Foundations of Deep Learning

ALF

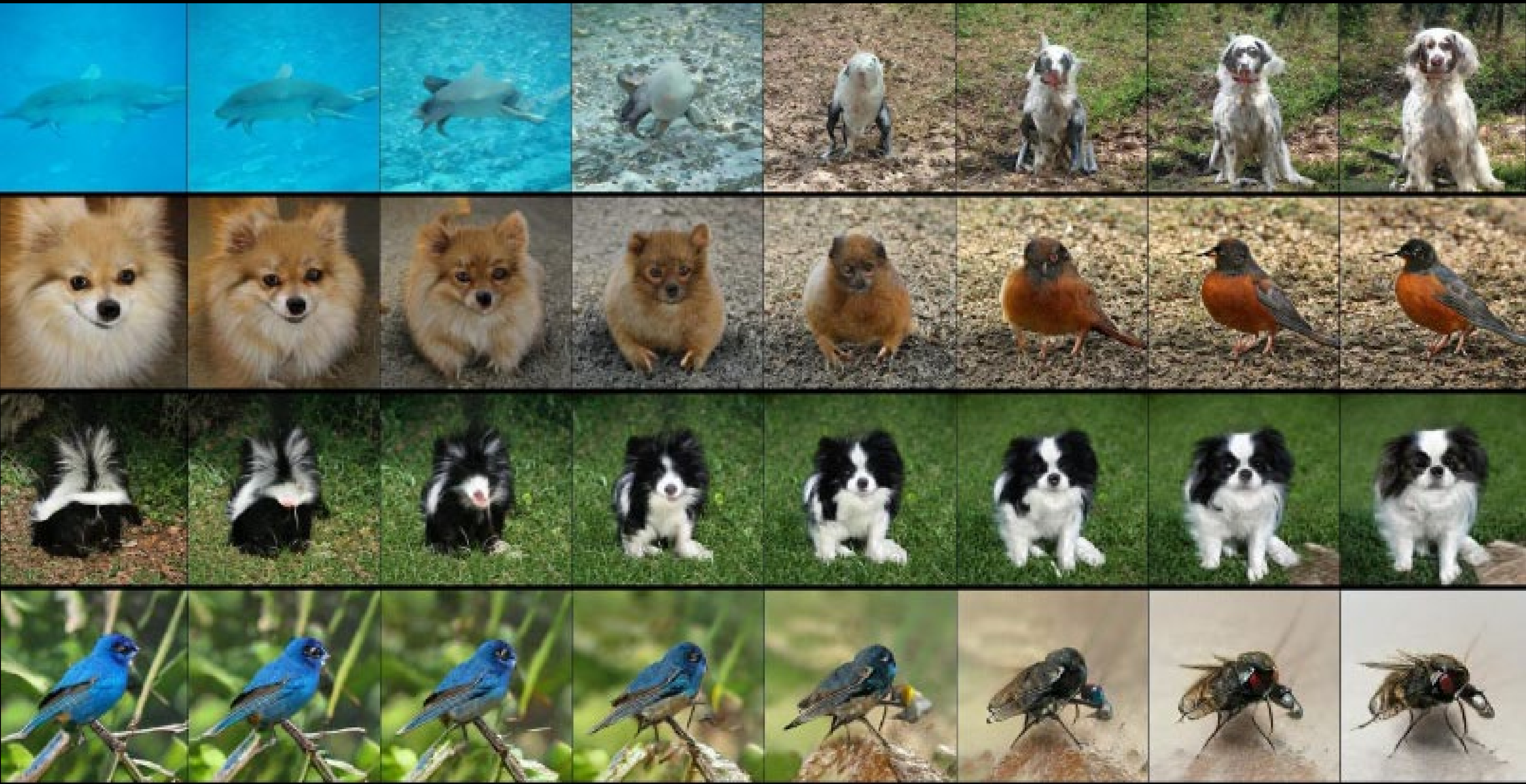
Alfredo Canziani

@alfcnz

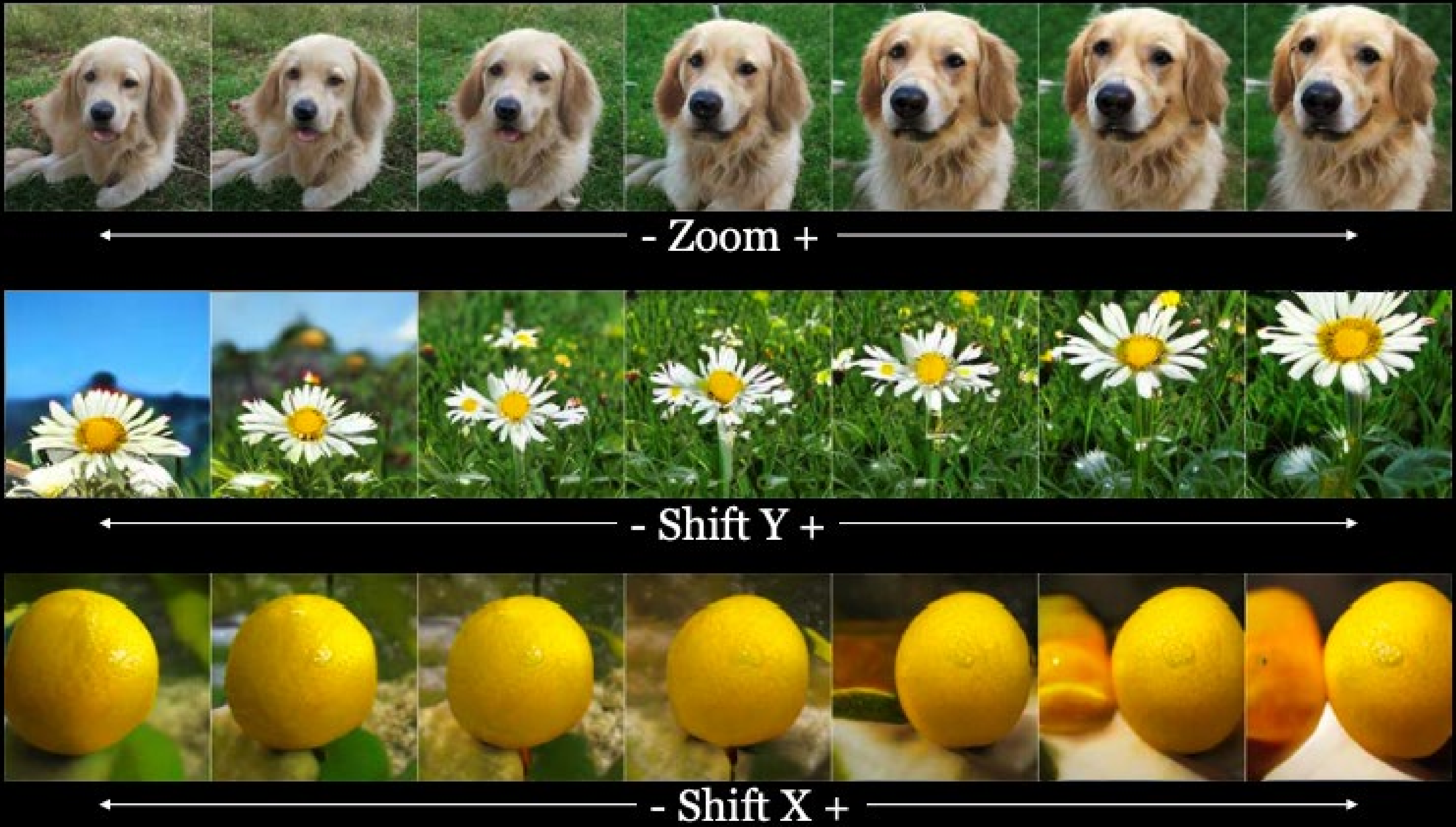
(Self-) Un-supervised learning

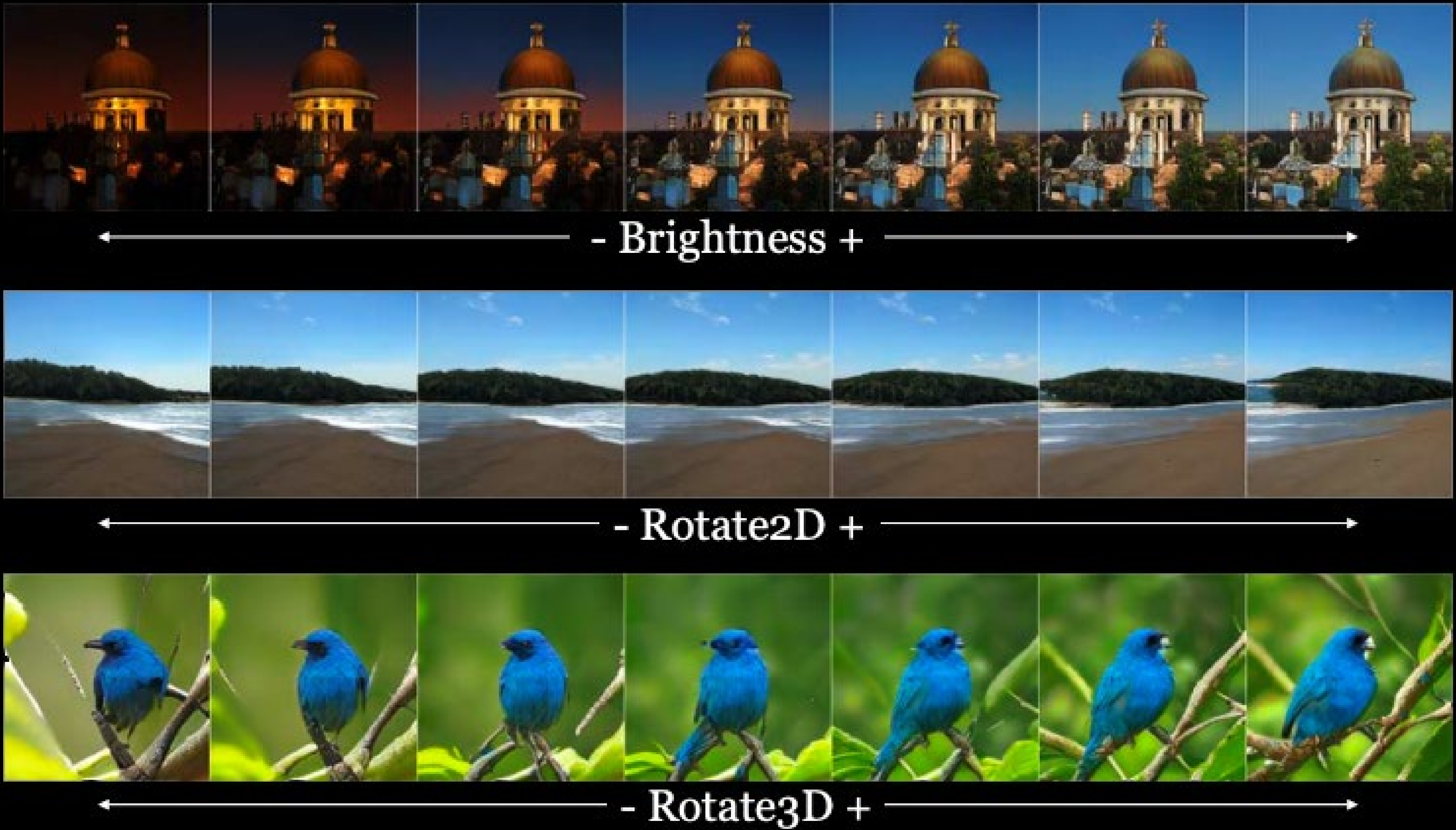
Generative models capabilities

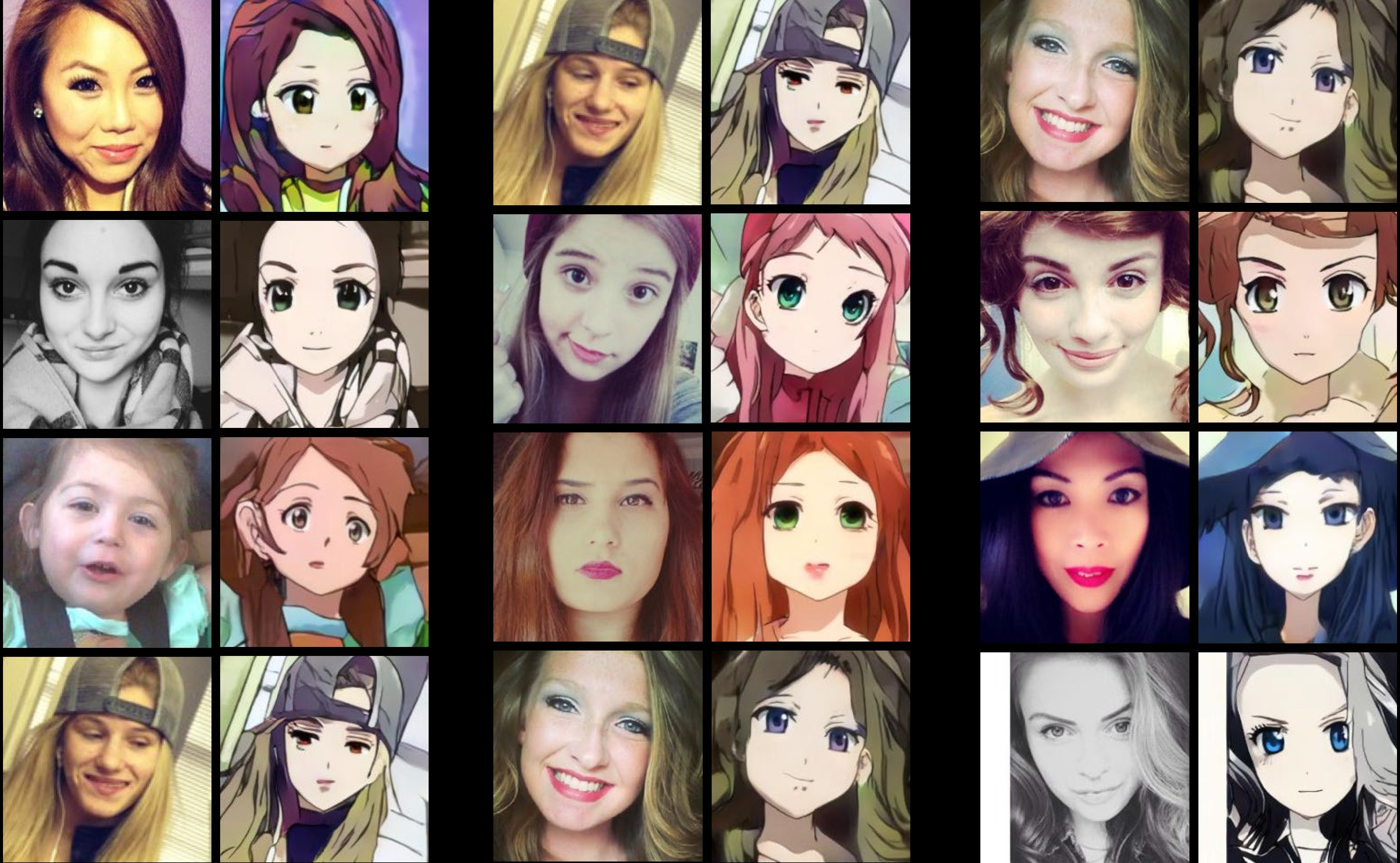
Karras (2019) StyleGAN2 – Analysing and Improving the Image Quality of StyleGAN

