially be generated; you simply need to find the right latent ecnoding to pass to the generator.

Next argument

Training the generator

Why training a generator is difficult?

- we need to model $P(X|z)$, which associates an image X with a latent variable z.
- Typically, neural networks learn functions in a supervised way, using pairs (z, X). However, in this case we don't have them.
- If we start with a real image, we need to determine its corresponding latent encoding
- If we start with a latent encoding, we must decide what inage to generate and identify the factors that could drive this generation.



Different classes of Generative Models

The different classes of Generative Models differ in the way the generator is trained

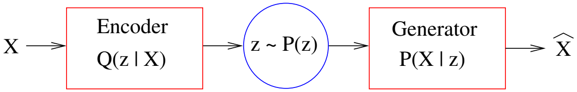
There shall discuss three main classes:

- **Variational Autoencoders** (VAEs)
- **Generative Adversarial Networks** (GANs)
- **Diffusion Models**

In the next slides we shall briefly introduce them, and then we shall discuss the different models in more detail.

VAE

In a Variational Autoencoder the generator is coupled with an **encoder** producing a latent encoding z given X. This will be distributed according to an inference distribution $Q(z|X)$.

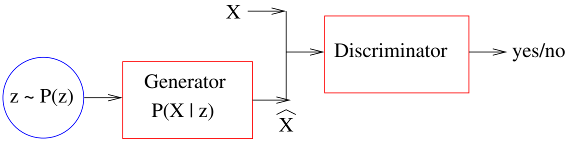


The loss function aims to:

- minimize the reconstruction error between X and $\widehat{X}$
- bring the marginal inference distribution $Q(z)$ close to the prior $P(z)$

GAN

In a Generative Adversarial Network, the generator is coupled with a **discriminator** trying to tell apart **real** data from **fake** data produced by the generator.



Detector and Generator are trained together.

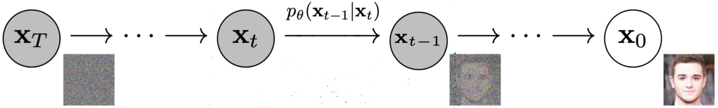
The loss function aims to:

- instruct the detector to spot the generator
- instruct the generator to fool the detector

Diffusion Models

In Diffusion Models the latent space is understood as a strongly noised version of the image to be generated.

The generator is split into a long chain of **denoising steps**, where each step t attempts to remove gaussian noise with a given variance σ_t.



We train a **single network** implementing the denoising operation, parametric in σ_t.

Next argument

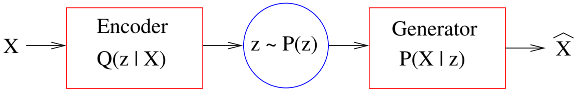
Variational Autoencoders

Suggested reading:

A survey on Variational Autoencoders from a GreenAI Perspective

VAE

In a Variational Autoencoder the generator is coupled with an **encoder** producing a latent encoding z given X. This will be distributed according to an inference distribution $Q(z|X)$.

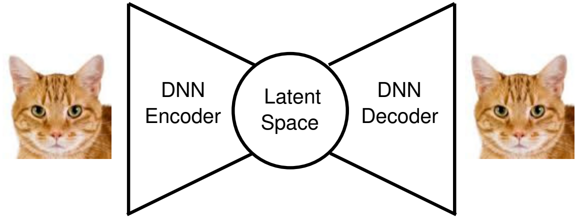


The loss function aims to:

- minimize the reconstruction error between X and $\widehat{X}$
- bring the marginal inference distribution $Q(z)$ close to the prior $P(z)$

The problem with the deterministic autoencoder

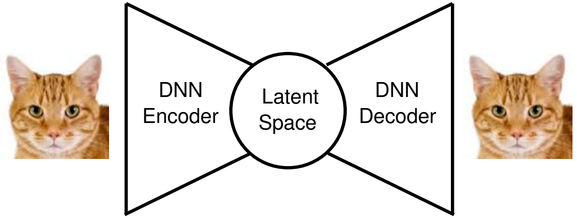
The VAEs approach is reminiscent of the autoencoder idea.
An autoencoder is a net trained to reconstruct input data out of a learned internal representation (e.g. minimizing quadratic distance)



Can we use the decoder to **generate** data by **sampling** in the latent space?
No, since we do not know the distribution of latent variables.

