Foundations of DL

Deep Learning

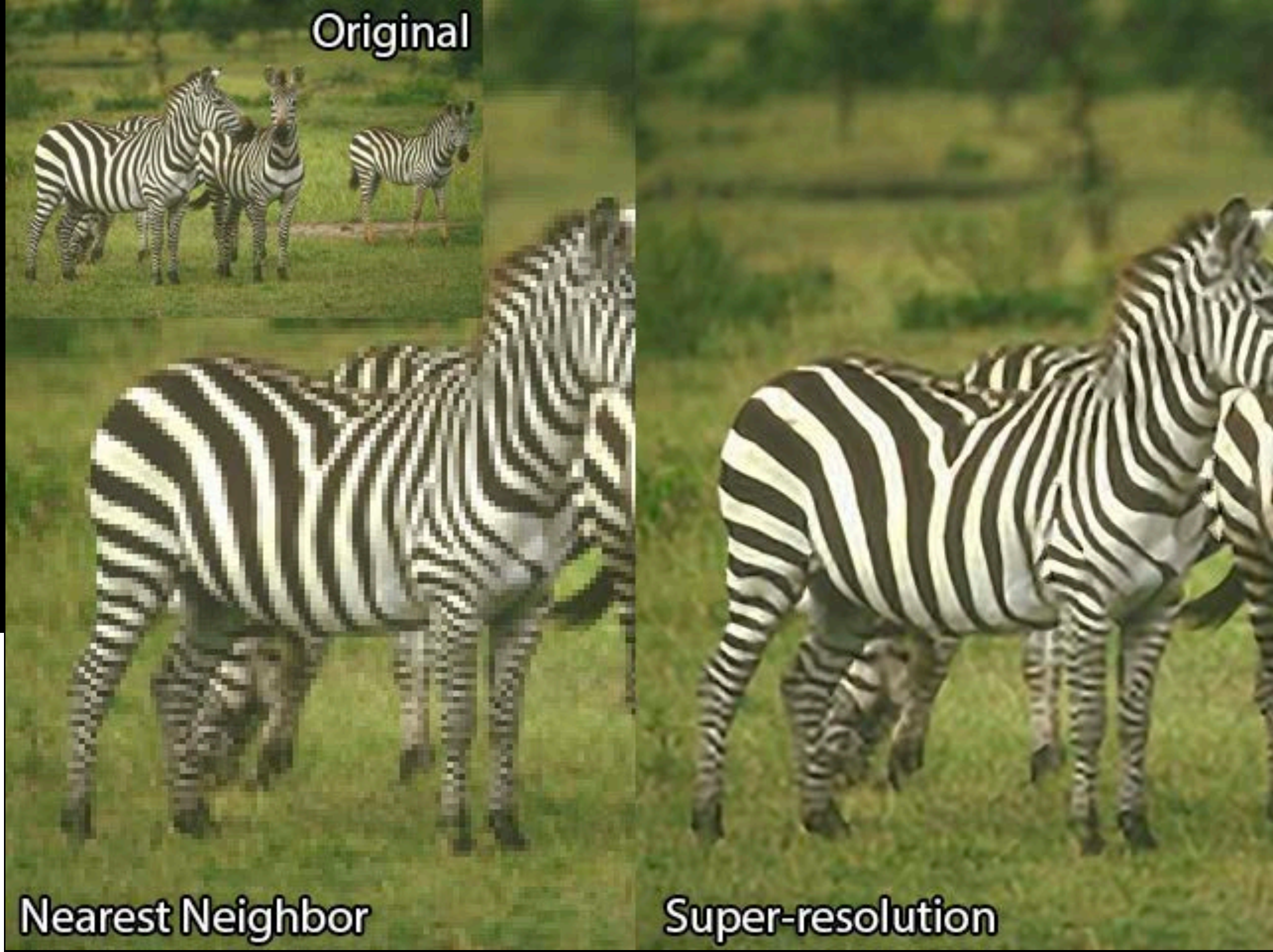
ALF

Alfredo Canziani, Ritchie Ng
@alfcnz, @RitchieNg

Generative models

(Semi-) Unsupervised learning

Super resolution

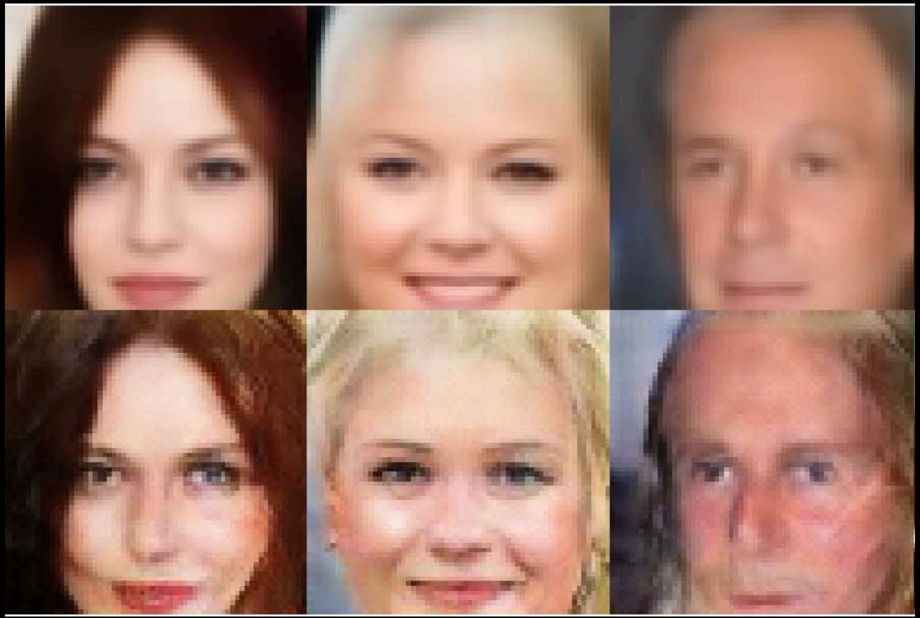


Glasner (2009) Super-resolution from a single image
www.wisdom.weizmann.ac.il/~vision/SingleImageSR.html