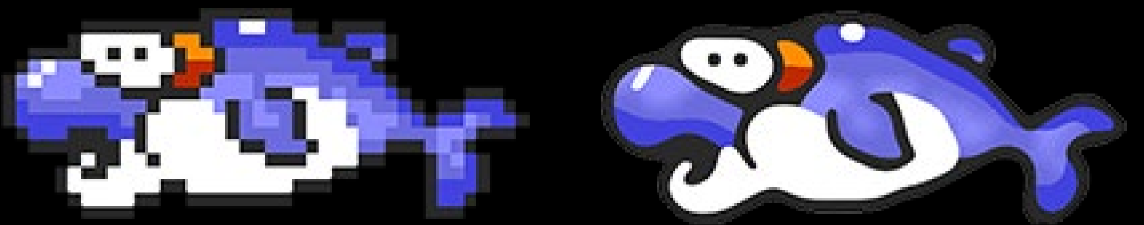
Brock (2018) Large Scale GAN Training for High Fidelity Natural Image Synthesis

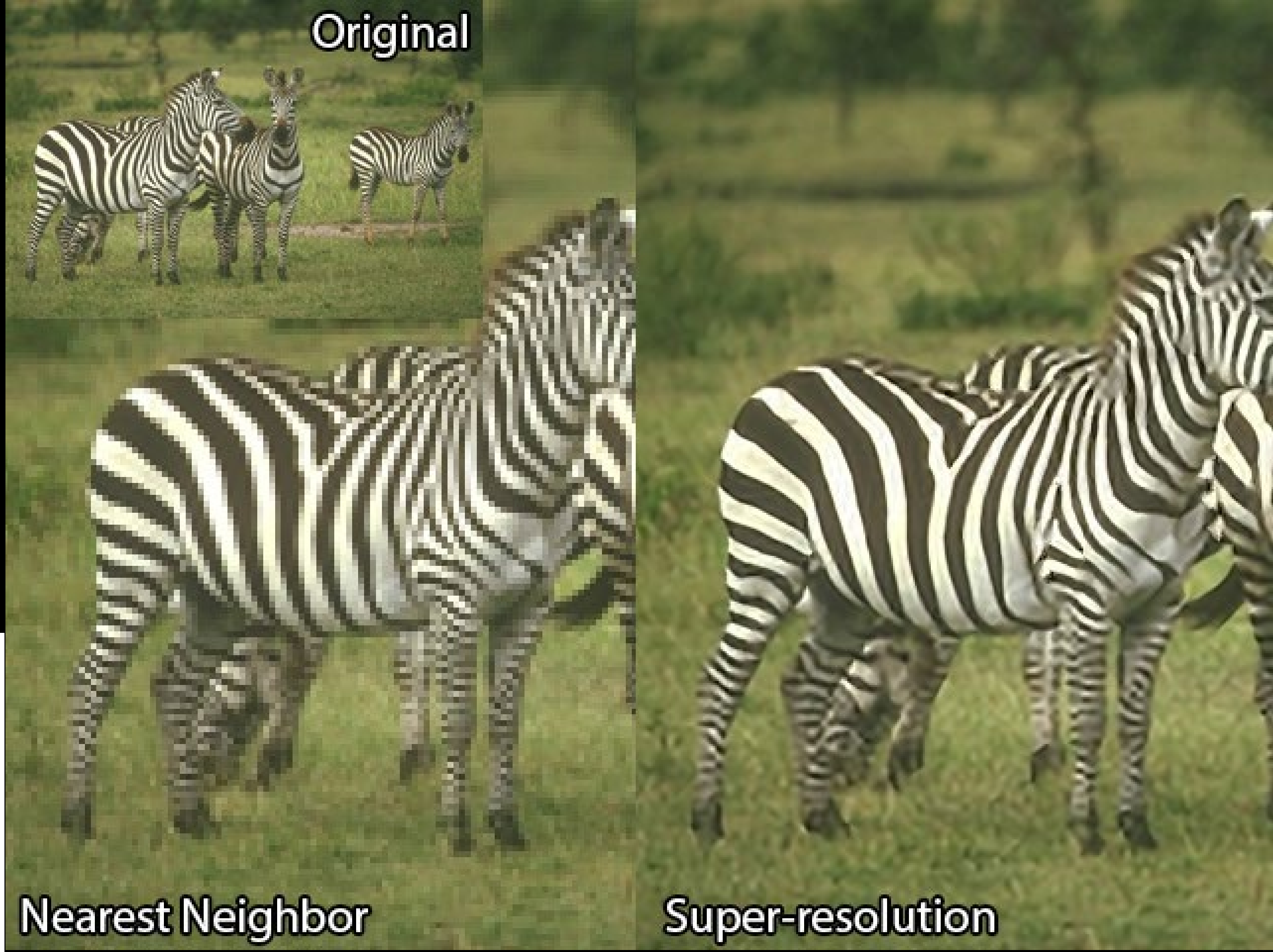

- Zoom +
- Shift Y +
- Shift X +


- Brightness +
- Rotate2D +
- Rotate3D +



Super resolution



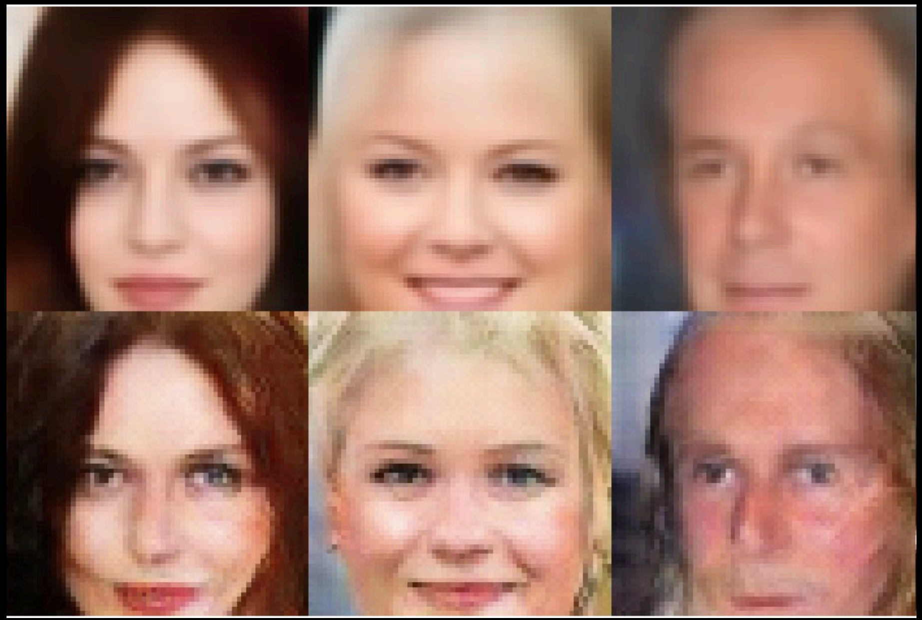


Glasner (2009) Super-resolution from a single image
www.wisdom.weizmann.ac.il/~vision/SingleImageSR.html