The problem with the deterministic autoencoder

The VAEs approach is reminiscent of the autoencoder idea.
An autoencoder is a net trained to reconstruct input data out of a learned internal representation (e.g. minimizing quadratic distance)

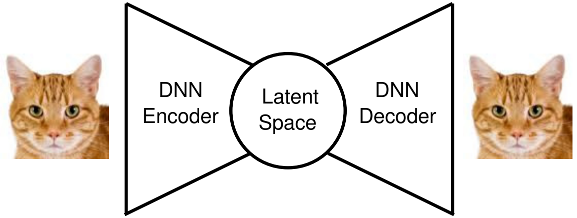


Can we use the decoder to **generate** data by **sampling** in the latent space?

No, since we do not know the distribution of latent variables.

The problem with the deterministic autoencoder

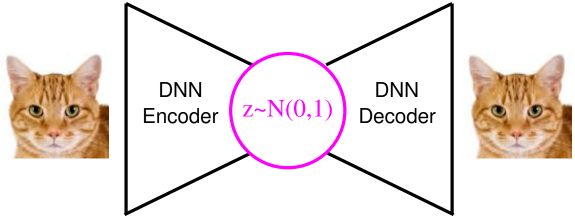
The VAEs approach is reminiscent of the autoencoder idea.
An autoencoder is a net trained to reconstruct input data out of a learned internal representation (e.g. minimizing quadratic distance)



Can we use the decoder to **generate** data by **sampling** in the latent space?
No, since we do not know the distribution of latent variables.

Variational autoencoder

In a Variational Autoencoder (VAE) we try **to force** latent variables to have a known prior distribution $P(z)$ (e.g. a Normal distribution)

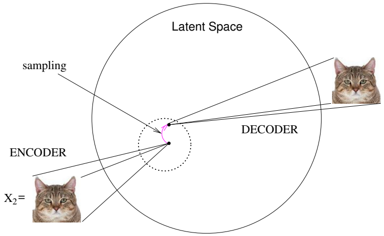


If the distribution computed by the generator is $Q(z|X)$ we try to force the marginal distribution $Q(z) = \mathbb{E}_{X \sim P_{data}} Q(z|X)$ to look like a normal distribution.

The VAE approach

Variational Autoencoders address the previous issue by introducing Gaussian Noise into the latent representation, bewteen the encoding and the decoding stages.

- The amount of noise (i.e. its variance) is **estimated** by the encoder and **learned** during training.
- A specific component of the loss function encourages this variance to be **as large as possible**



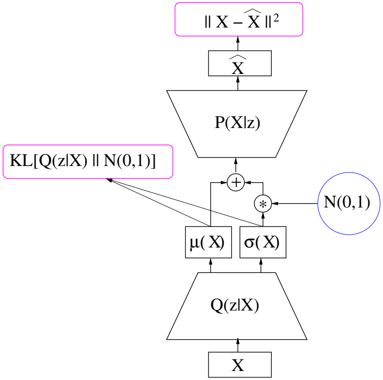
This ensures that the decoder is trained to generate meaningfull outputs for any point within the latent space.

The full picture

- The encoder returns
  - $\sigma(x)$: the scale of the Gaussian noise
  - $\mu(x)$: the actual encoding point
- The latent variable is resampled as

$$z = \mu(x) + \sigma(x) \cdot \epsilon$$

  where ϵ $N(0,1)$.
- z is passed as input to the decoder producing $\hat{X}$, a reconstruction of X



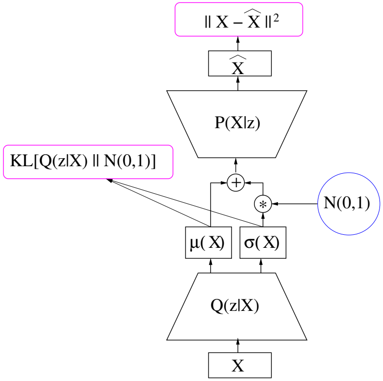
The VAE loss

The loss is a composition of two terms, with contrastive effects:

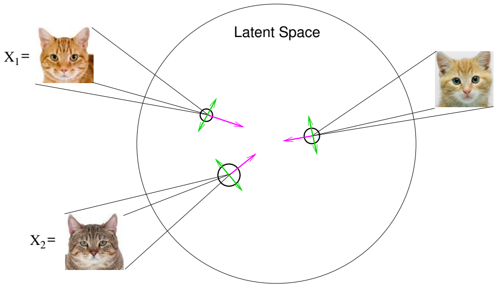
- A **reconstuction** loss, e.g.

$$\|X - \hat{X}\|^2$$

- A **regularization component**, enlarging the noise as much as possible (Kullback-Leibler component)



The effect of KL-divergence



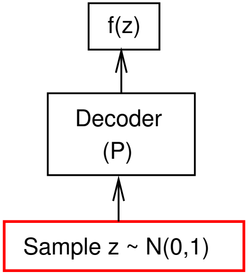
The effect of the KL-divergence on latent variables consist in

- pushing $\mu_z(X)$ towards 0, so as to center the latent space around the origin
- push $\sigma_z(X)$ towards 1, augmenting the "coverage" of the latent space, essential for generative purposes.

