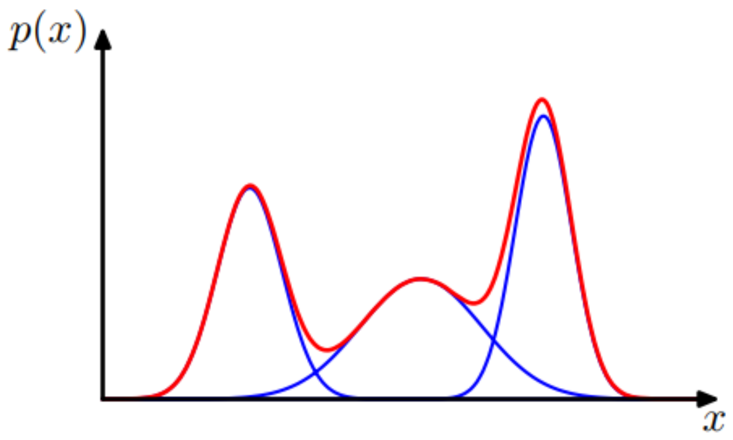
Generative neural networks

Romain Tavenard (Université de Rennes)

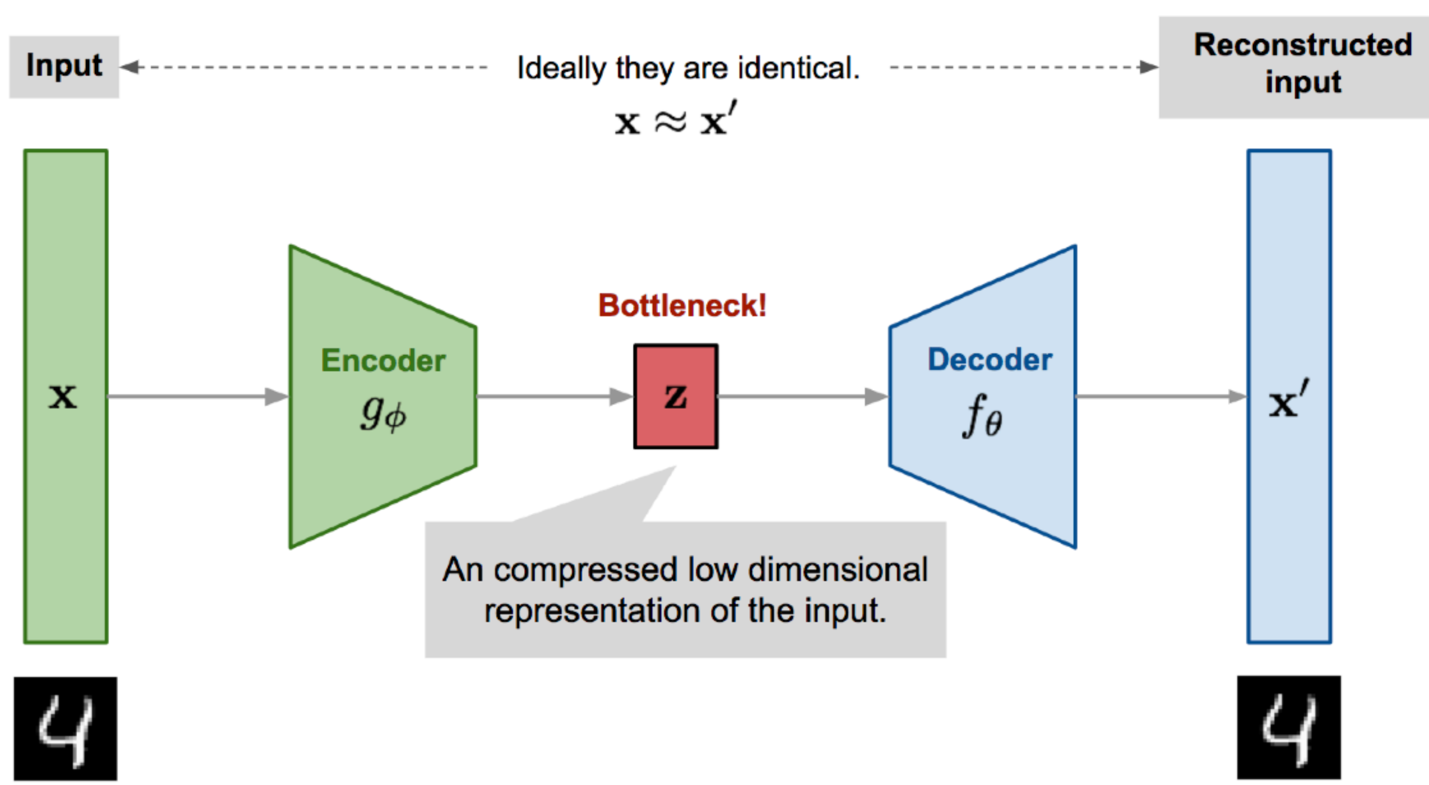
Generative models in a nutshell

- Goal: model $p(x)$ or $p(x|y)$
  - explicitly
    - *eg.* Gaussian Mixture Models
  - **implicitly**
    - at least allow for sampling (*i.e.* generate new data)
    - *e.g.* Variational Auto Encoders,
      Generative Adversarial Networks