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

Inpainting

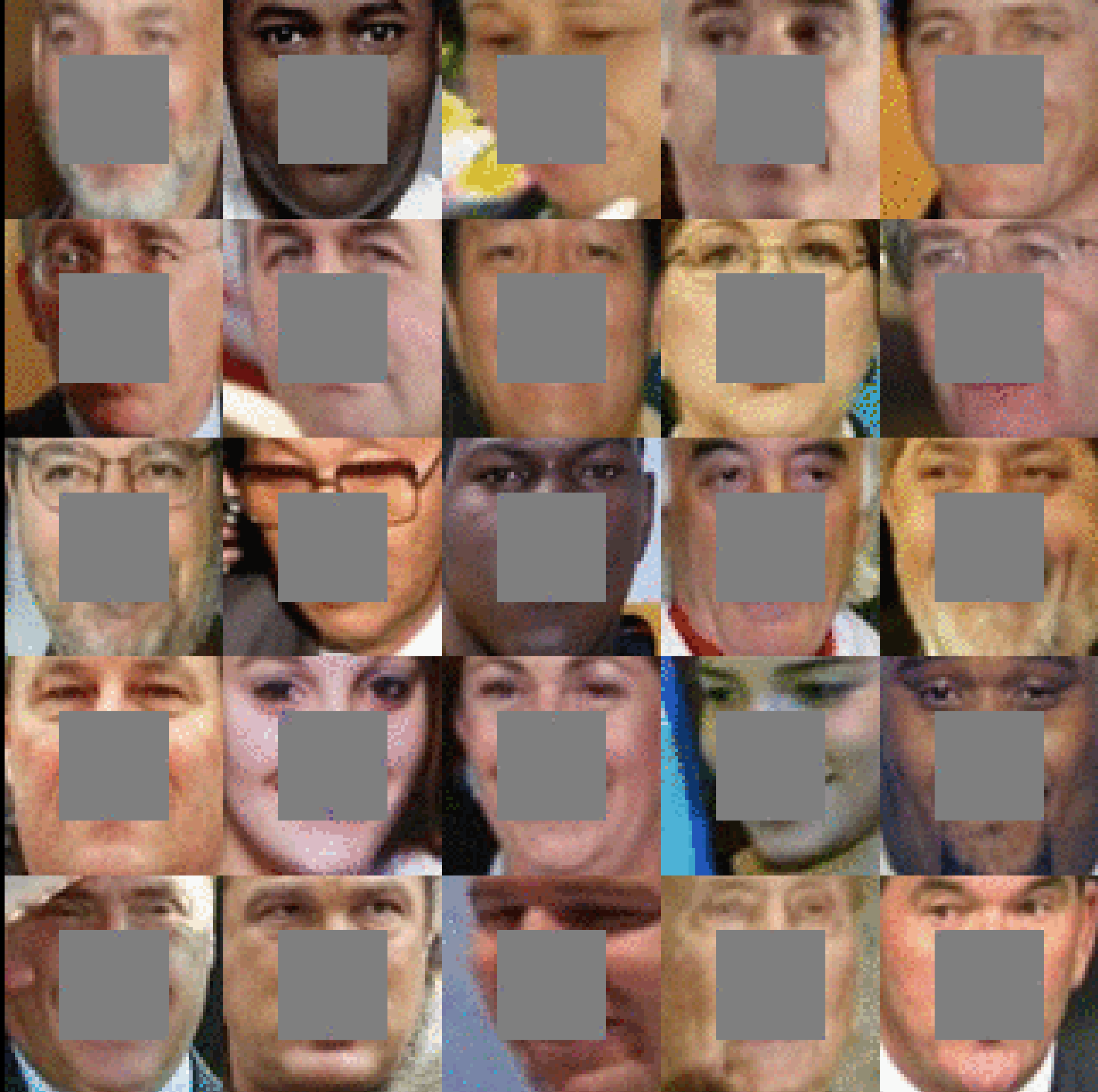
VAE



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

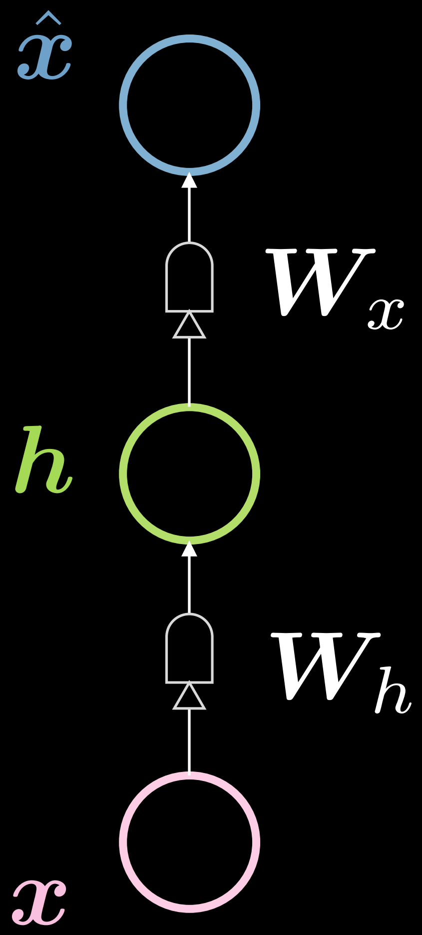
Reed (2016) Generative adversarial text to image synthesis

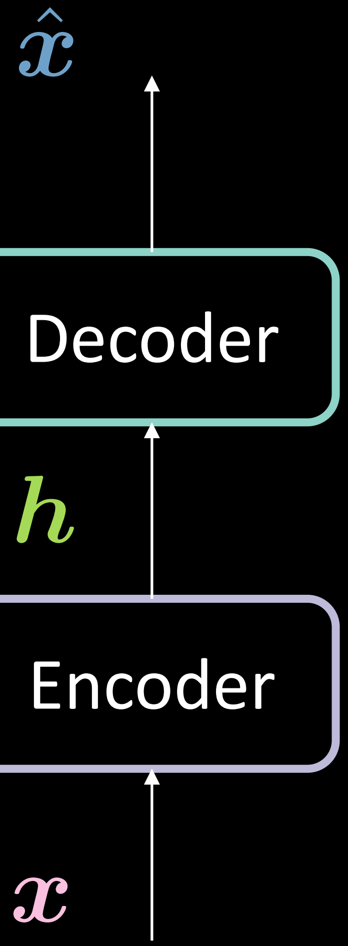
github.com/reedscot/icml2016

Autoencoders

Unsupervised learning

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

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

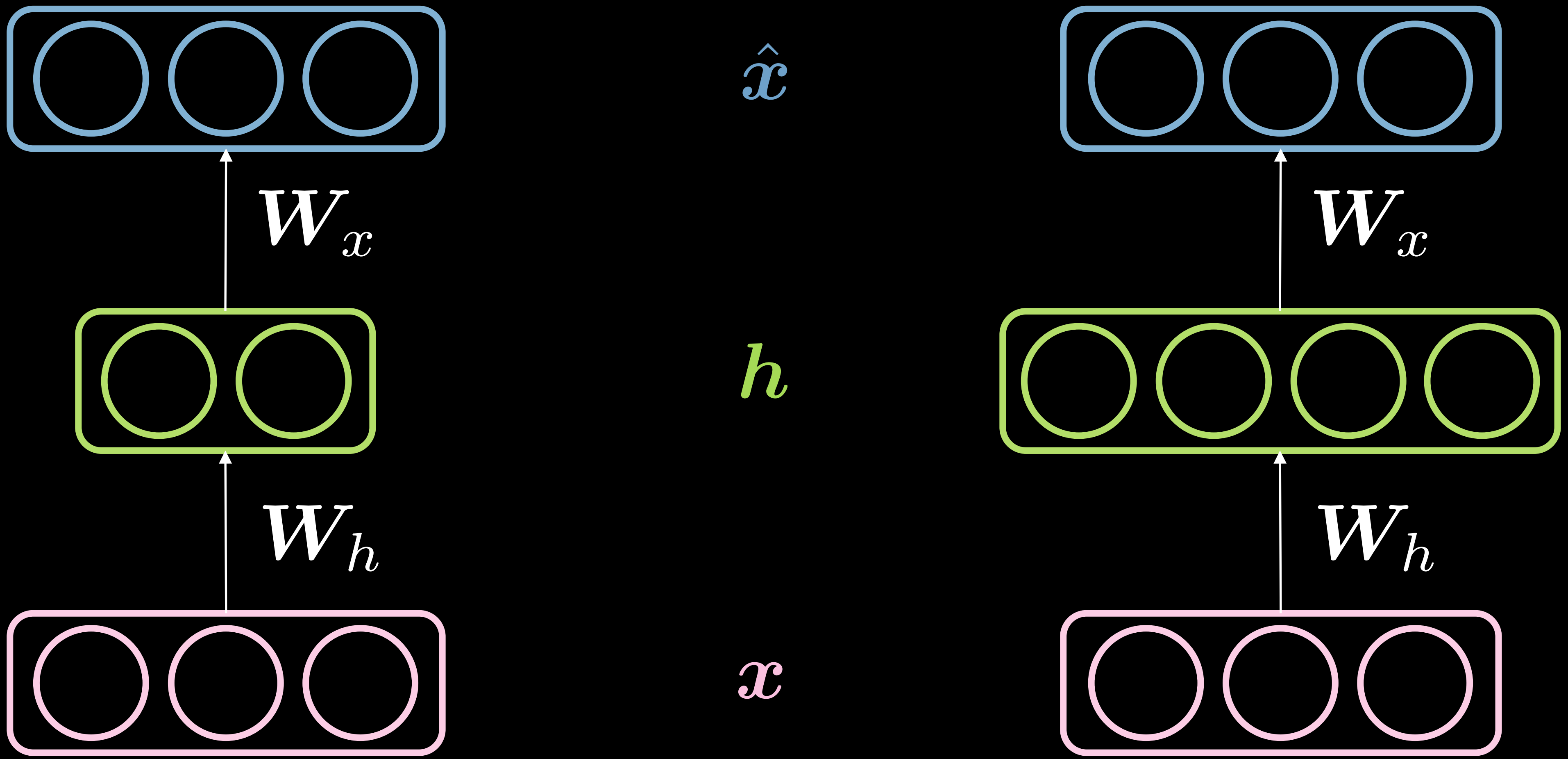
binary input

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

real valued input

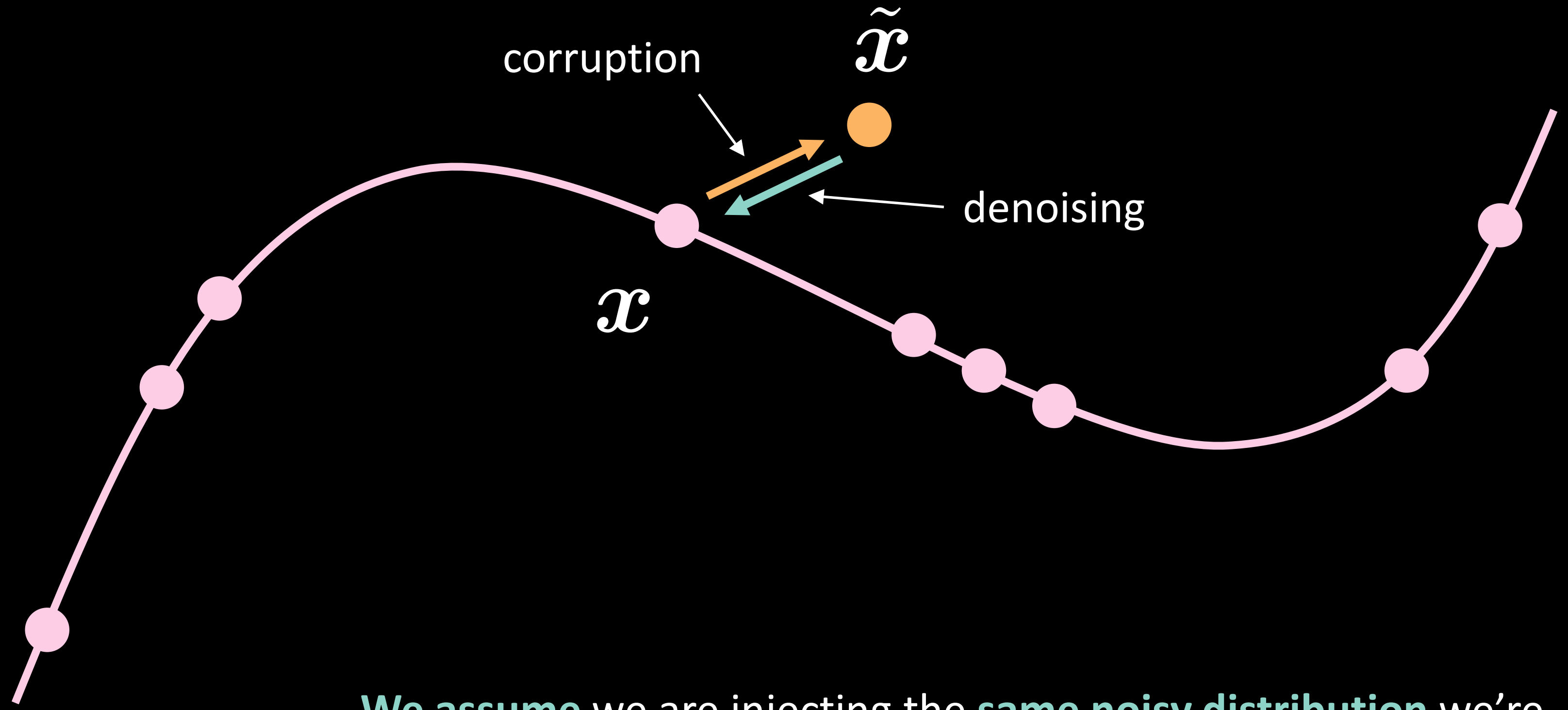
$$\ell(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{2}\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2$$

