Machine Learning

in 2018… Deep Learning!

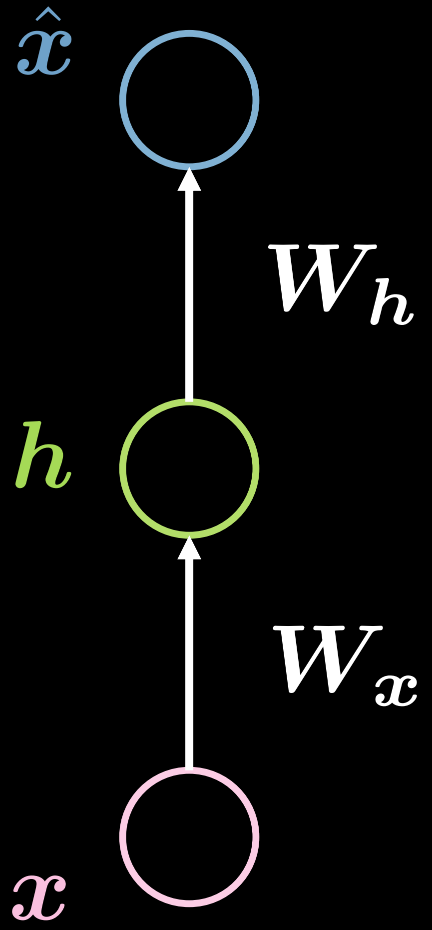
Alfredo Canziani, Alexey Svyatkovskiy
@AlfredoCanziani, @asvyatko

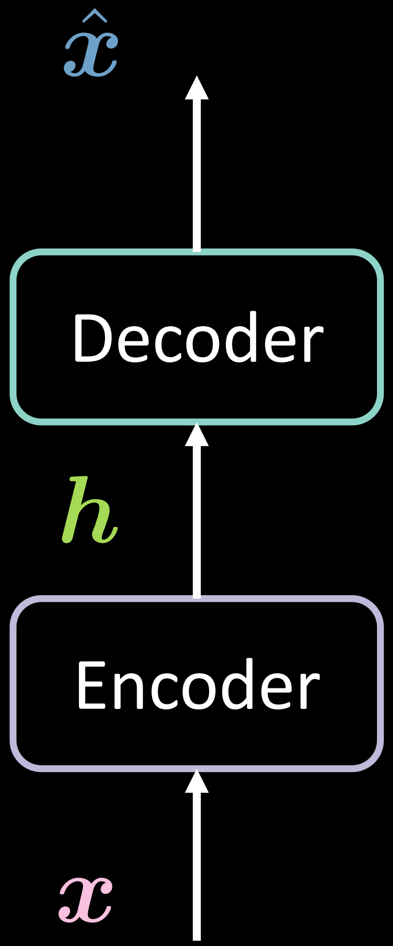
ALF

Autoencoders

Unsupervised learning / Generative models

Autoencoder





$$\boldsymbol{h} = f(\boldsymbol{W_x x} + \boldsymbol{b_x})$$

$$\hat{\boldsymbol{x}} = g(\boldsymbol{W_h h} + \boldsymbol{b_h})$$

$$\boldsymbol{x}, \hat{\boldsymbol{x}} \in \mathbb{R}^n$$

$$\boldsymbol{h} \in \mathbb{R}^d$$

$$\boldsymbol{W_x} \in \mathbb{R}^{d \times n}$$

$$\boldsymbol{W_h} \in \mathbb{R}^{n \times d}$$

If “tight weights”, then

$$\boldsymbol{W_h} \doteq \boldsymbol{W_x}^\top$$

Reconstruction losses

$$\mathcal{L} = \frac{1}{m}\sum_{j=1}^{m} \ell(\boldsymbol{x}^{(j)}, \tilde{\boldsymbol{x}}^{(j)})$$