Garcia (2016) srez --- github.com/david-gpu/srez

Inpainting

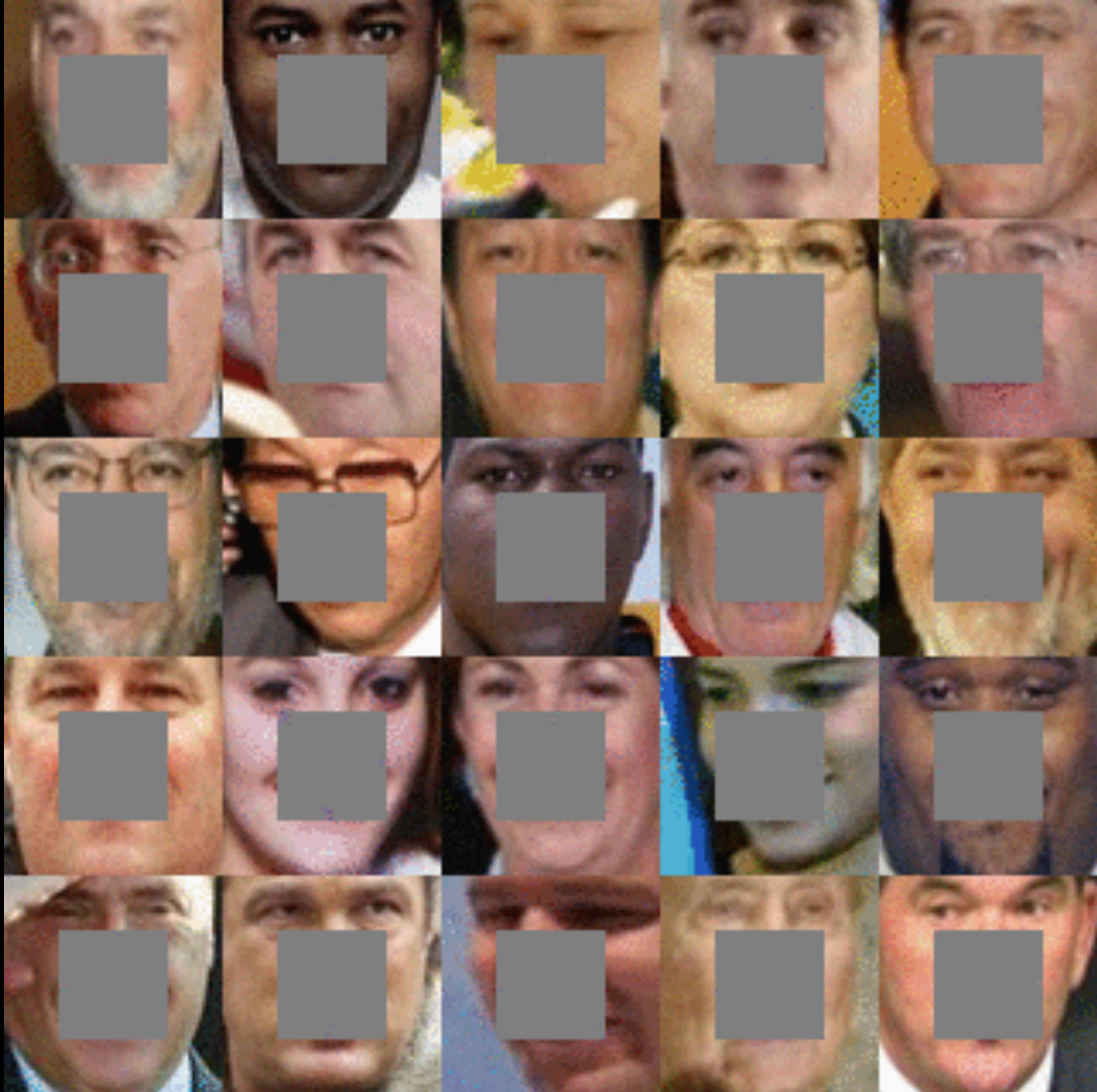
VAE



GAN

Yeh (2017)
Semantic Image Inpainting
with Deep Generative Models

bit.ly/DCGAN-inpainting



Caption to image

This vibrant red bird has a pointed black beak

This bird is yellowish orange with black wings

The bright blue bird has a white coloured belly

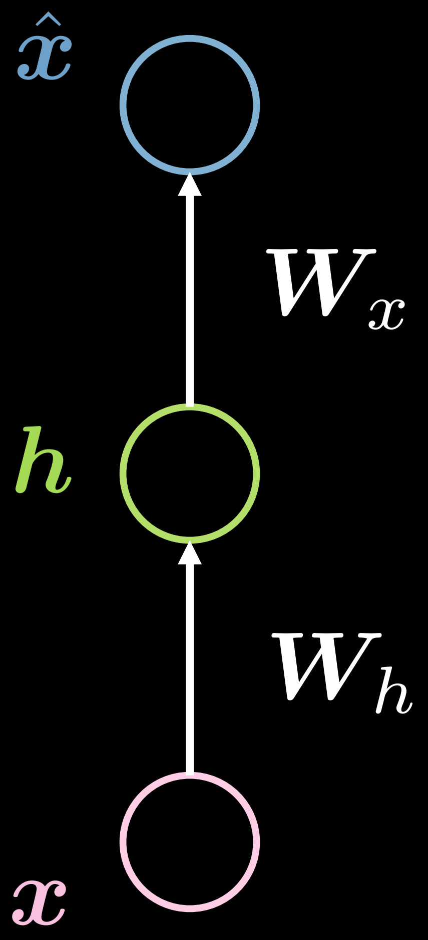
Reed (2016) Generative adversarial text to image synthesis