Under-/over-complete hidden layer

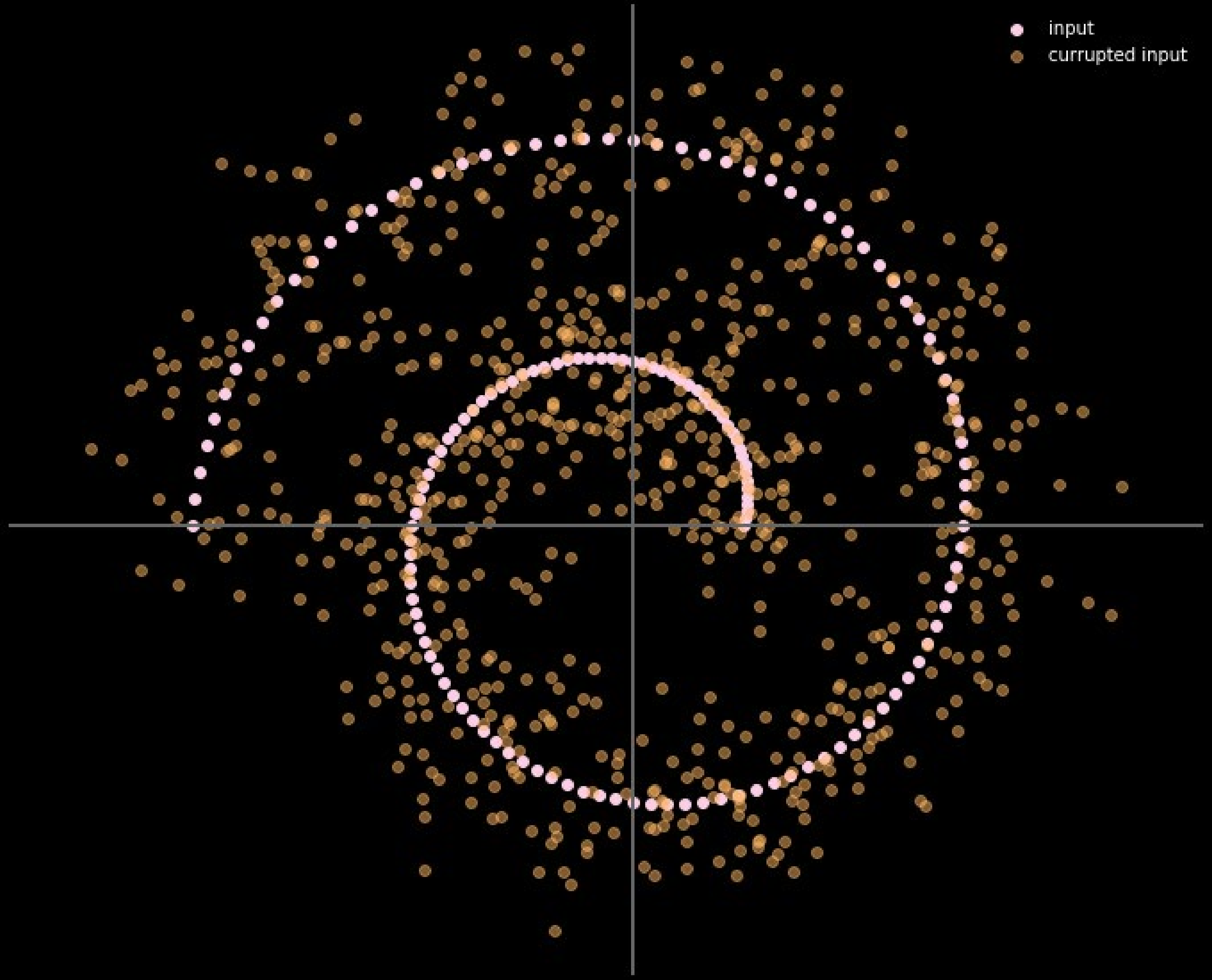


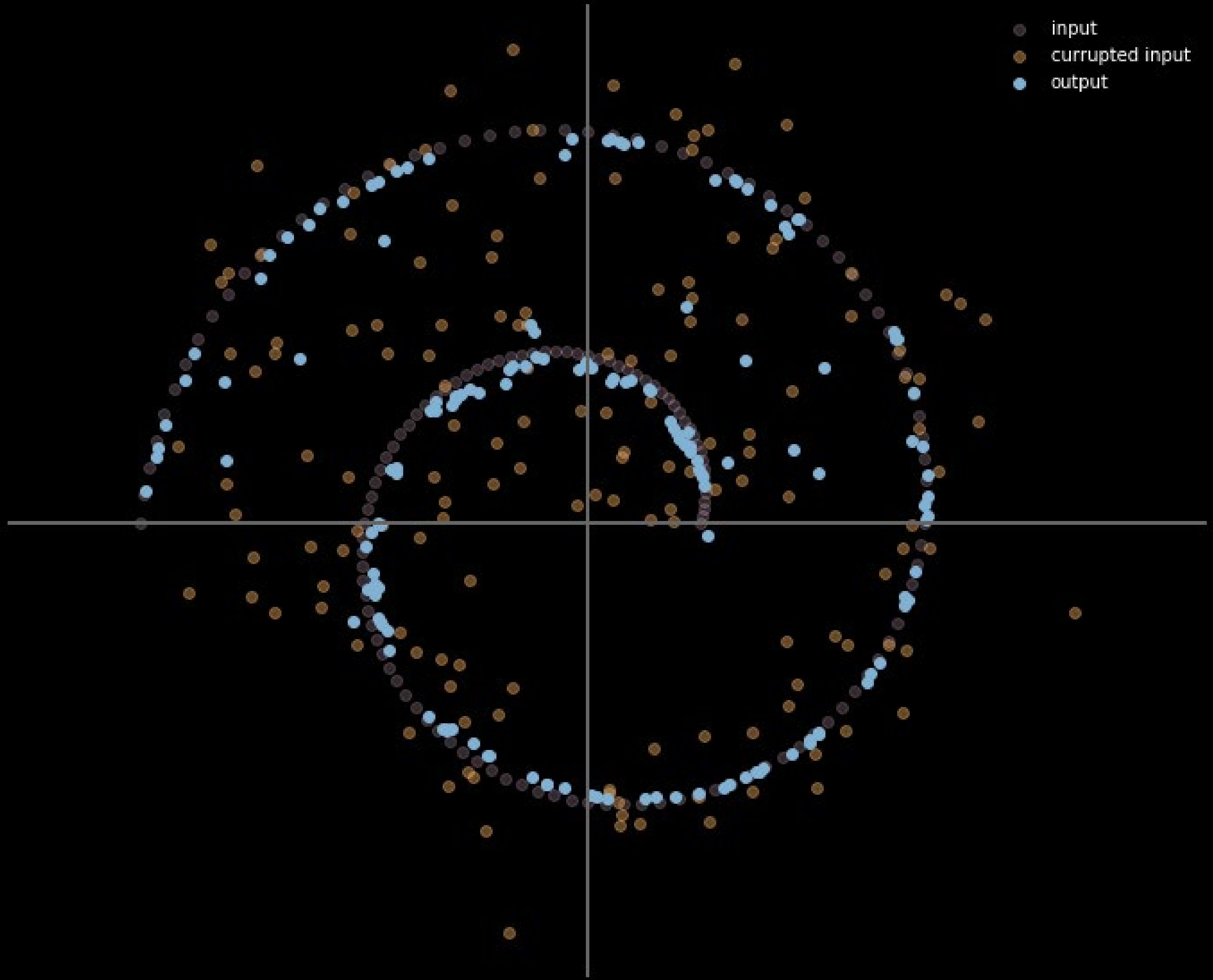
Denoising autoencoder

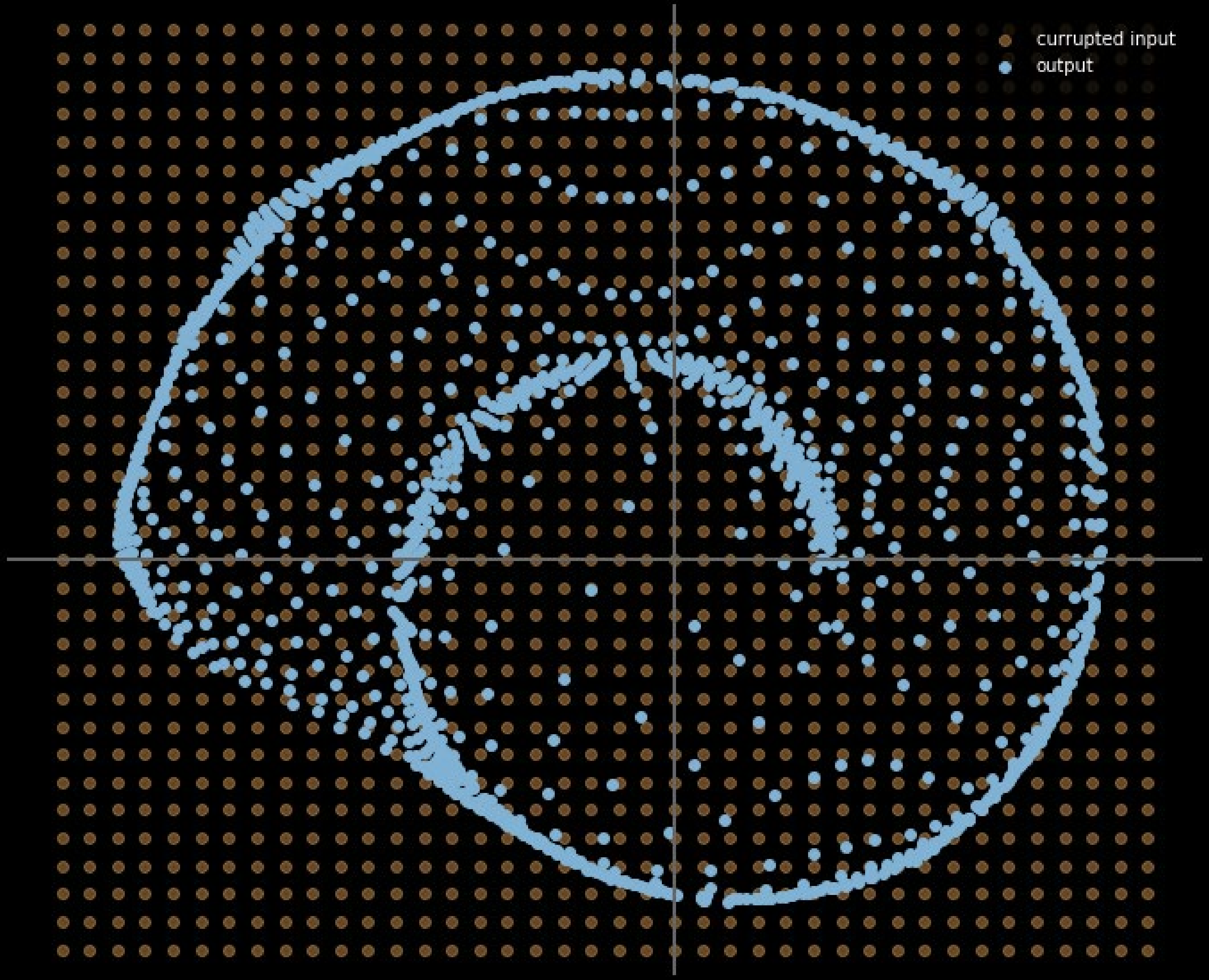
$$\tilde{\boldsymbol{x}} \sim p(\tilde{\boldsymbol{x}} \mid \boldsymbol{x})$$