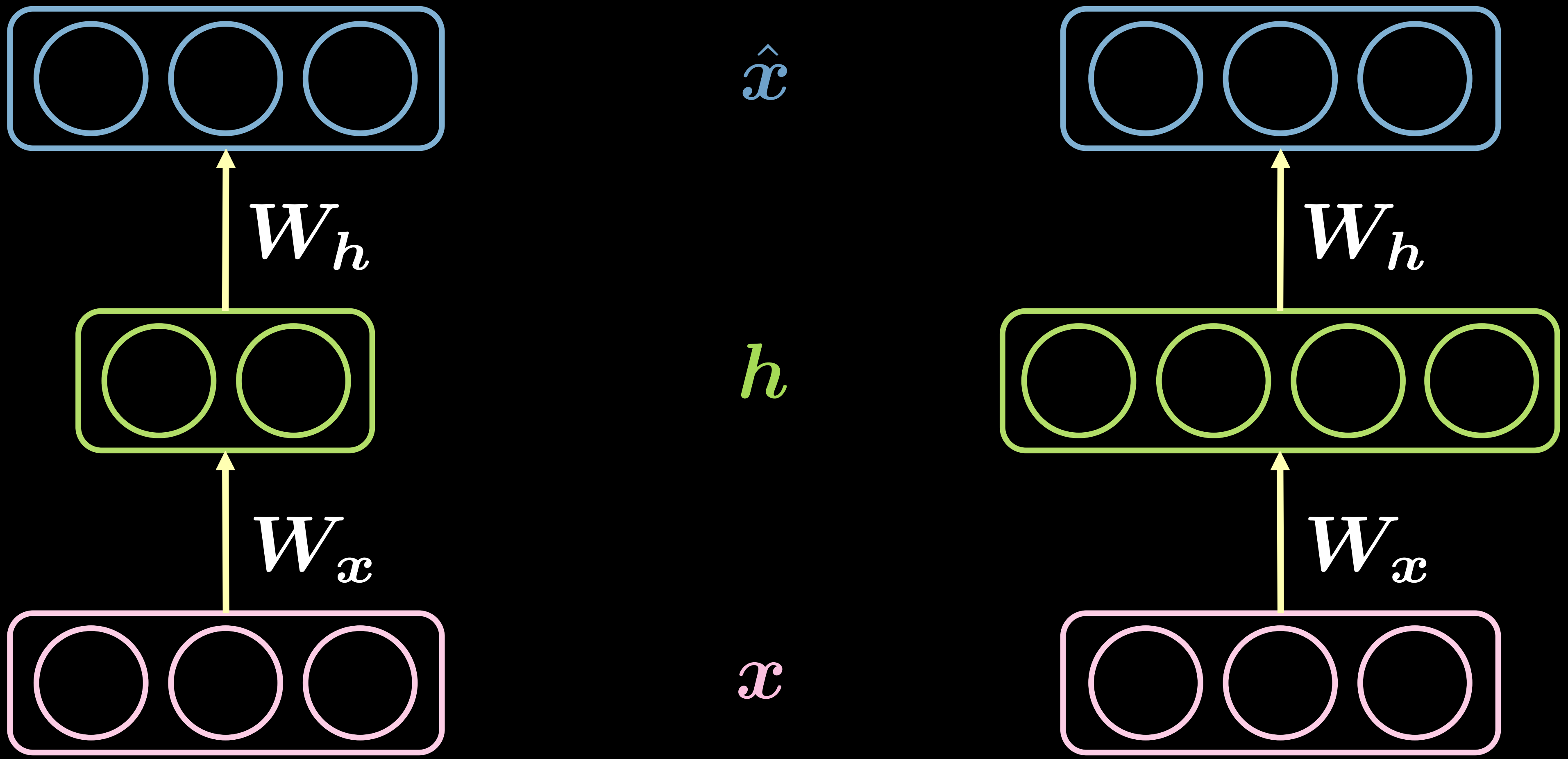
binary input

$$\ell(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = -\sum_{i=1}^{n} [x_i \log(\hat{x}_i) + (1 - x_i)\log(1 - \hat{x}_i)]$$

real valued input

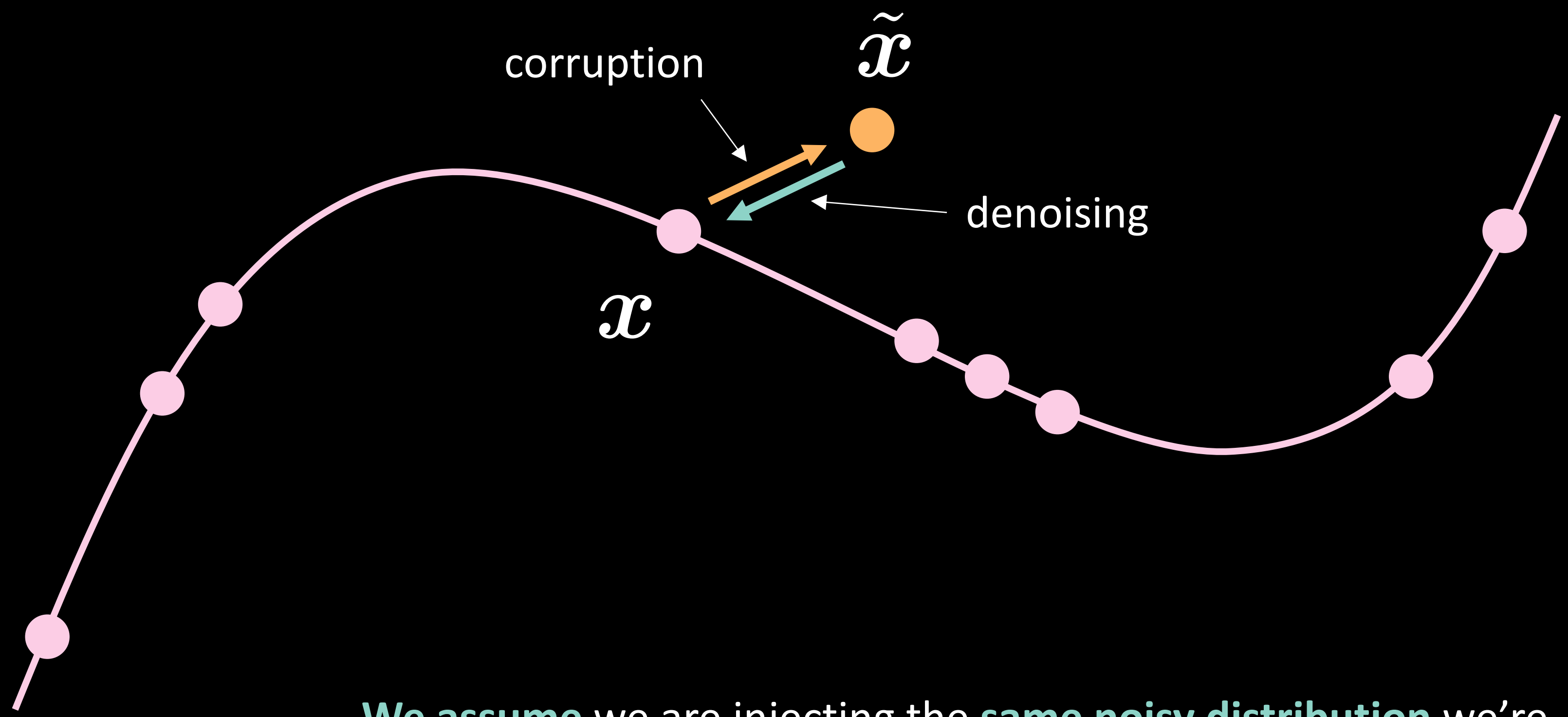
$$\ell(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \frac{1}{2}\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2$$

Under-/over-complete hidden layer



Denoising autoencoder

$$\tilde{\boldsymbol{x}} \sim p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$$