Auto-encoders
[Hinton & Salakhutdinov, 2006]

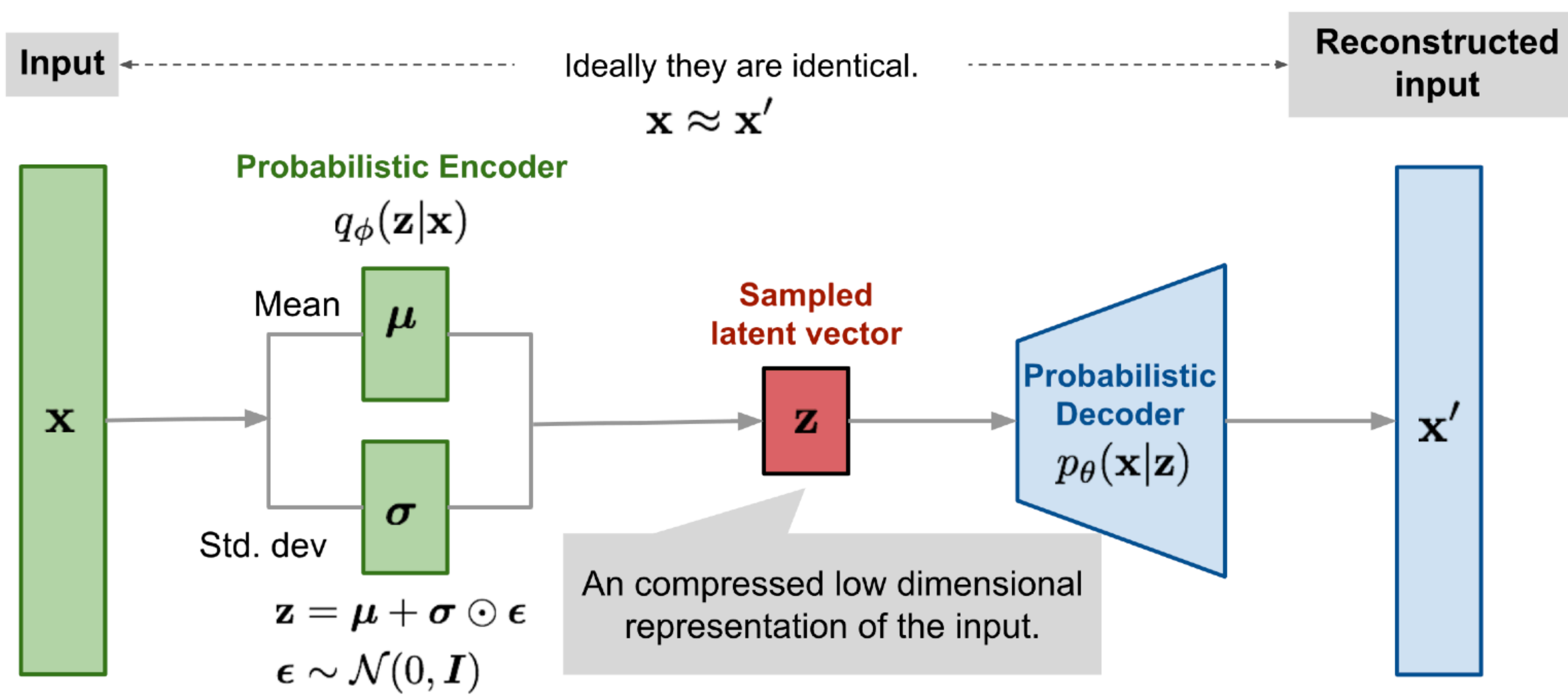
- Encode information in a latent space
  - typically lower-dimensional
  - similar in spirit to a non-linear PCA
  - not a generative model *per se*!