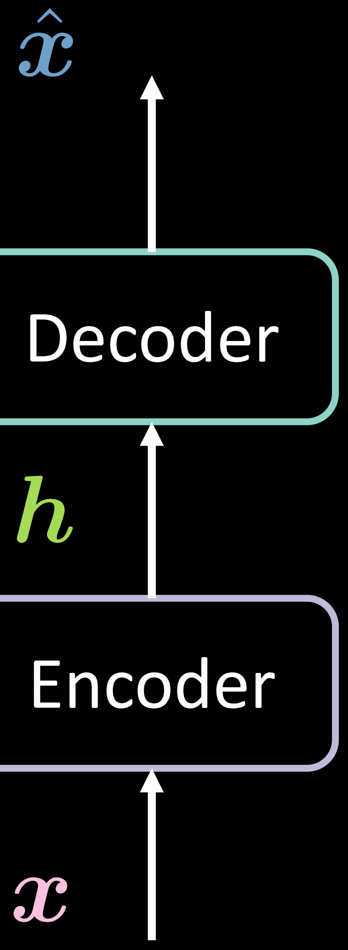
github.com/reedscot/icml2016

Autoencoders

Unsupervised learning / Generative models

Autoencoder





$$\boldsymbol{h} = f(\boldsymbol{W}_h \boldsymbol{x} + \boldsymbol{b}_h)$$

$$\hat{\boldsymbol{x}} = g(\boldsymbol{W}_x \boldsymbol{h} + \boldsymbol{b}_x)$$

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

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

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

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

If “tight weights”, then

$$\boldsymbol{W}_x \doteq \boldsymbol{W}_h^\top$$

Reconstruction losses

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

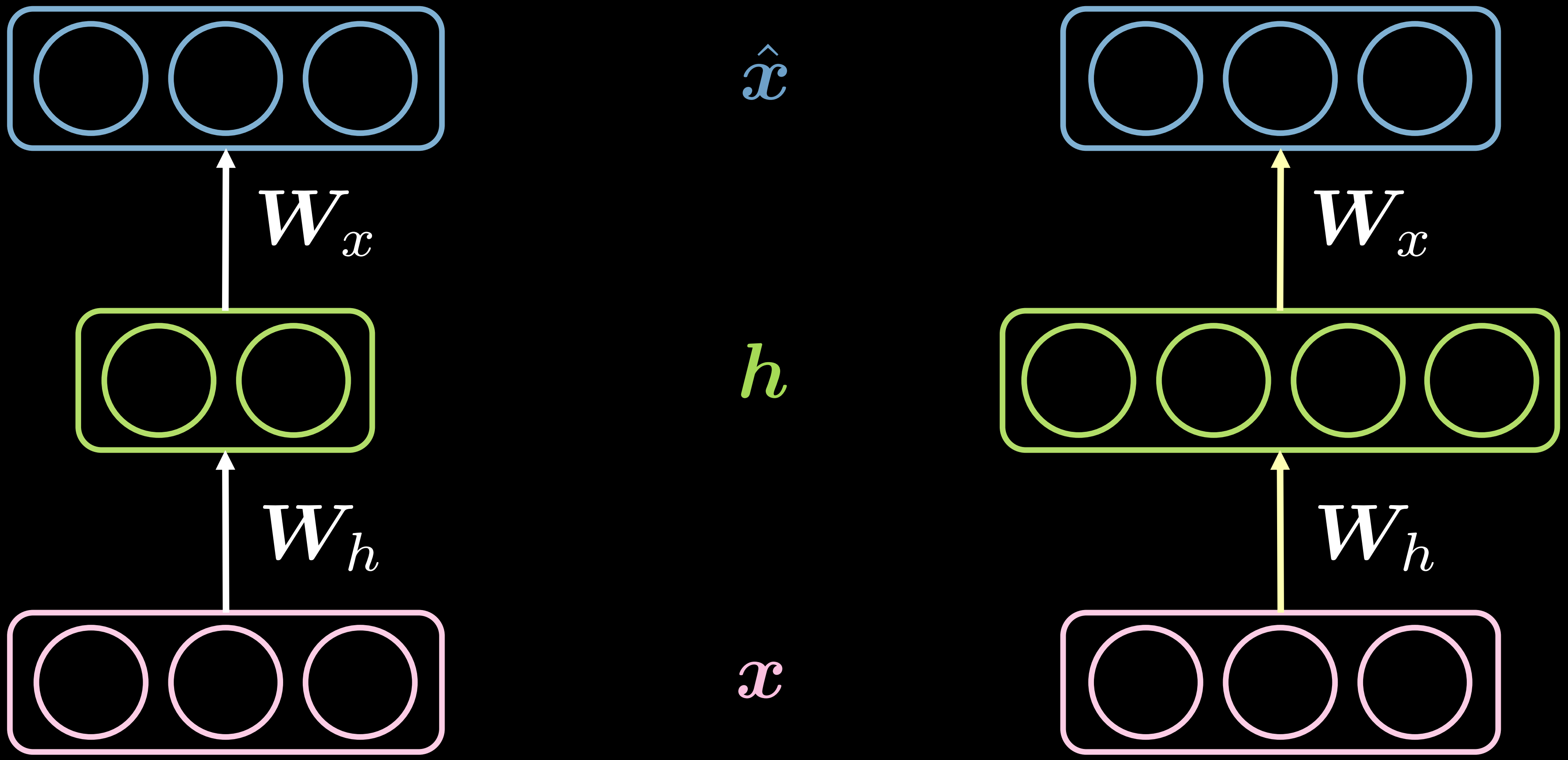
binary input

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

real valued input

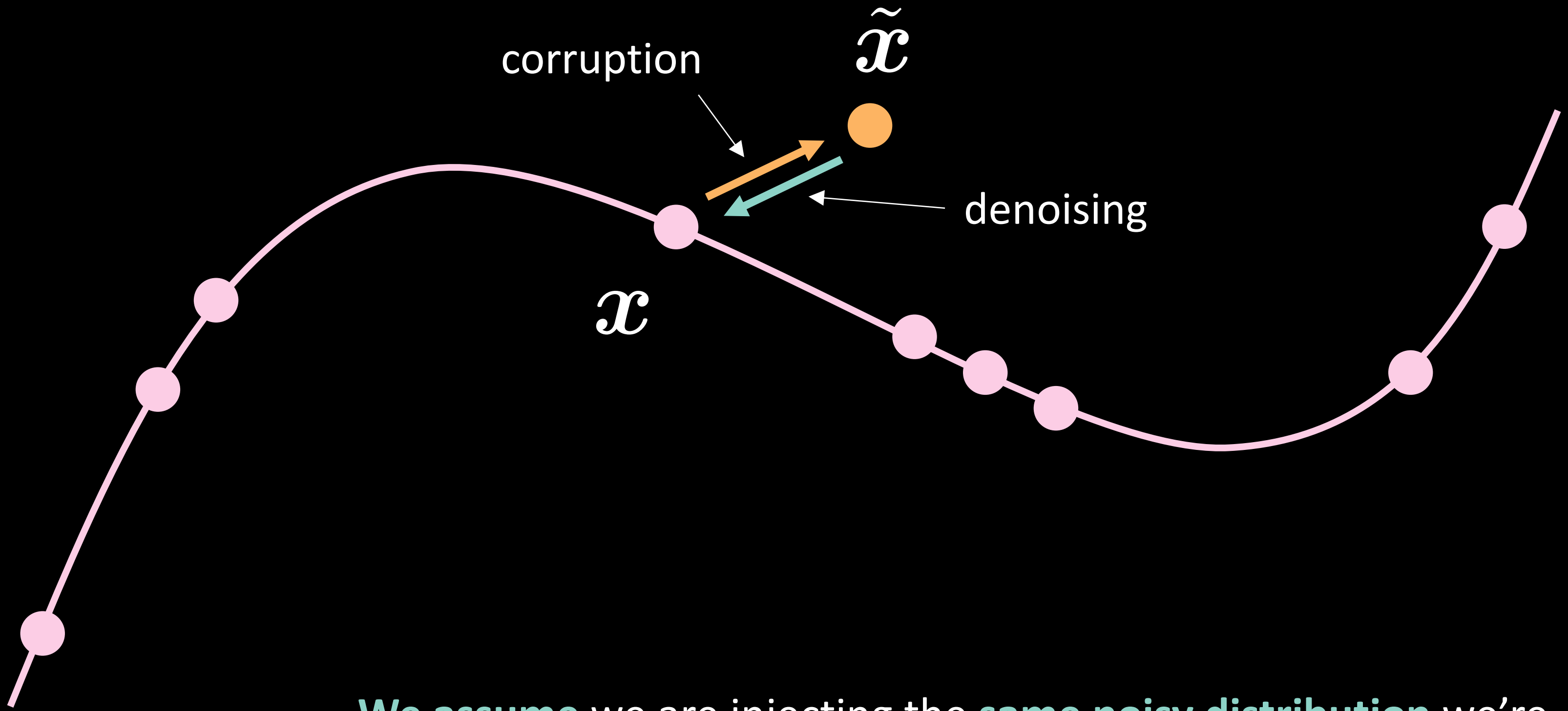
$$\ell(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{2}\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2$$

Under-/over-complete hidden layer



Denoising autoencoder

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



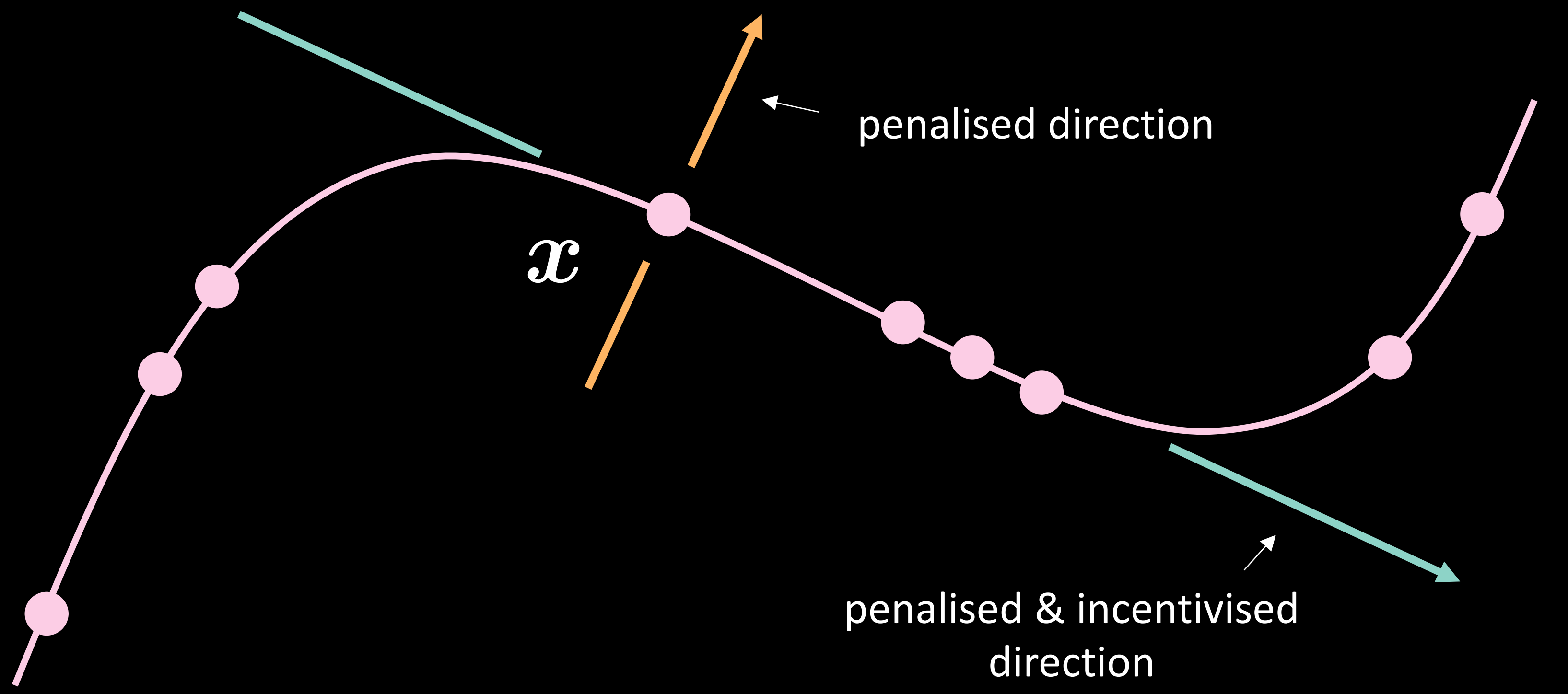
We assume we are injecting the **same noisy distribution** we're going to observe in reality. In this way, we can learn how to robustly recover from it.