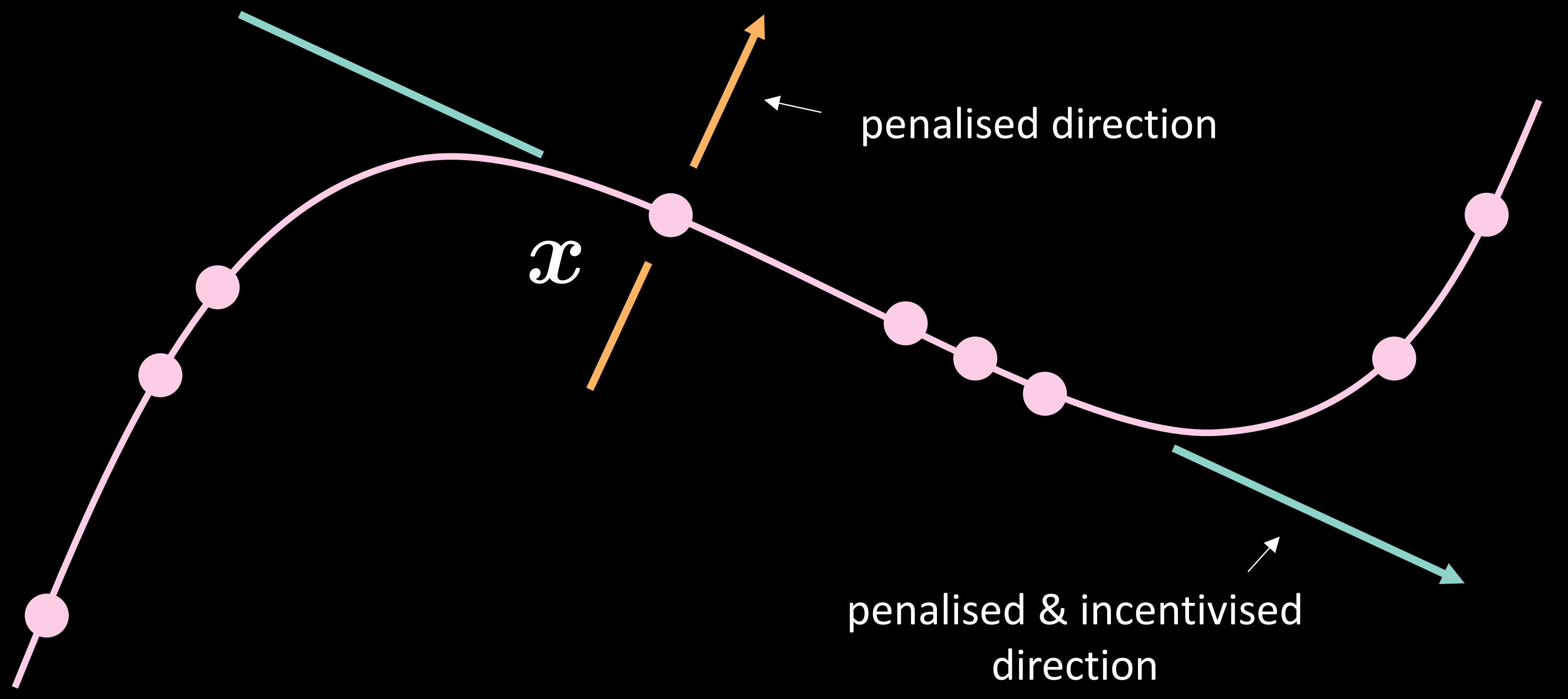
We assume we are injecting the **same noisy distribution** we're going to observe in reality. In this way, we can learn how to robustly recover from it.

Contractive autoencoder

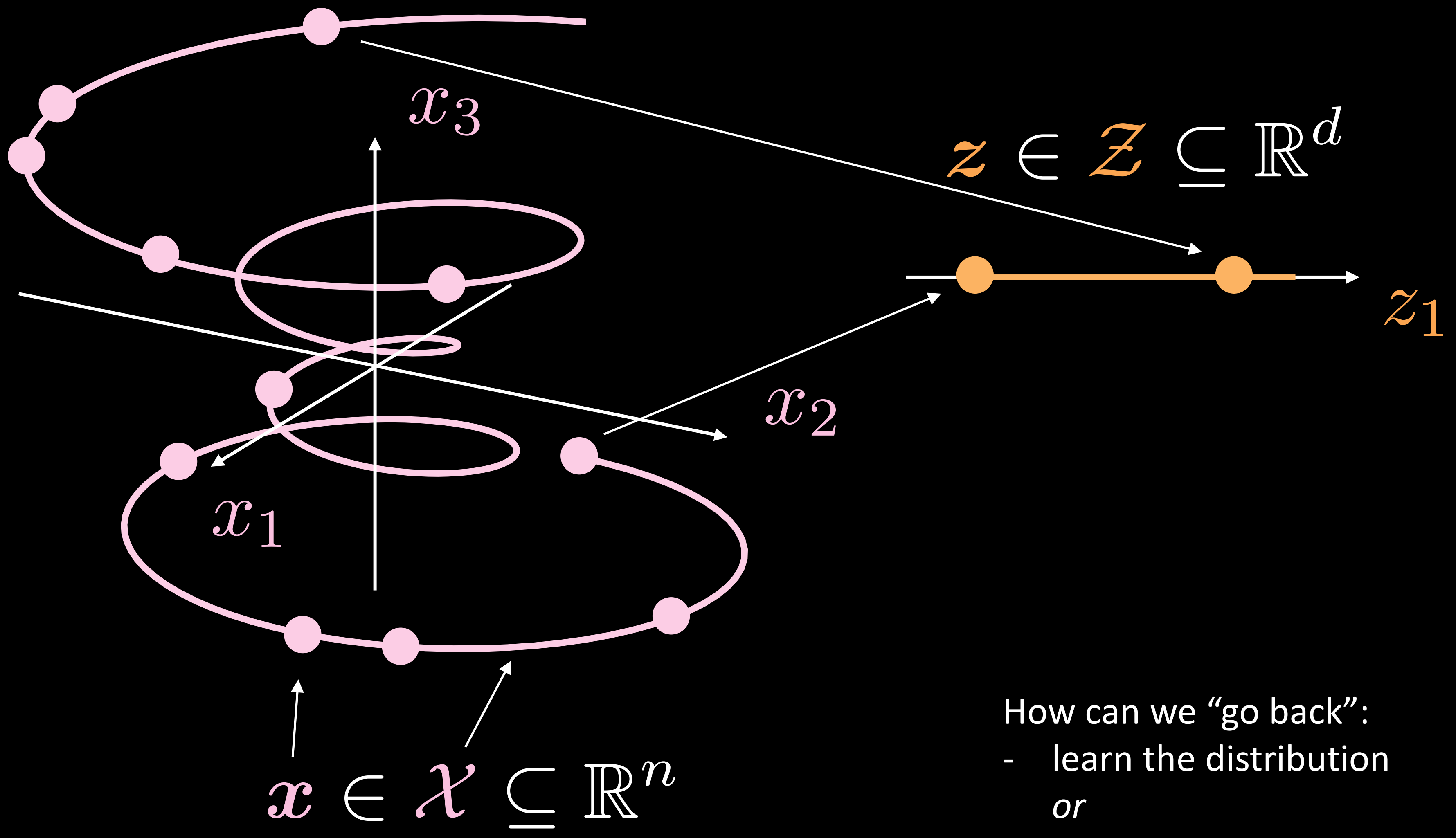
penalises insensitivity to **reconstruction directions**