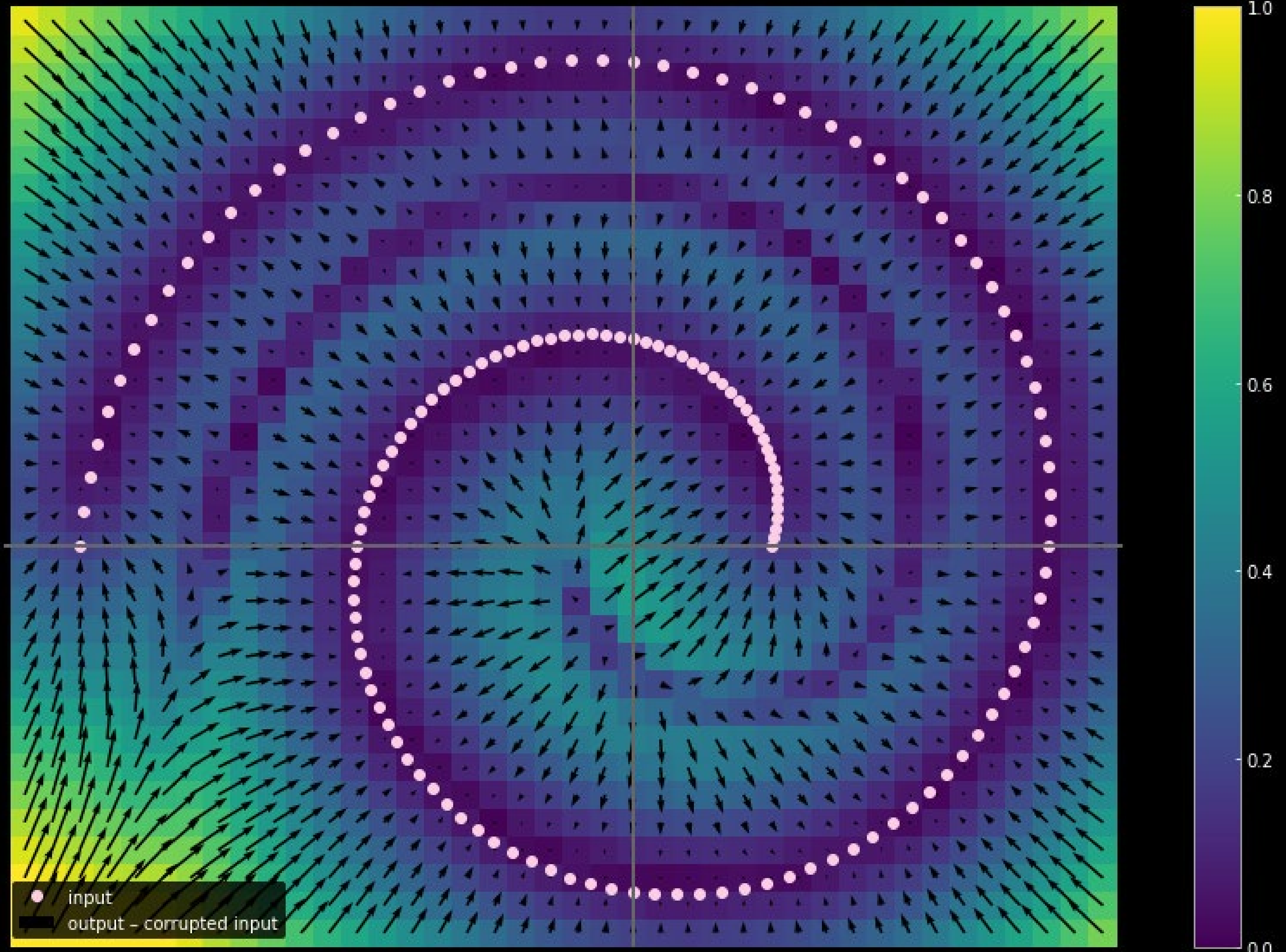


We assume we are injecting the **same noisy distribution** we're going to observe in reality. In this way, we can learn how to robustly recover from it.