Contractive autoencoder

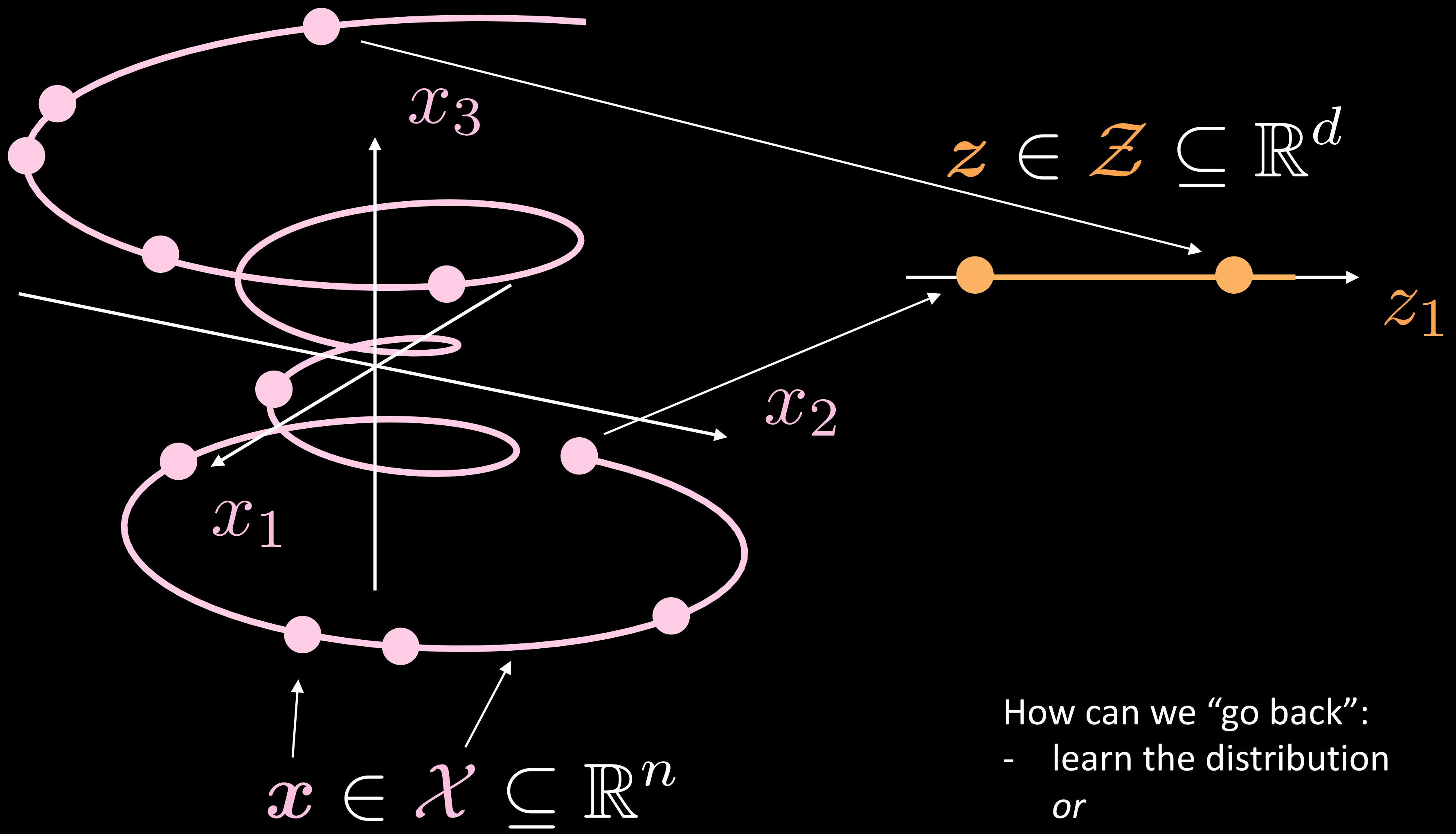
penalises <u>insensitivity</u> to **reconstruction directions**

penalises <u>sensitivity</u> to the **any direction**

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



Basic auto-encoder



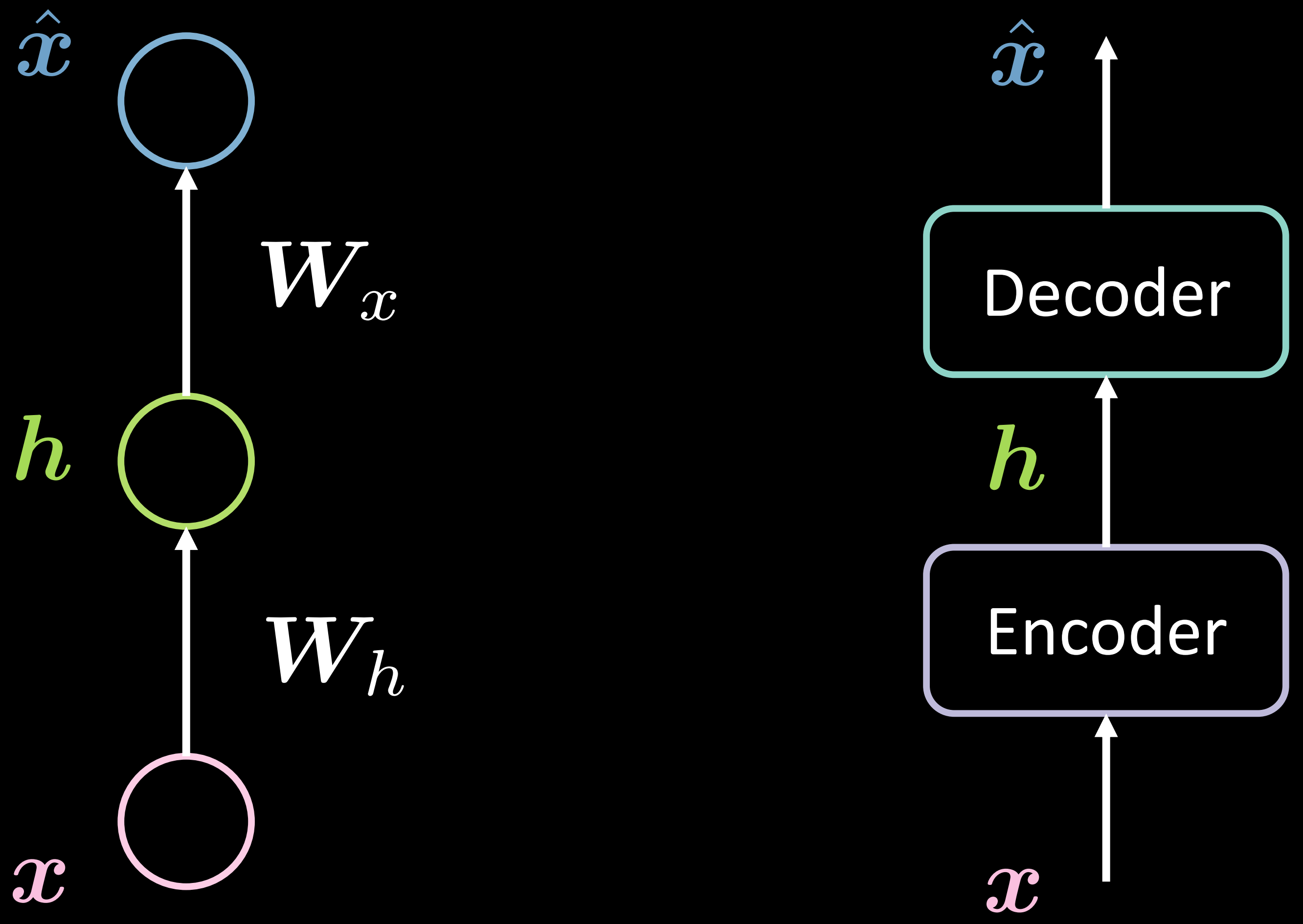
How can we "go back":