Generation of new samples

The encoder is only used to jointly train the decoder.

One training is completed, we only use the decoder for sampling.



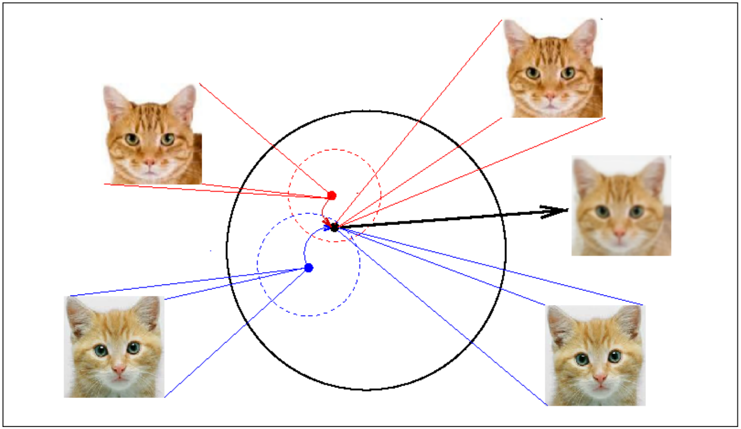
Next argument

Problems with VAEs

Problems with VAEs

- balancing loglikelihood and KL regularizer in the loss function
- variable collapse phenomenon
- marginal inference vs prior mismatch
- blurriness (aka variance loss)

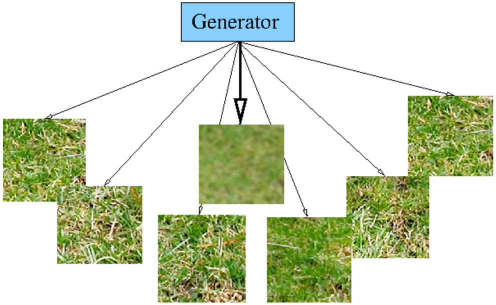
Blurriness



The patch of grass problem

Blurriness is particularly problematic in cases of images wigh high intrinsic variability, such patches of grass, leaves, or hair.

This type of information is essentially incompressible, and VAEs struggle to extract the main factors of variation.



Next argument

A bit of theory

A bit of theory

In a Vae there are two main distrbutions of interest:

- the encoder distribution $q_\phi(z|x)$ (also called inference distribution)
- the true distribution $p_\theta(z|x))$

we can take as our learning objective the minimization of their distance, e.g. measured by Kullback-Leibler divergence:

$$\arg\min_{\theta,\phi} \mathop{\mathbb{E}}_{\mathbb{D}}[D_{KL}(q_\phi(z|x)||p_\theta(z|x))]$$

Let us expand this distance for a given x.

Evidence Lower Bound (ELBO)

$$
\begin{aligned}
&D_{KL}(q_\phi(z|x)||p_\theta(z|x)) \\
&= \mathbb{E}\, q_\phi(z|x)[\log q_\phi(z|x) - \log p_\theta(z|x)] \\
&= \mathbb{E}\, q_\phi(z|x)[\log q_\phi(z|x) - \log p_\theta(x|z) - \log p_\theta(z) + \log p_\theta(x)] \\
&= D_{KL}(q_\phi(z|x)||p(z)) - \mathbb{E}\, q_\phi(z|x)[\log p_\theta(x|z)] + \log p_\theta(x)
\end{aligned}
$$

Thus

$$
\begin{aligned}
\overbrace{\mathbb{E}\, q_\phi(z|x)[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))}^{ELBO} &= \log p_\theta(x) - D_{KL}(q_\phi(z|x)||p_\theta(z|x)) \quad (*) \\
&\leq \log p_\theta(x)
\end{aligned}
$$

Evidence Lower Bound (ELBO)

The VAE learning objective is the maximization of the ELBO.

This goal, will concurrently optimize two things:

1. It will approximately maximize the marginal likelihood $p_\theta(x)$, meaning that that the generative model will improve the decoder performance;
2. it will minimize the KL divergence of the approximation $q_\phi(z|x)$ from the true posterior $p_\theta(z|x)$, improving the encoder performance.