penalises sensitivity to the **any direction**

$$\ell(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \boxed{\ell_{\text{reconstruction}}} + \boxed{\lambda \|\nabla_{\boldsymbol{x}} \boldsymbol{h}\|^2}$$

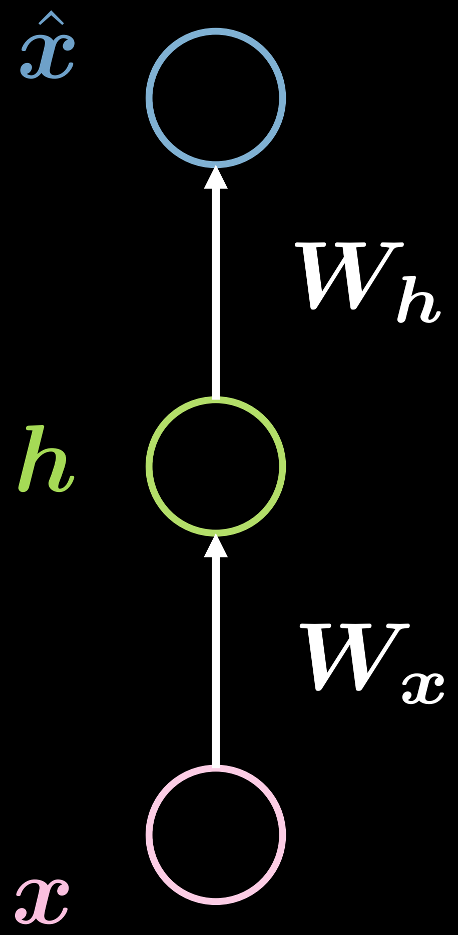


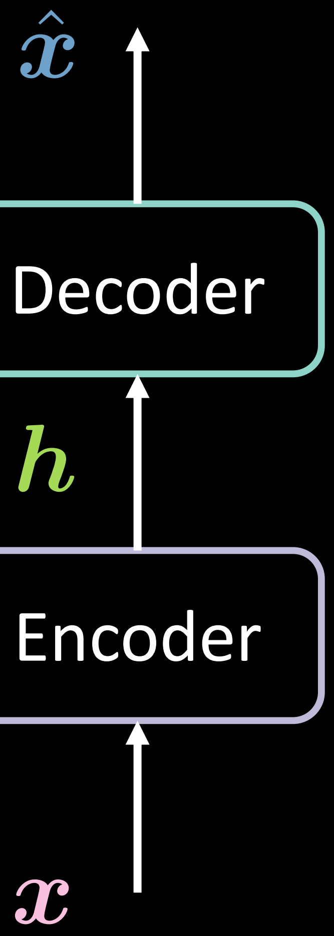
Basic auto-encoder



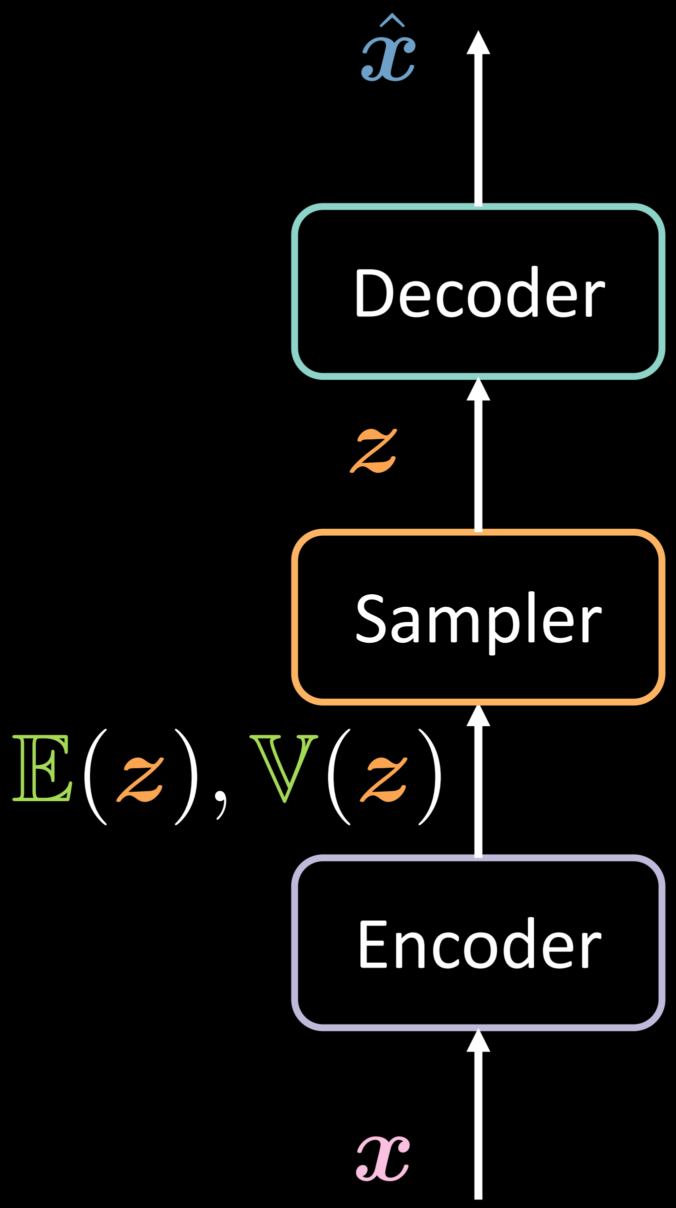
How can we “go back”:
- learn the distribution
  or
- enforce some structure