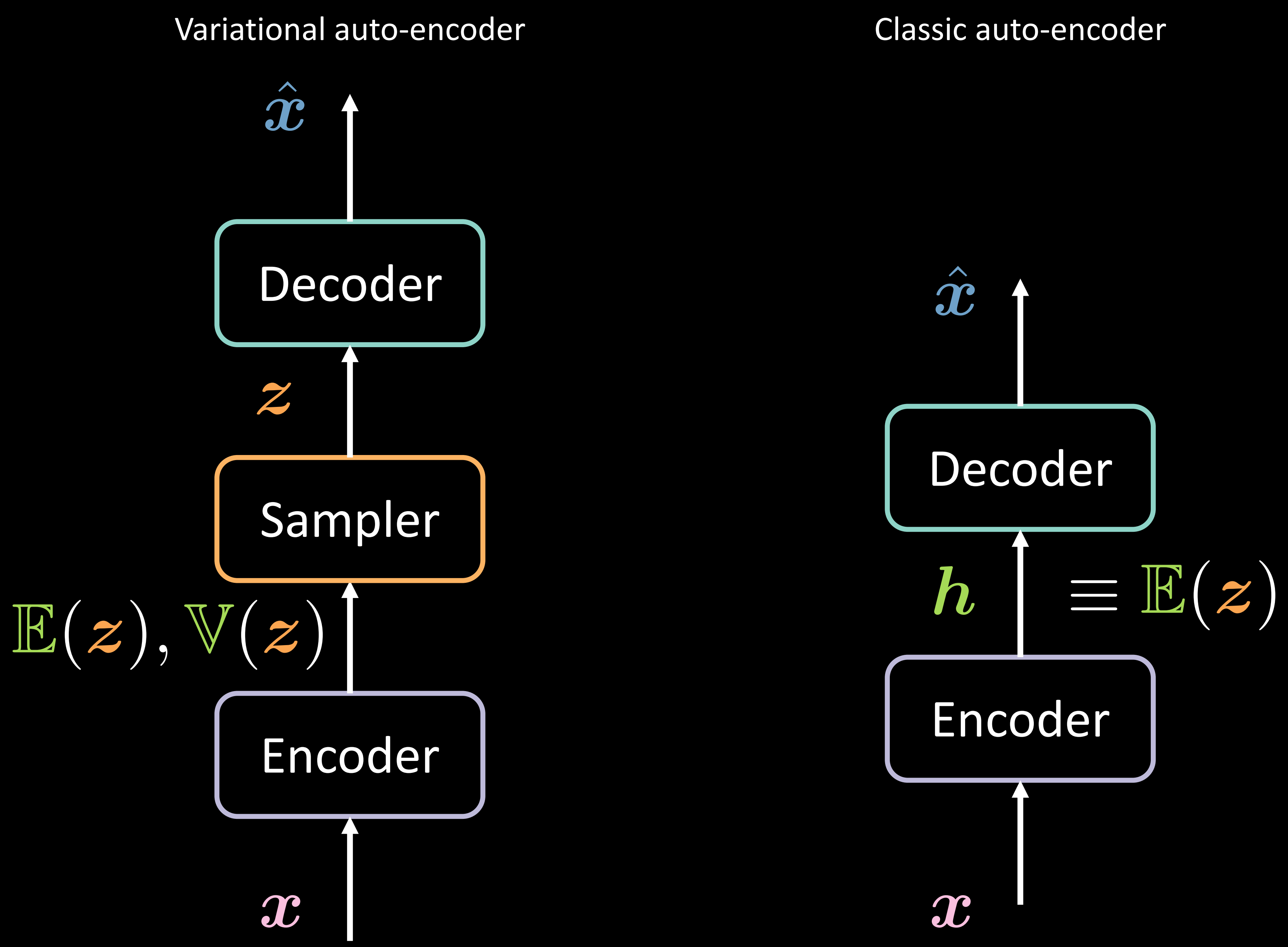
- learn the distribution

  or

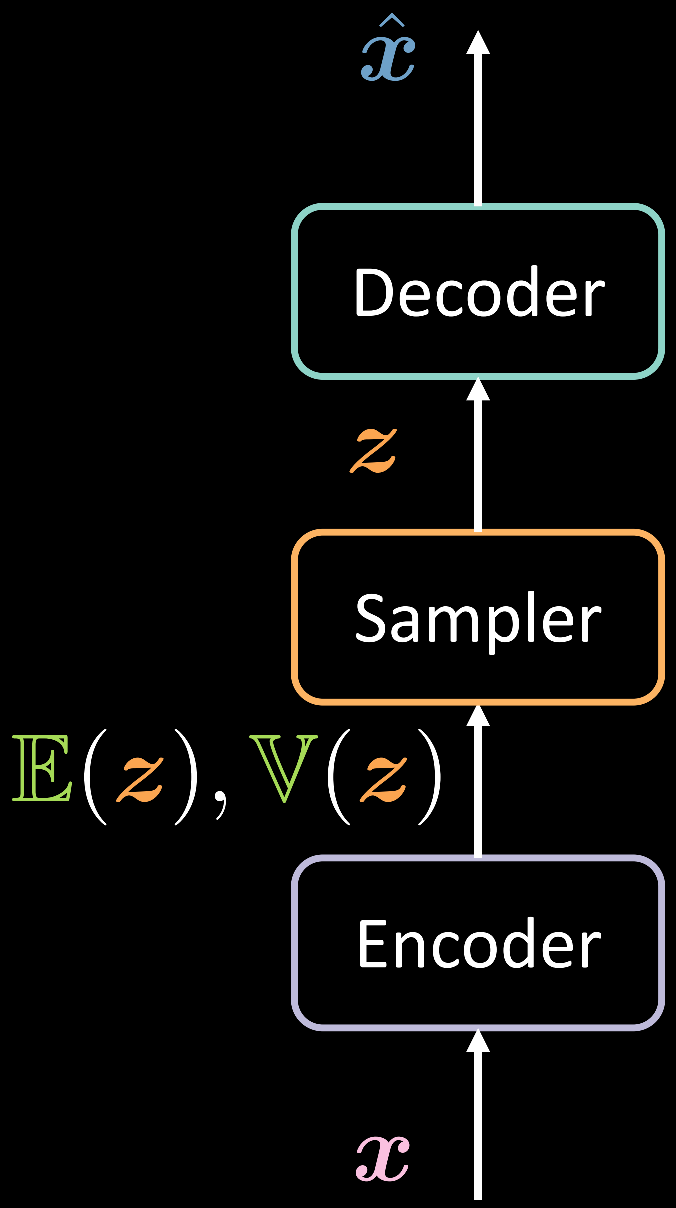
- enforce some structure

Auto-encoder (recap)




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

Variational auto-encoder



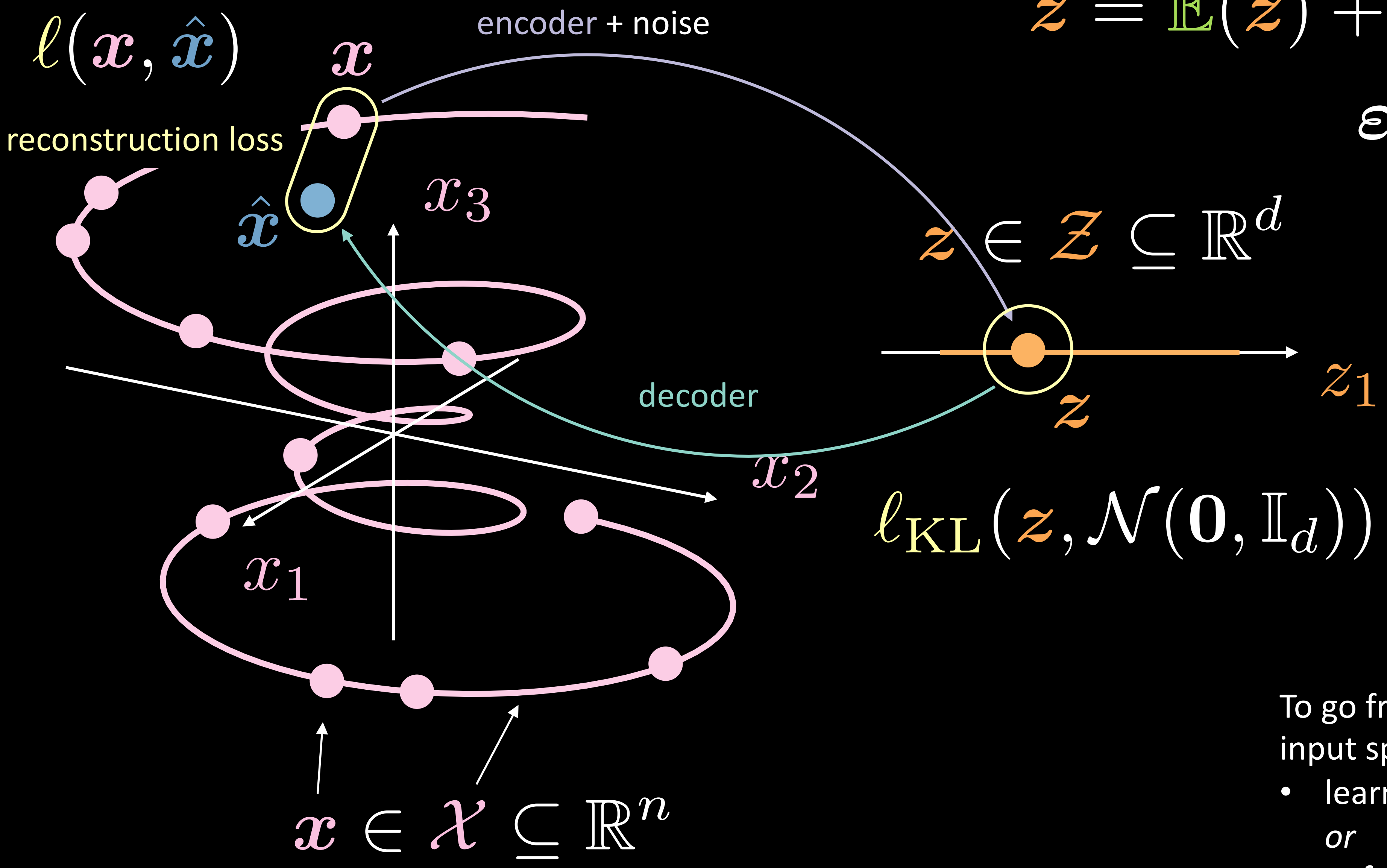
$$\text{decoder} : \mathcal{Z} \to \mathbb{R}^n$$
$$z \mapsto \hat{x}$$