Source: lilianweng.github.io

Variational auto-encoders [Kingma & Welling, 2014]

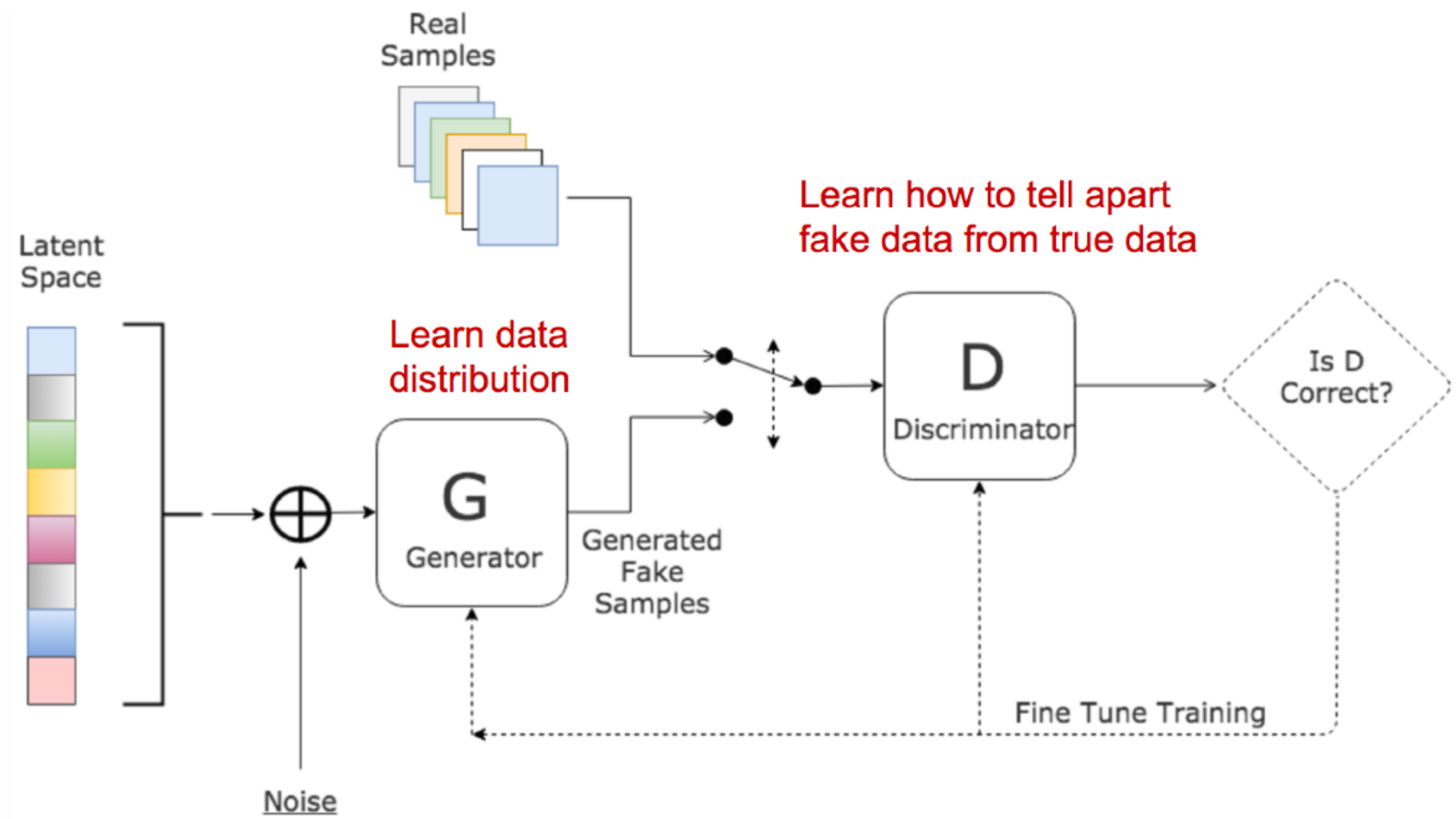
- <u>Goal:</u> turn auto-encoders into generative models
- <u>Idea:</u> set a prior on the distribution of z
(through penalization of the loss function)
- <u>Generative process:</u> draw z from the prior, and decode it



Source: <u>lilianweng.github.io</u>

Generative Adversarial Networks [Goodfellow *et al.*, 2014]

- Model:



Source: lilianweng.github.io

- Loss function:

$$\min_G \max_D L(D, G) = \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

$$= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_z(x)}[\log(1 - D(x))]$$

Generative Adversarial Networks [Goodfellow *et al.*, 2014]

- Generative process
  - Draw z from p_z
  - Pass it to the generator to compute $G(z)$
- Optimization
  - Alternate between generator and discriminator
  - Very unstable process in practice

$$\begin{aligned}\min_G \max_D L(D, G) &= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \\ &= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_z(x)}[\log(1 - D(x))]\end{aligned}$$

Generative Adversarial Networks

- Many variants to the original model
  - Class-conditional variants
  - Different losses
  - Different structures
- Very realistic samples generated (BigGAN, StyleGAN)