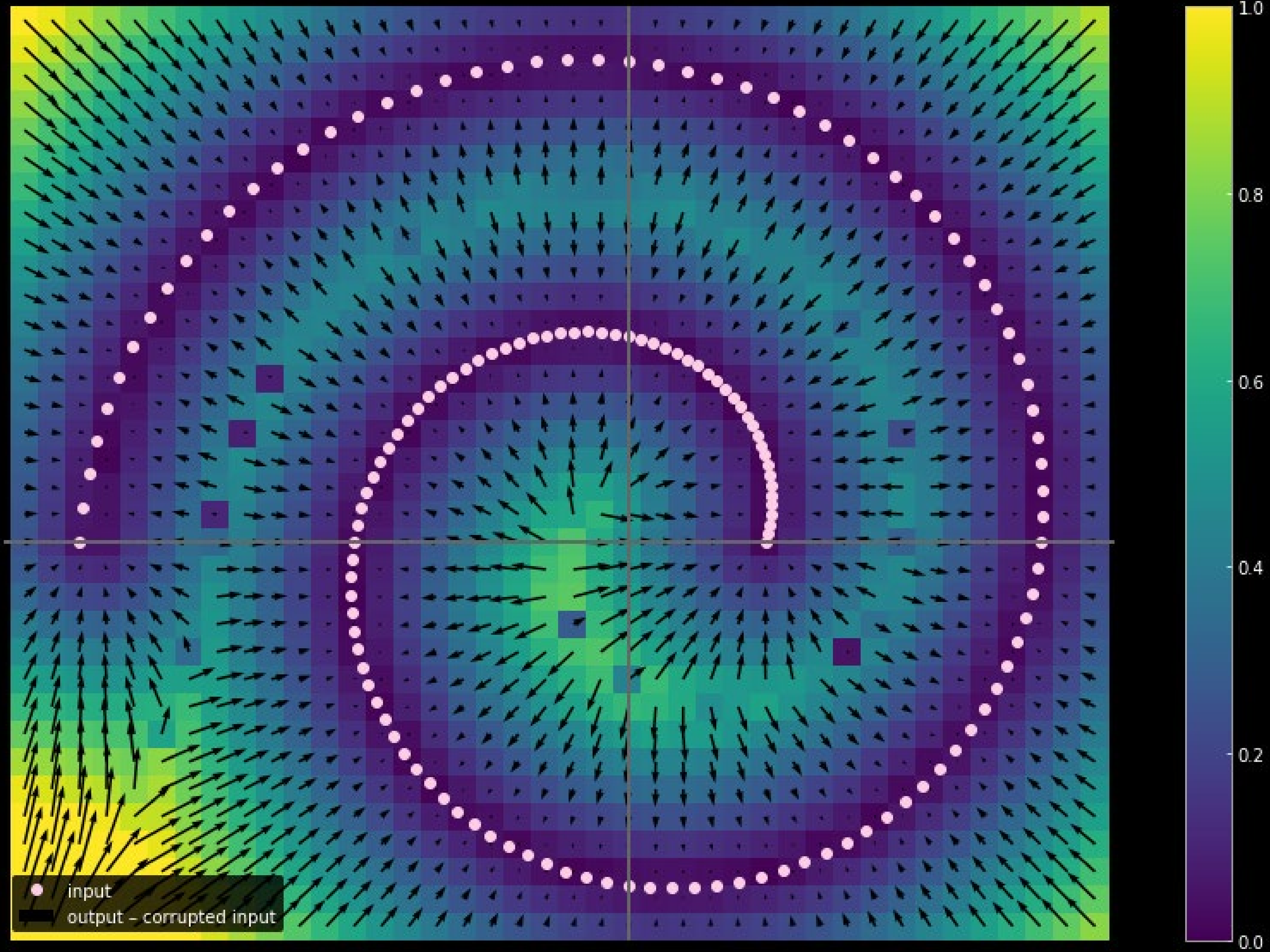

input
currupted input


input
currupted input
output

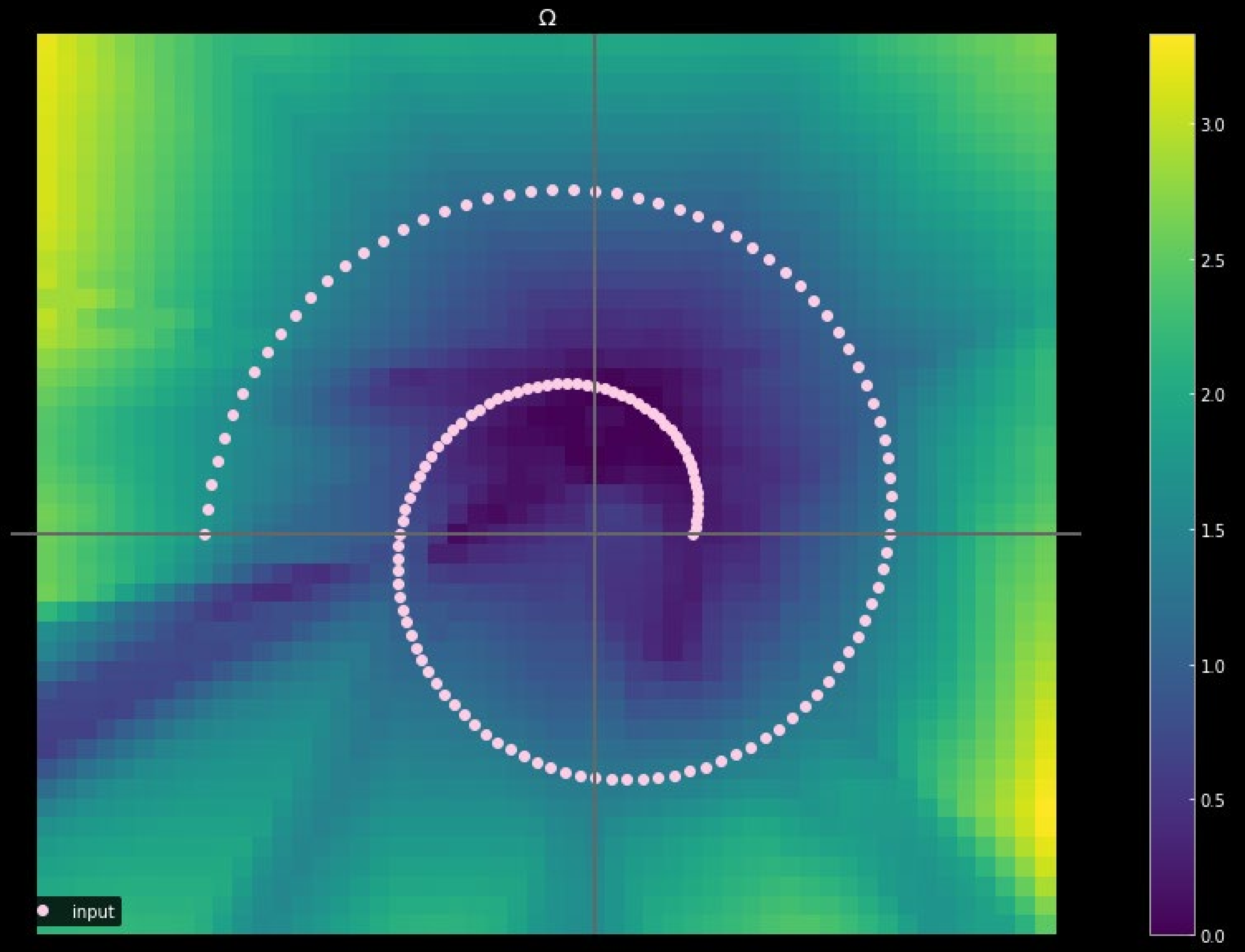

currupted input
output



1.0
0.8
0.6
0.4
0.2
0.0
input
output – corrupted input



input
output – corrupted input
1.0
0.8
0.6
0.4
0.2
0.0

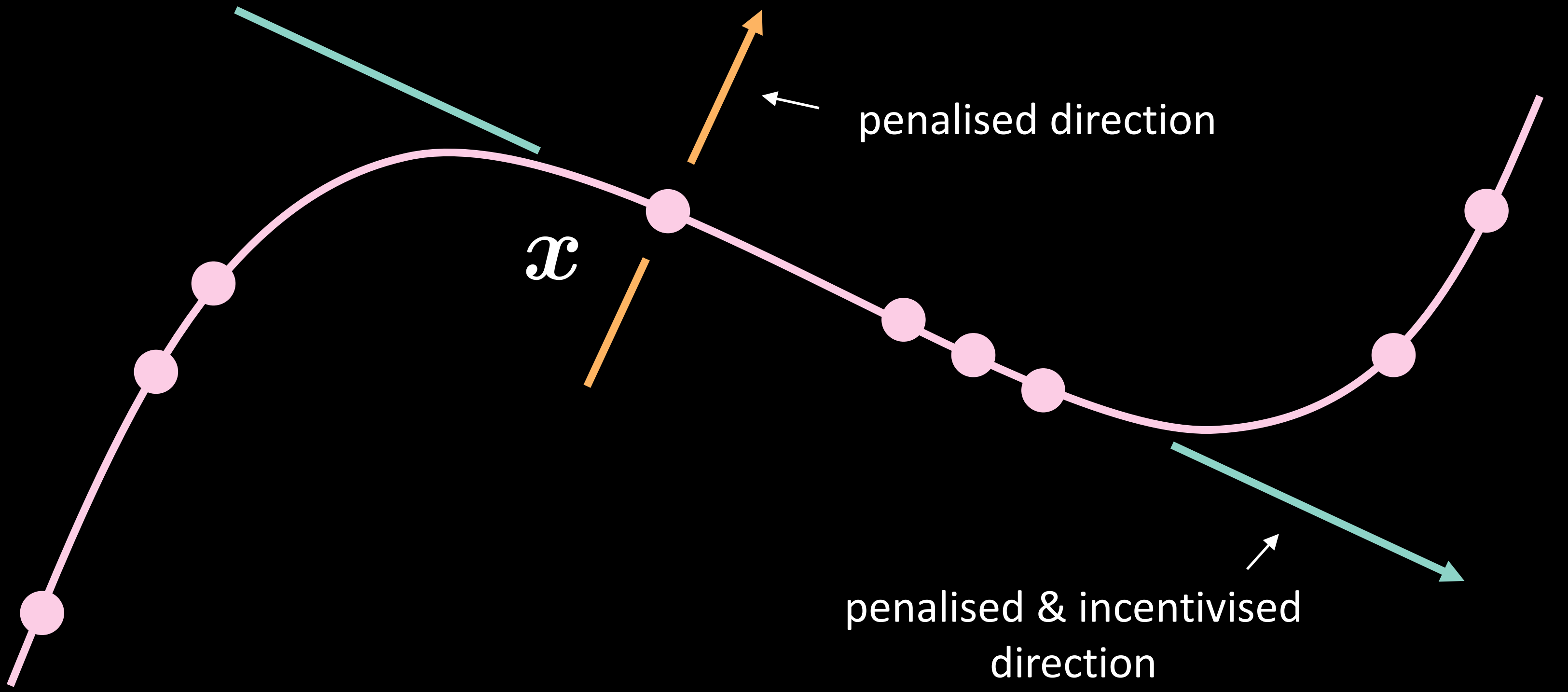

Ω
input
3.0
2.5
2.0
1.5
1.0
0.5
0.0

Contractive autoencoder

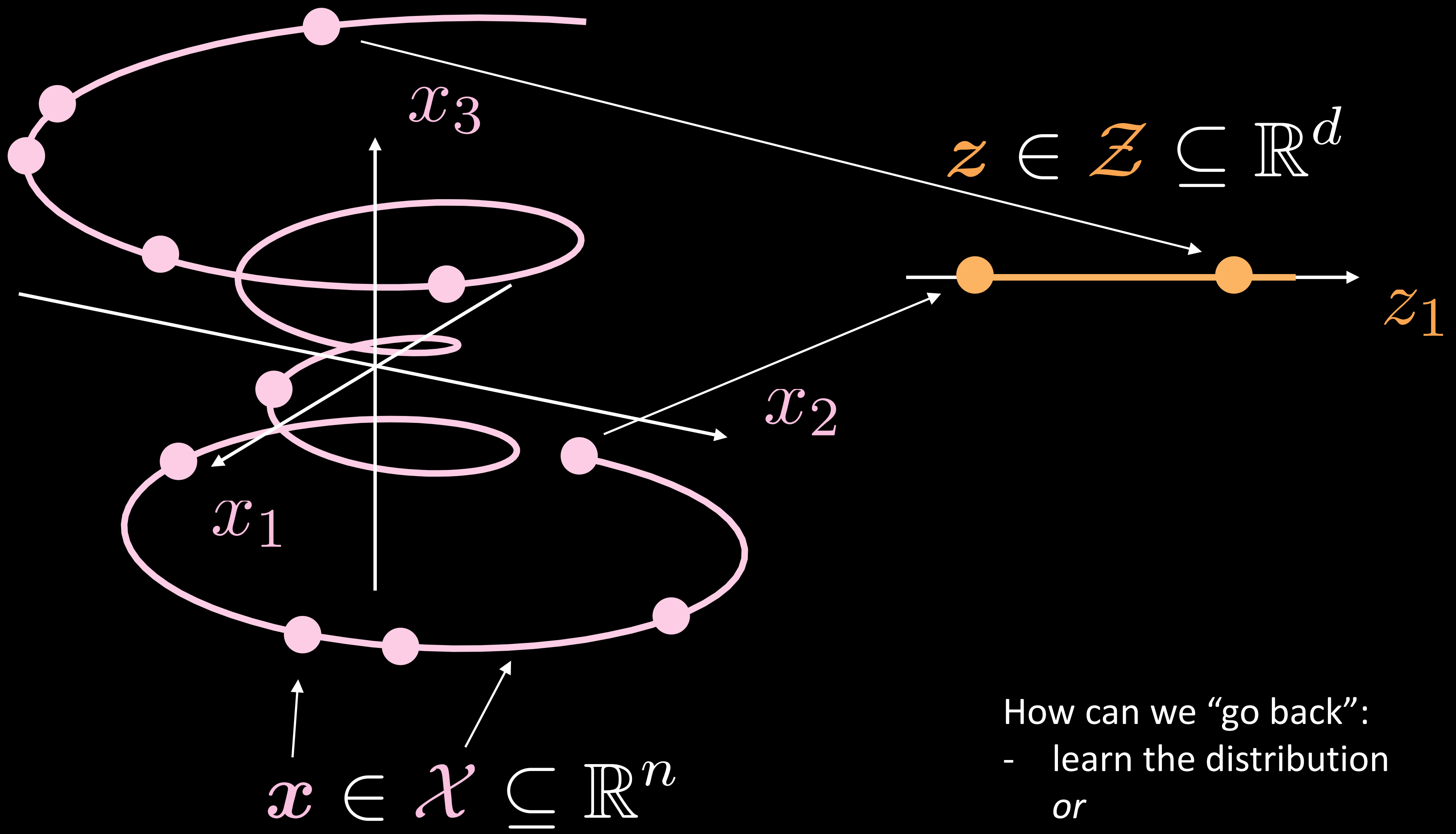
penalises insensitivity to **reconstruction directions**

penalises sensitivity 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

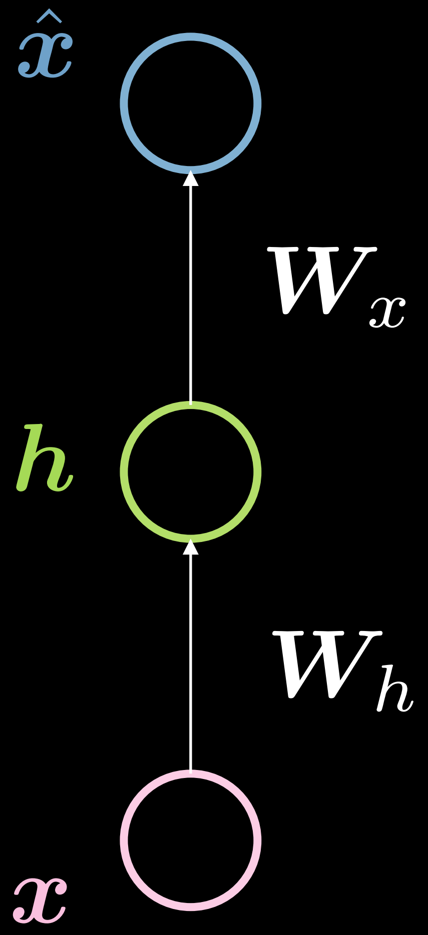


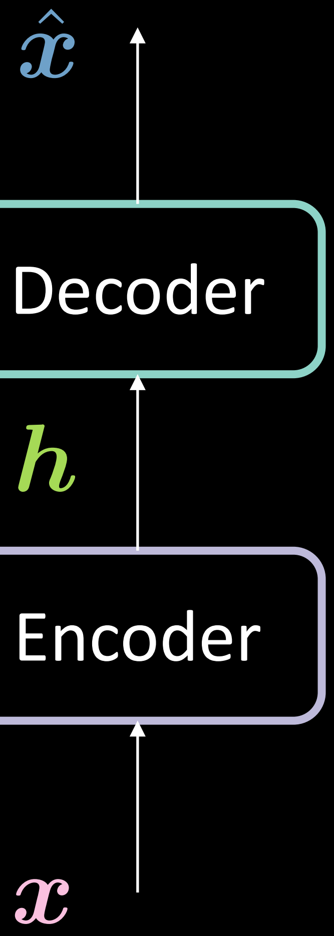
How can we "go back":
- learn the distribution
  or
- enforce some structure