Auto-encoder (recap)

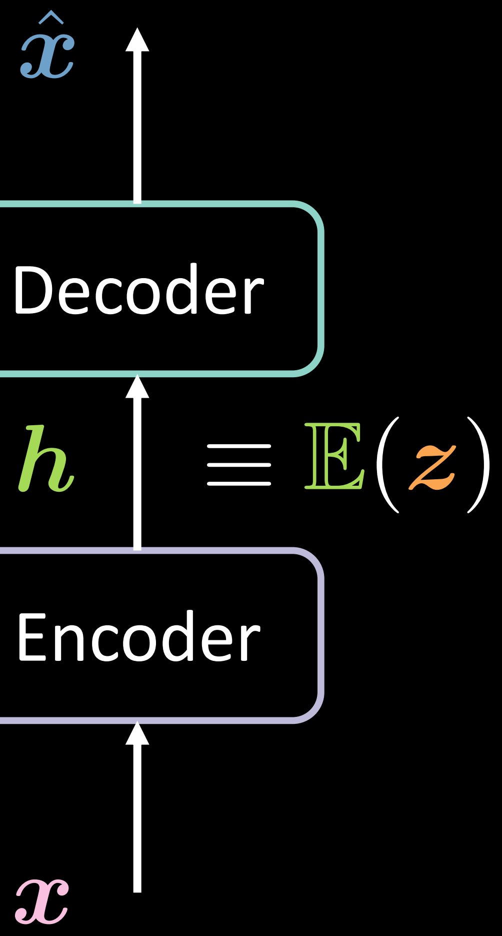




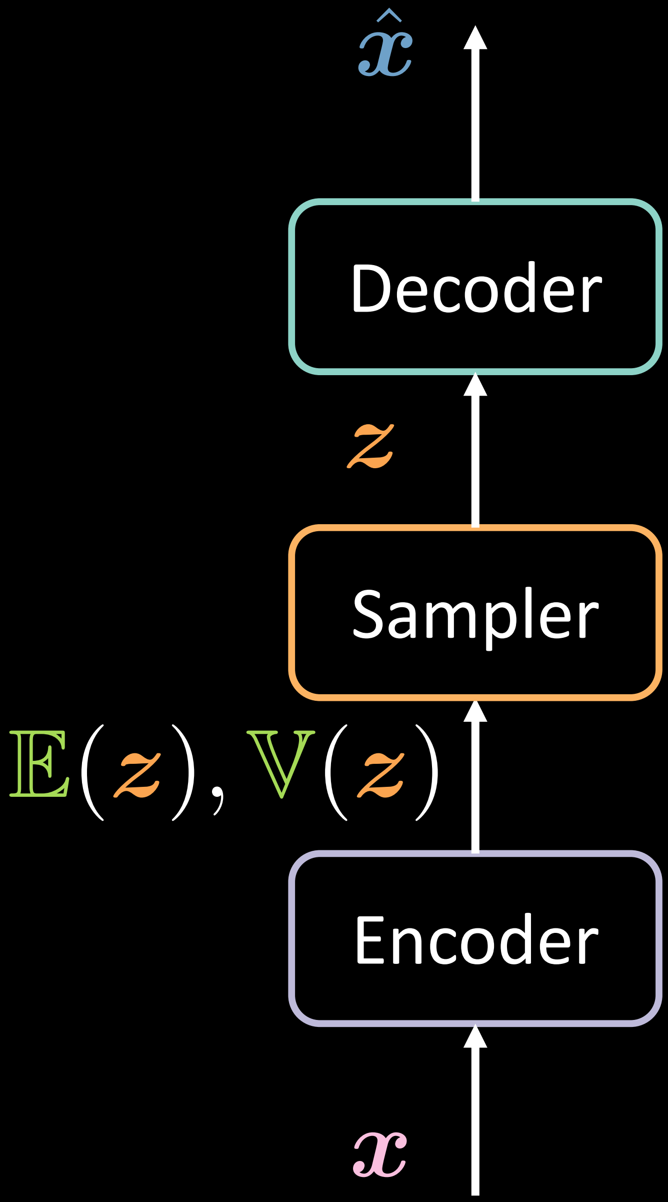
Variational auto-encoder



Classic auto-encoder



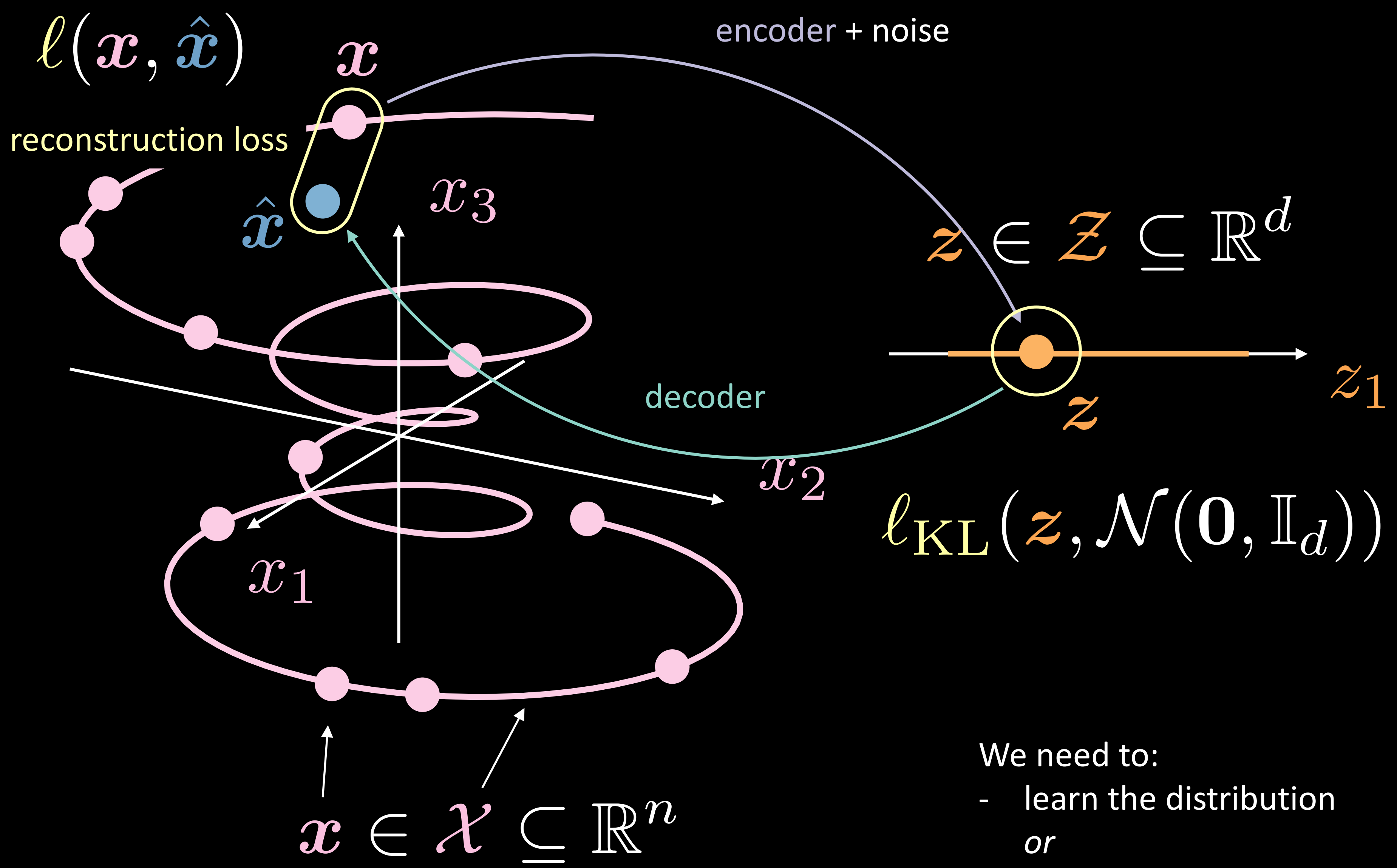
Variational auto-encoder