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

Variational auto-encoder

$$z = \mathbb{E}(z) + \varepsilon \odot \sqrt{\mathbb{V}(z)}$$

$$\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbb{I}_d)$$

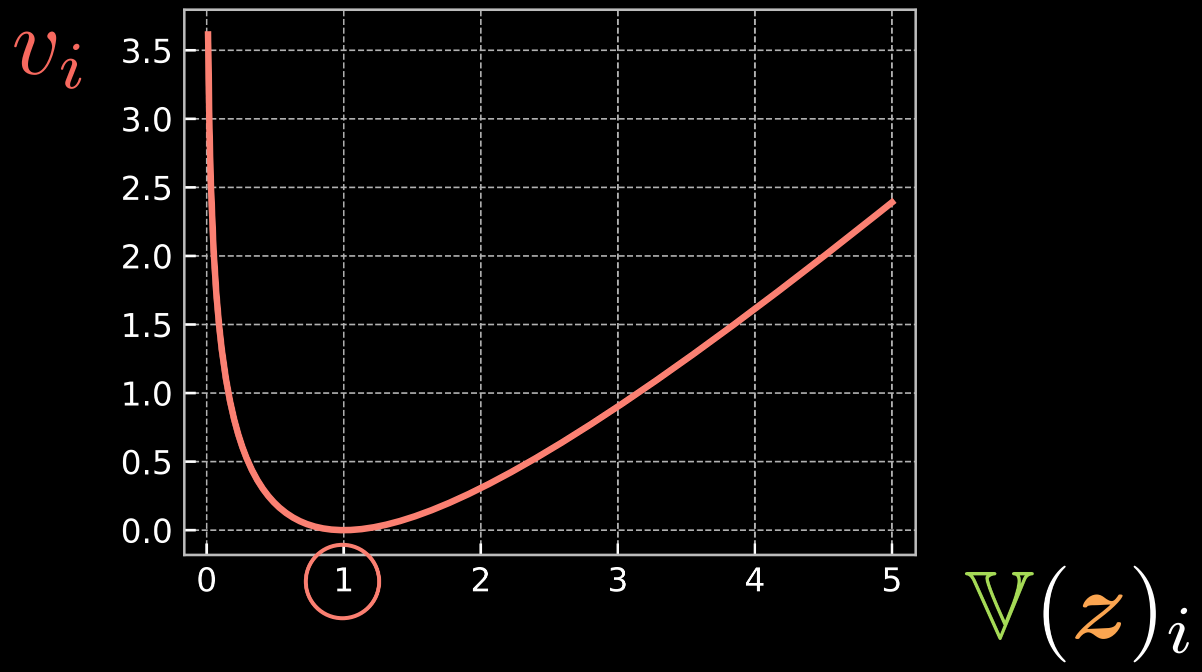


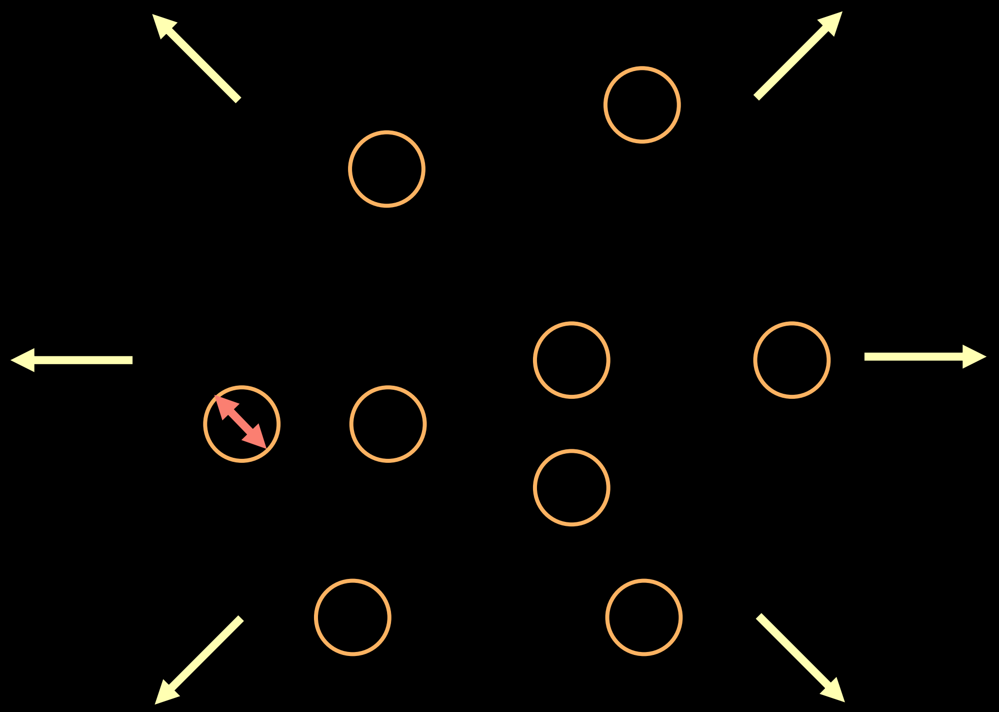
To go from the latent to the input space we need to:

- learn the distribution
  or
- enforce some structure

Variational auto-encoder

$$\begin{aligned}\ell(\boldsymbol{x},\hat{\boldsymbol{x}}) &= \ell_{\text{reconstruction}} + \beta\ell_{\text{KL}}(\boldsymbol{z},\mathcal{N}(\boldsymbol{0},\mathbb{I}_d)) \\ &= \ell_{\text{reconstruction}} + \\ &+ \beta\textstyle\sum_{i=1}^{d}\Big(\underbrace{\mathbb{V}(\boldsymbol{z}) - \log\left[\mathbb{V}(\boldsymbol{z})\right] - \boldsymbol{1}}_{\boldsymbol{v}} + \mathbb{E}(\boldsymbol{z})^2\Big)_i\end{aligned}$$

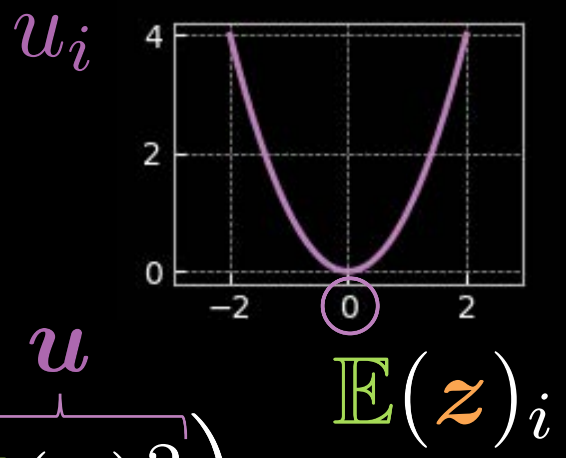


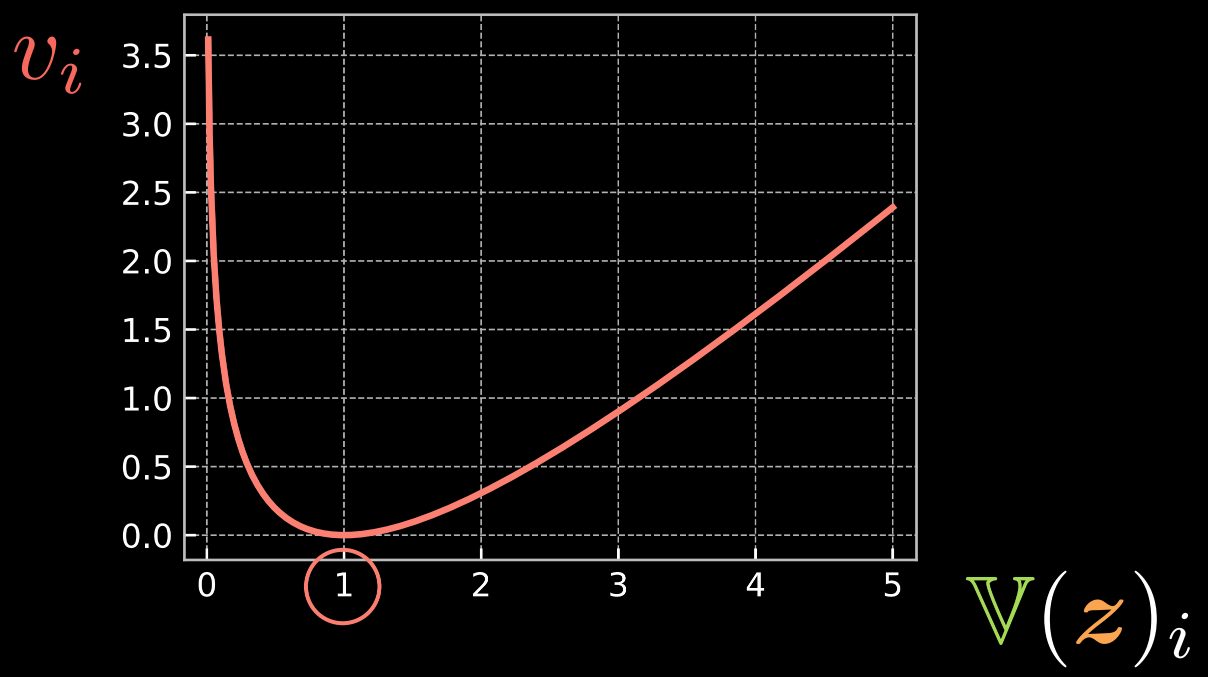


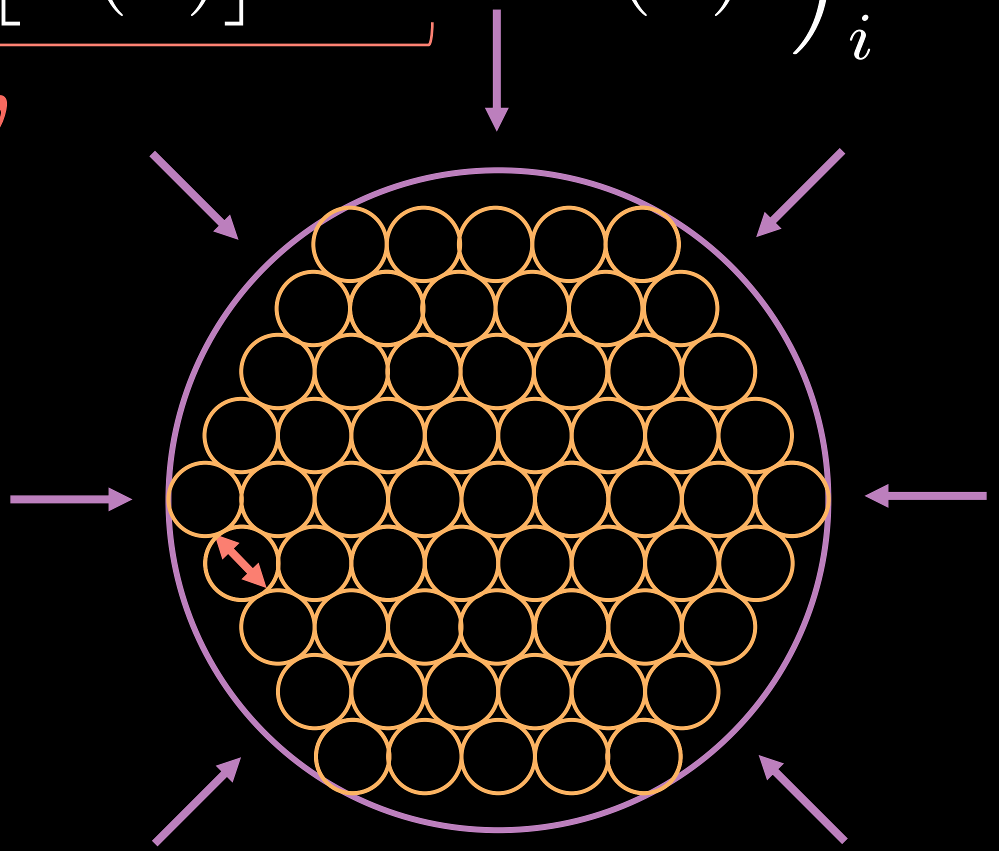
Variational auto-encoder

$$\ell(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \ell_{\text{reconstruction}} + \beta \ell_{\text{KL}}(\boldsymbol{z}, \mathcal{N}(\mathbf{0}, \mathbb{I}_d))$$

$$= \ell_{\text{reconstruction}} + \beta \sum_{i=1}^{d} \Big(\underbrace{\mathbb{V}(\boldsymbol{z}) - \log\left[\mathbb{V}(\boldsymbol{z})\right] - \mathbf{1}}_{\boldsymbol{v}} + \overbrace{\mathbb{E}(\boldsymbol{z})^2}^{\boldsymbol{u}} \Big)_i$$