$$\text{decoder} : \mathcal{Z} \to \mathbb{R}^n$$
$$z \mapsto \hat{x}$$

$$\text{encoder} : \mathcal{X} \to \mathbb{R}^{2d}$$
$$x \mapsto h$$

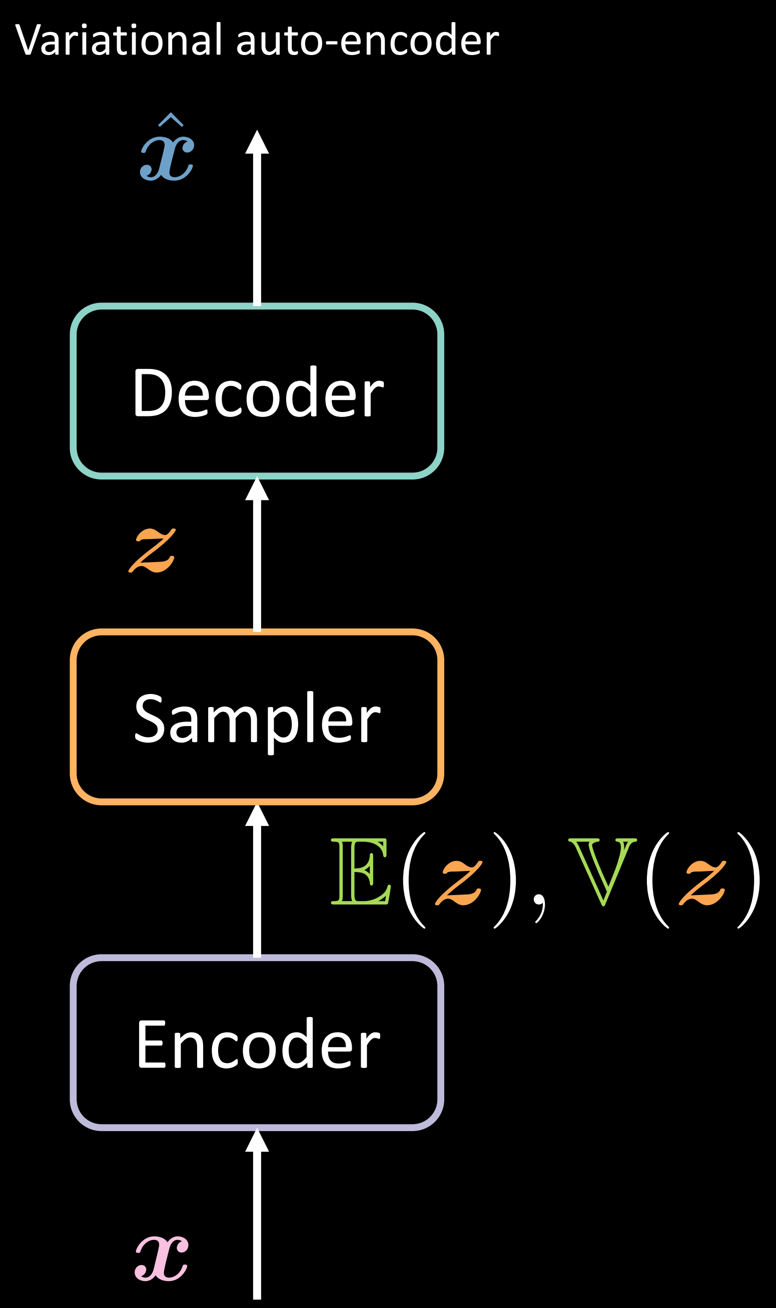
Variational auto-encoder

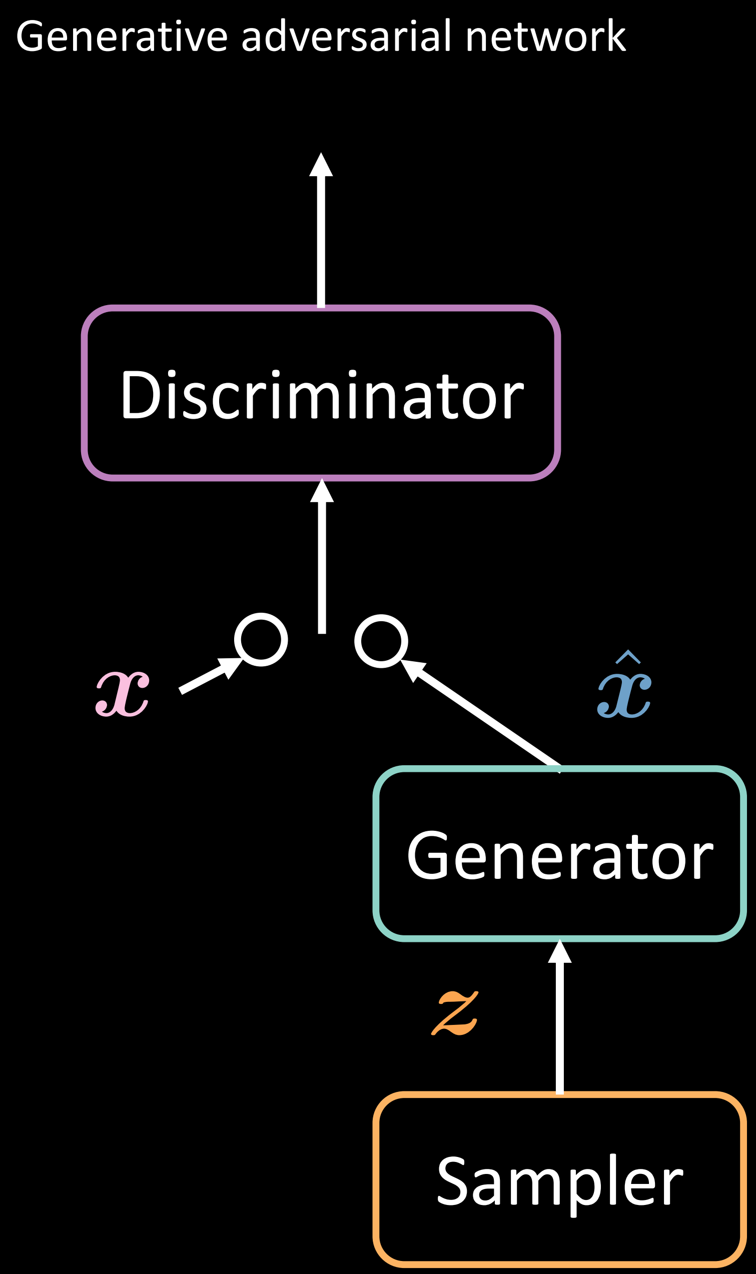


We need to:
- learn the distribution
  or
- enforce some structure

Generative adversarial nets

Unsupervised learning / Generative models


Variational auto-encoder
$\hat{x}$
Decoder
z
Sampler
$\mathbb{E}(z), \mathbb{V}(z)$
Encoder
x


Generative adversarial network
Discriminator
x
$\hat{x}$
Generator
z
Sampler

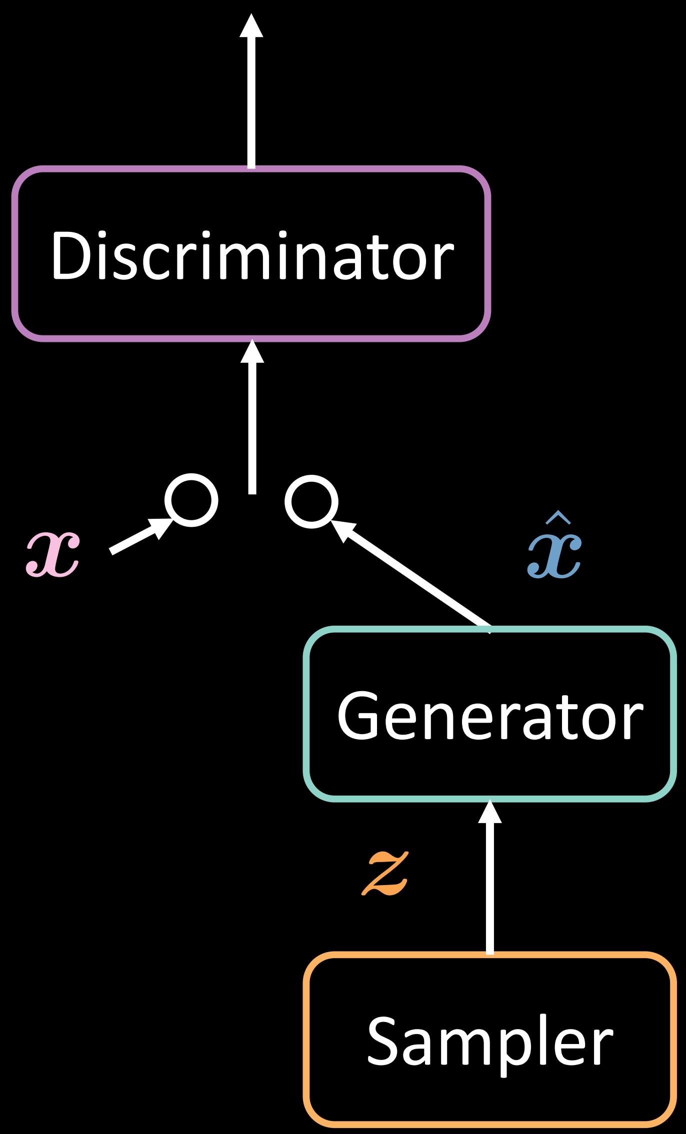
$$D : \mathbb{R}^n \to (0, 1)$$

$$\boldsymbol{x} \vee \hat{\boldsymbol{x}} \mapsto \ell$$

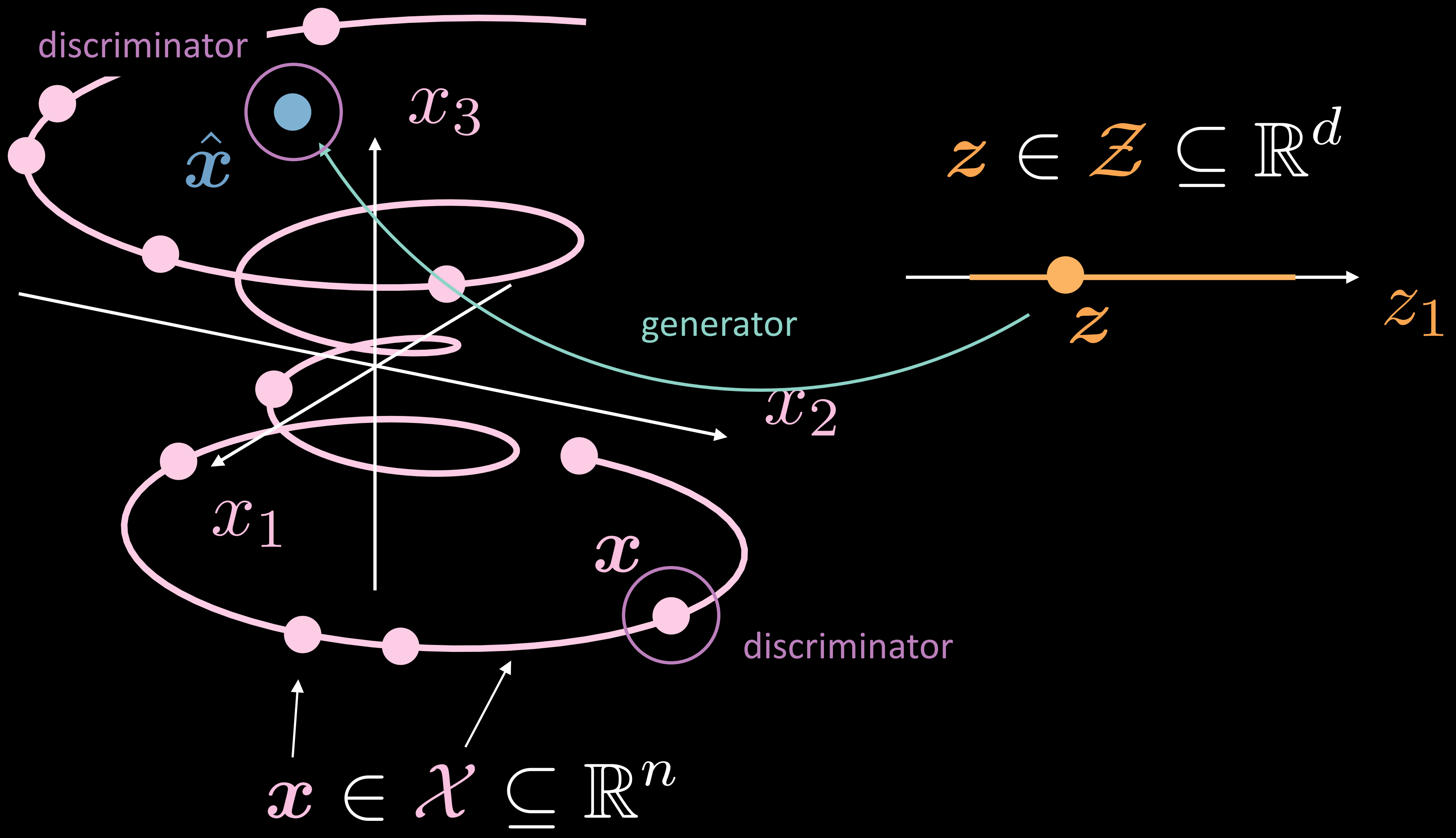
$$G : \mathcal{Z} \to \mathbb{R}^n$$

$$\boldsymbol{z} \mapsto \hat{\boldsymbol{x}}$$

Generative adversarial network



Generative adversarial network



Value function

$$V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \\ + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D[G(\boldsymbol{z})])]$$

$$\min_{G} \max_{D} V(D, G)$$

Goodfellow *et al.* (2014) Generative adversarial nets