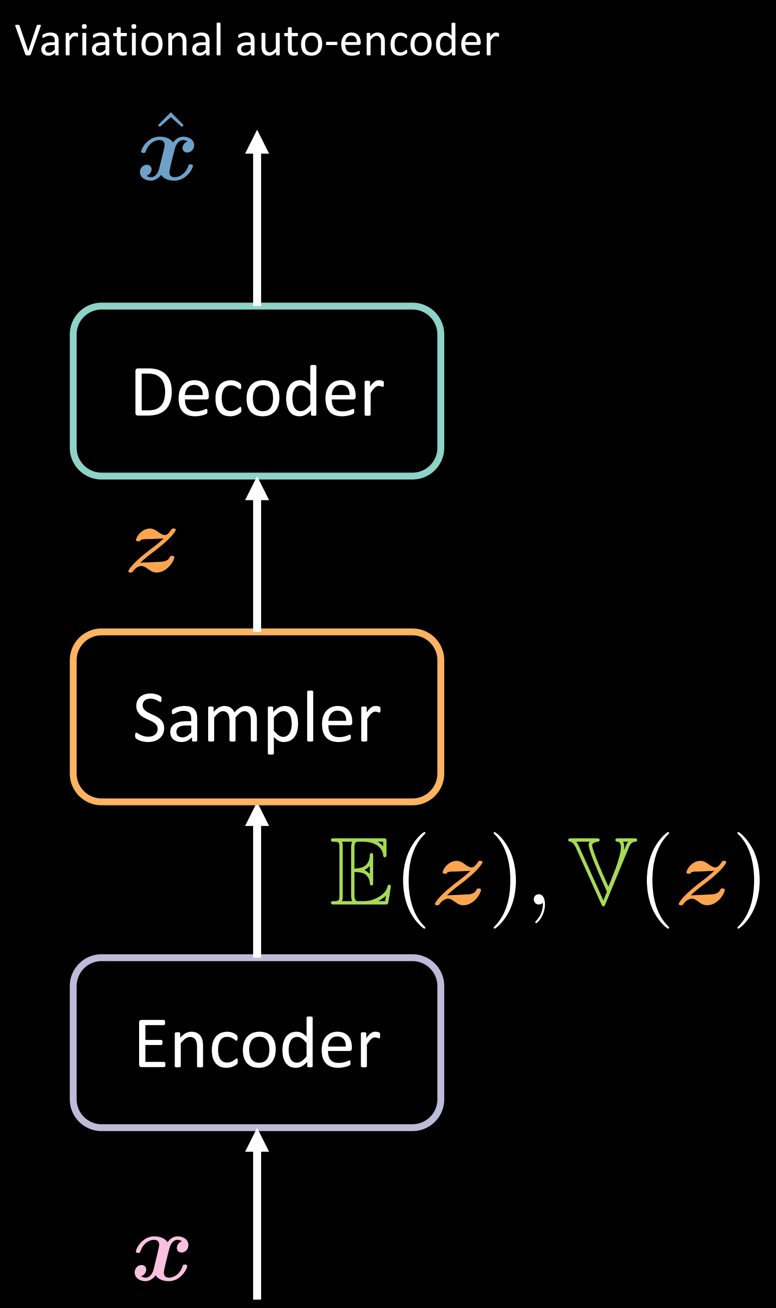


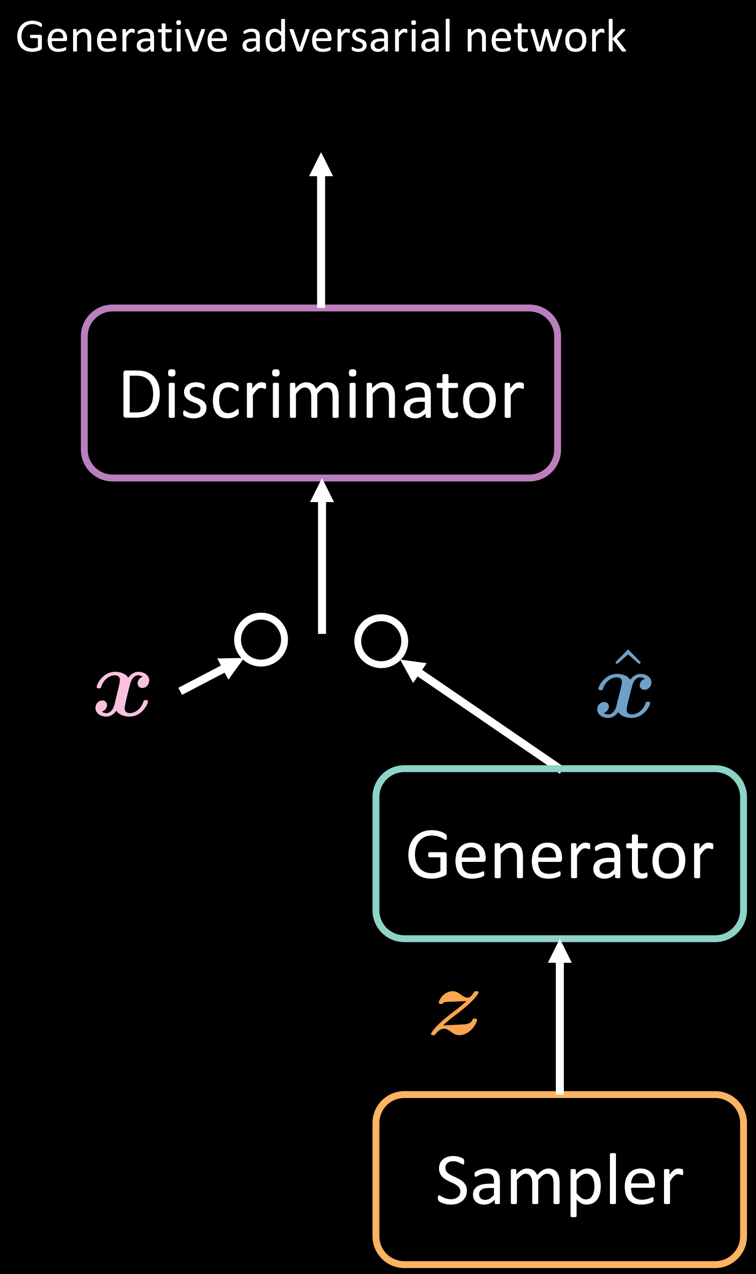




Generative adversarial nets

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

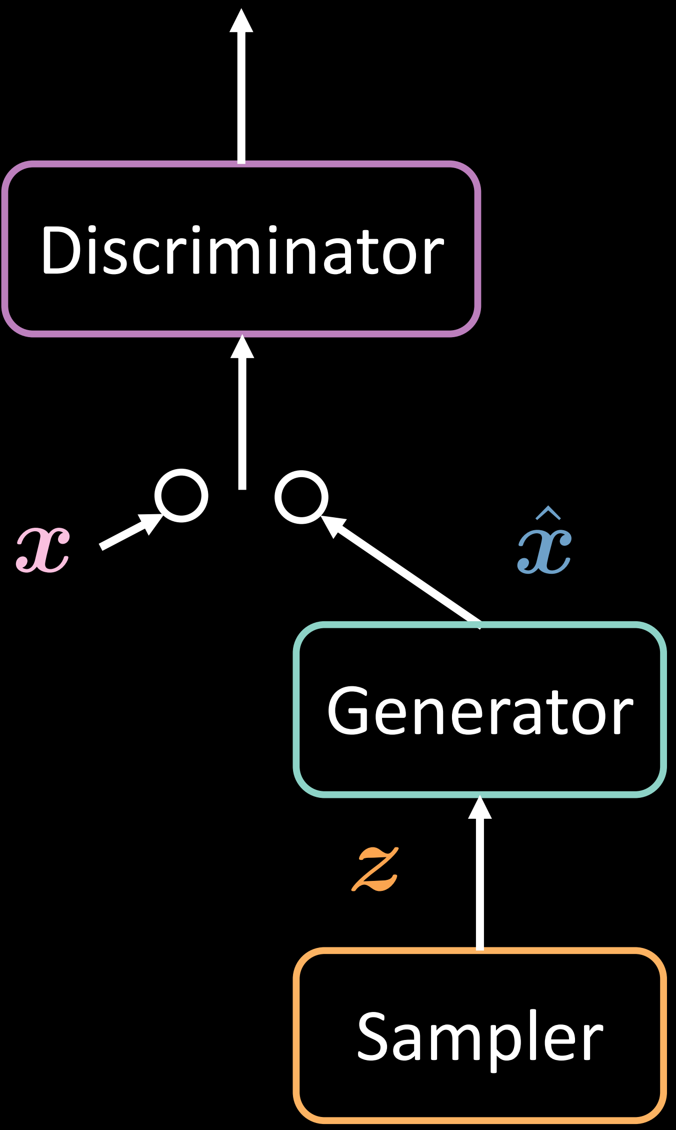
Generative adversarial network

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

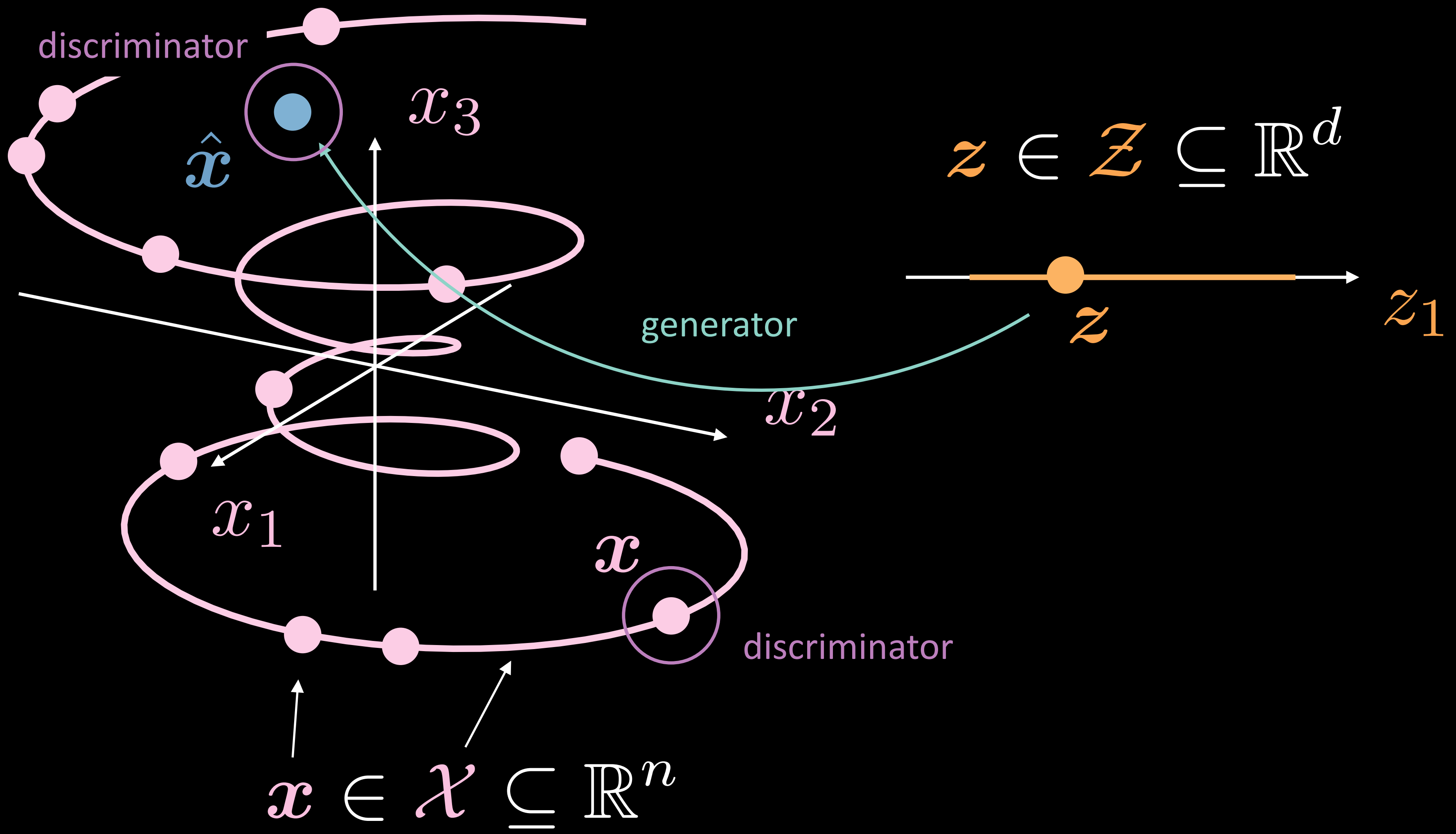
$$\boldsymbol{x} \vee \hat{\boldsymbol{x}} \mapsto \ell$$

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

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



Generative adversarial network



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