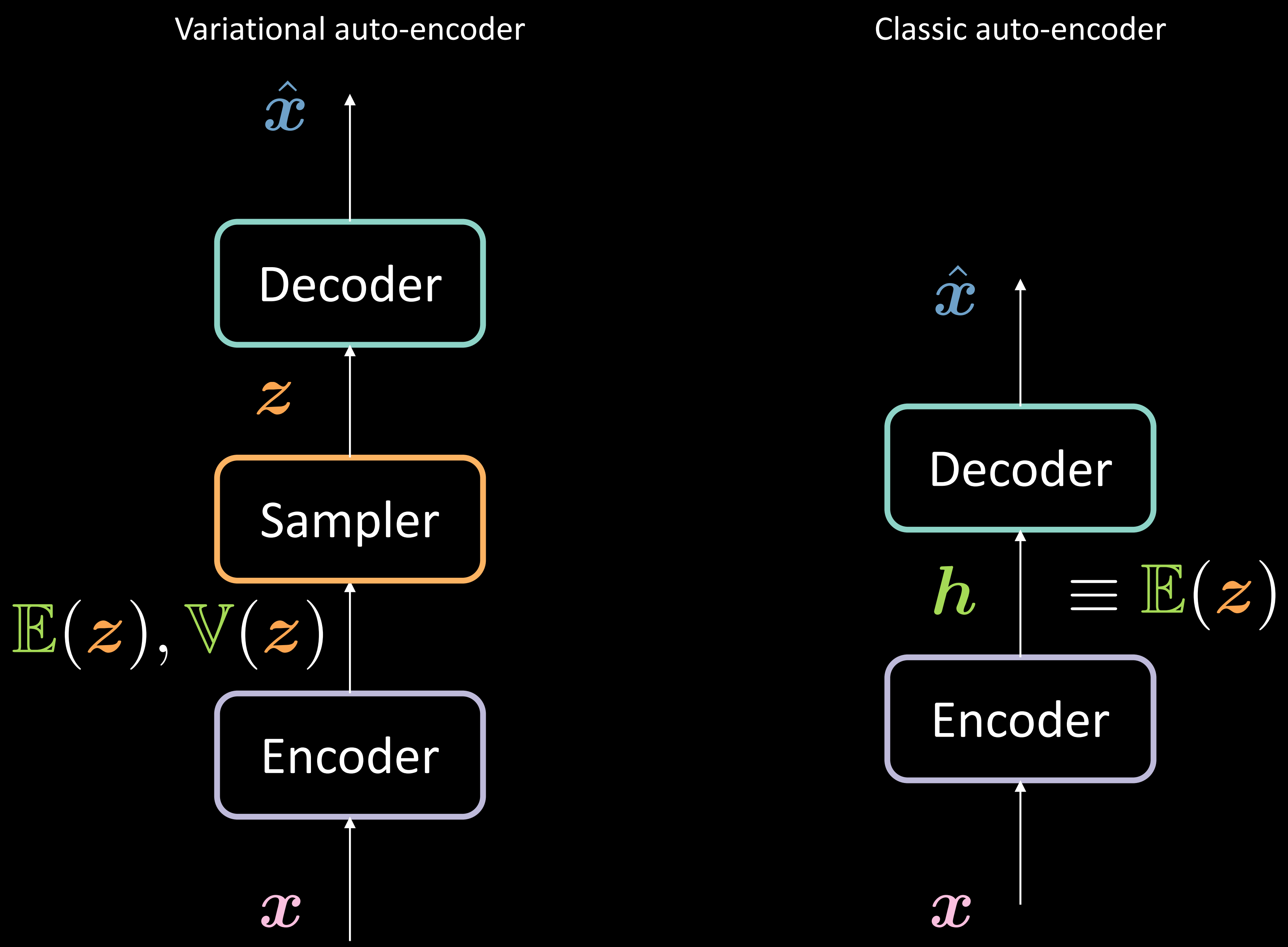
Dealing with distributions

Generative models

Auto-encoder (recap)

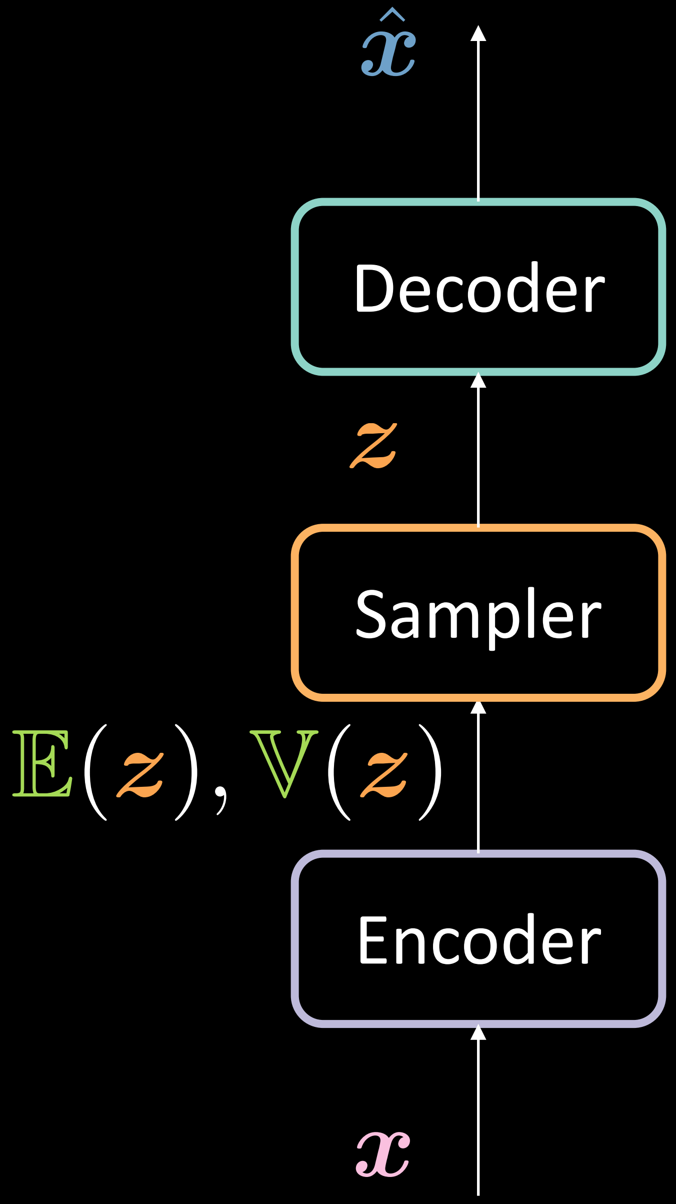






Variational auto-encoder
$\hat{x}$
Decoder
z
Sampler
$\mathbb{E}(z), \mathbb{V}(z)$
Encoder
x
Classic auto-encoder
$\hat{x}$
Decoder
$h \equiv \mathbb{E}(z)$
Encoder
x

Variational auto-encoder



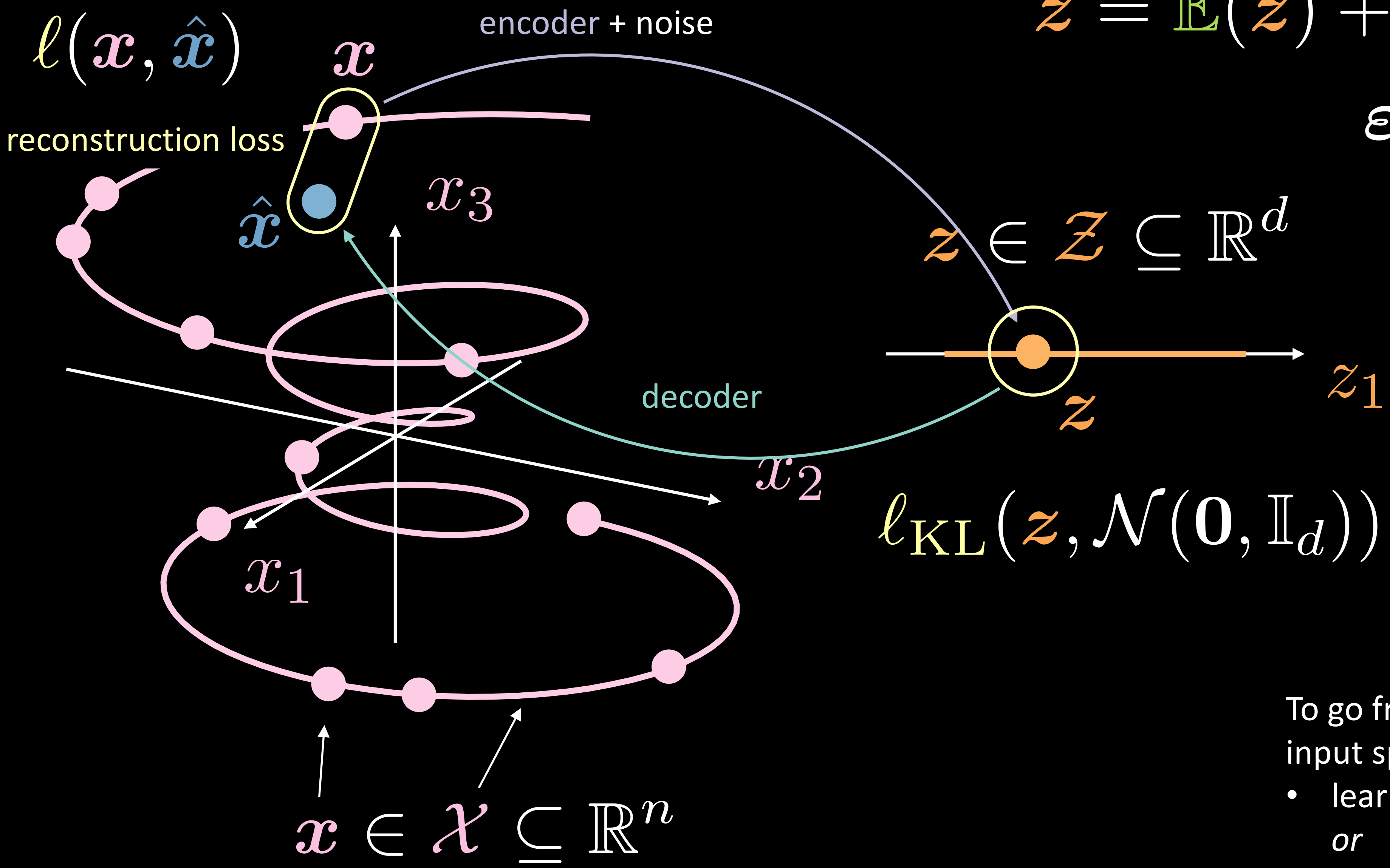
$$\text{decoder} : \mathcal{Z} \to \mathbb{R}^n$$
$$z \mapsto \hat{x}$$

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

Variational auto-encoder

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

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



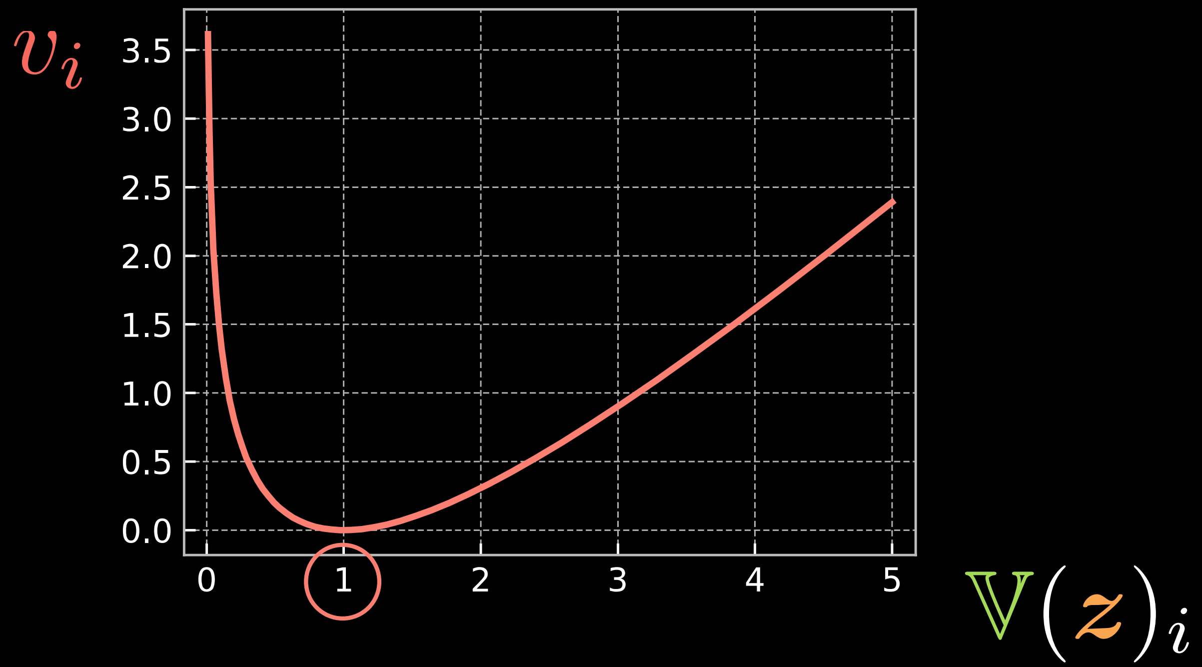
To go from the latent to the input space we need to:

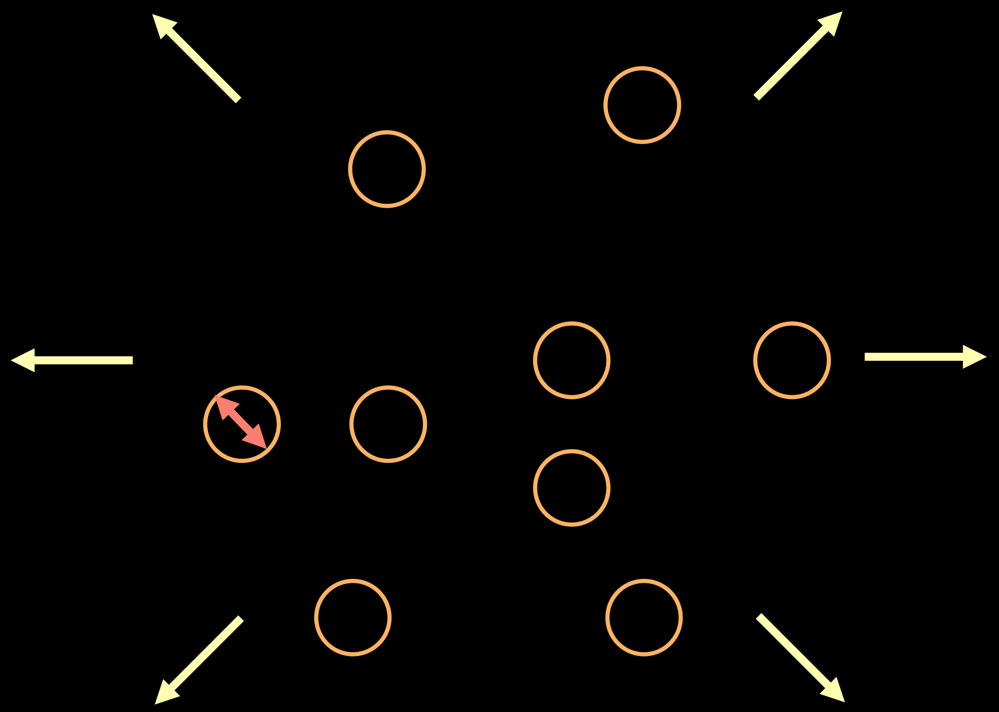
- learn the distribution
  or
- enforce some structure

Variational auto-encoder

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

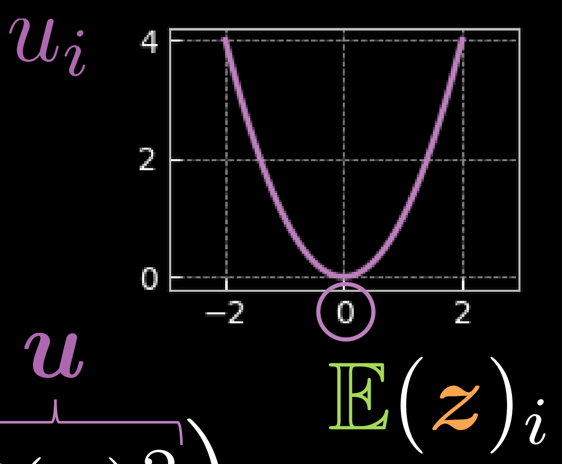
$$= \ell_{\text{reconstruction}} + \frac{\beta}{2} \sum_{i=1}^{d} \Big(\underbrace{\mathbb{V}(\boldsymbol{z}) - \log\left[\mathbb{V}(\boldsymbol{z})\right] - \mathbf{1}}_{\boldsymbol{v}} + \mathbb{E}(\boldsymbol{z})^2 \Big)_i$$

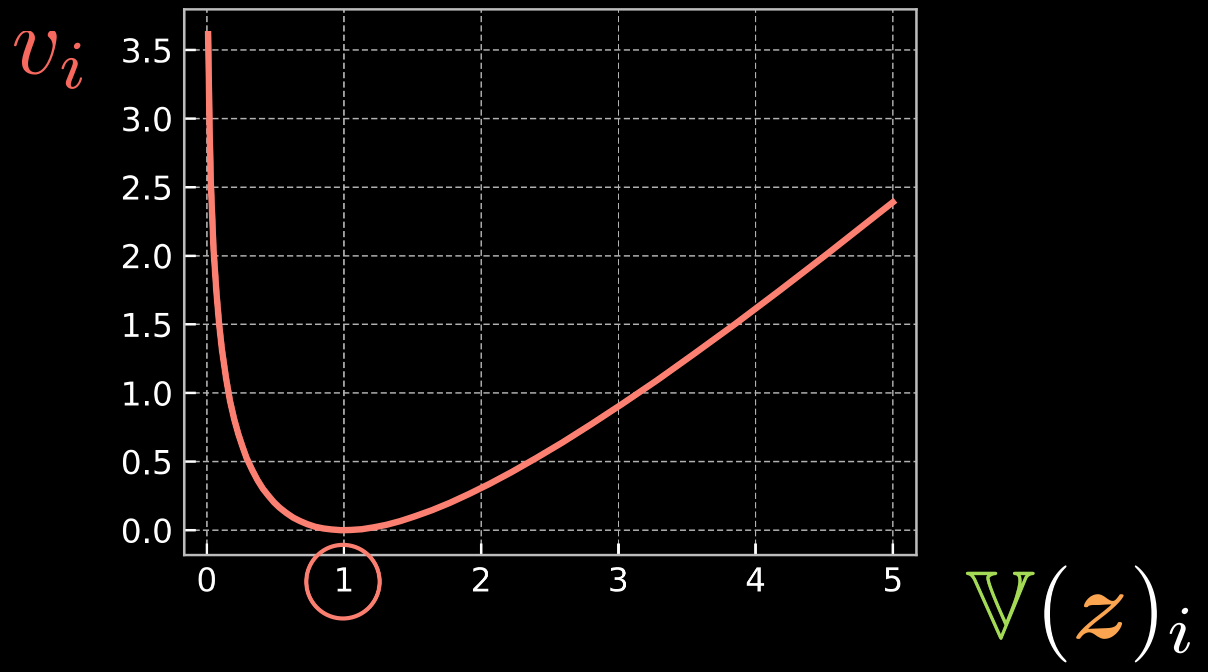


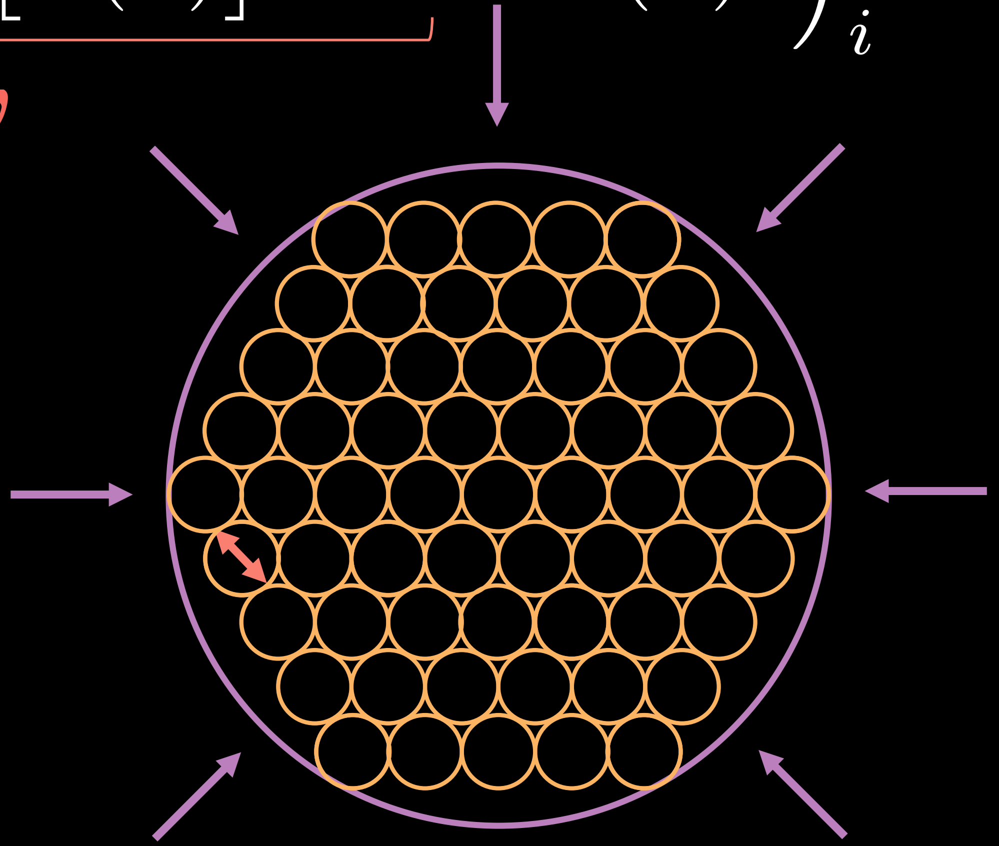


Variational auto-encoder

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

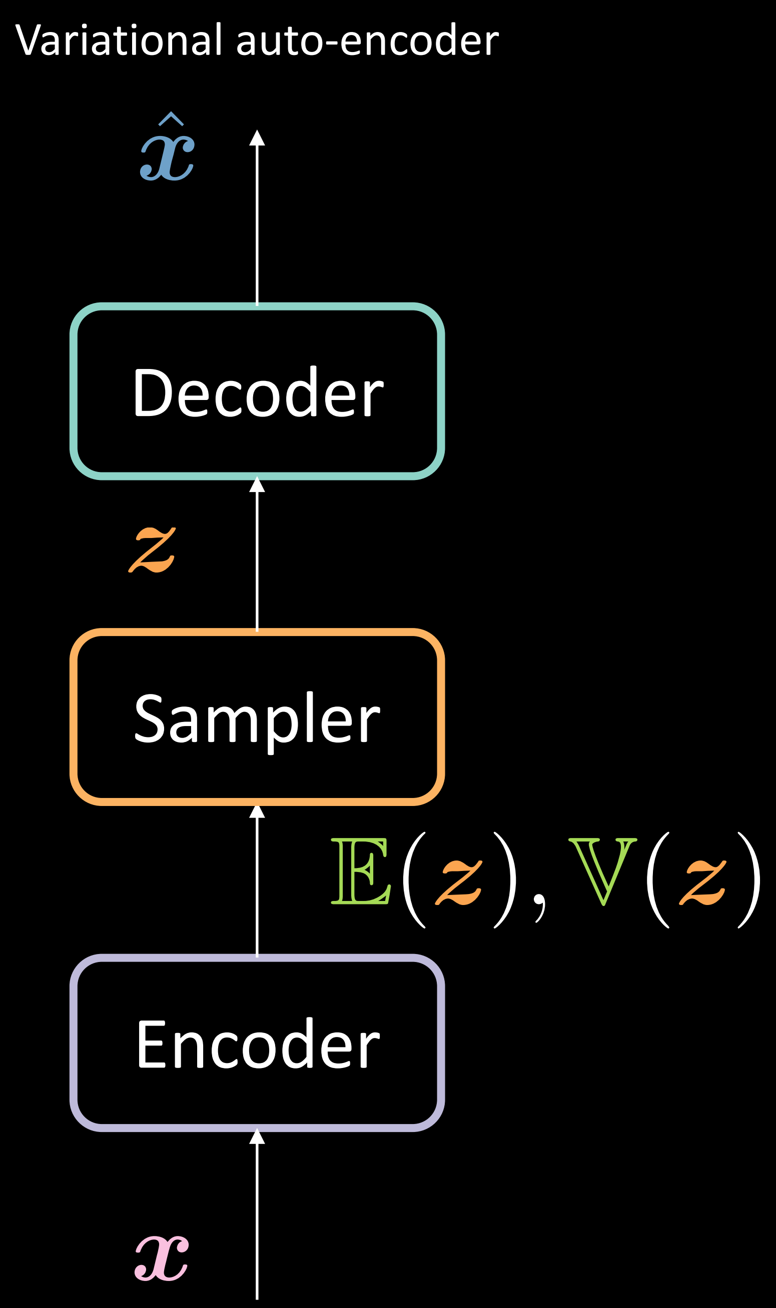


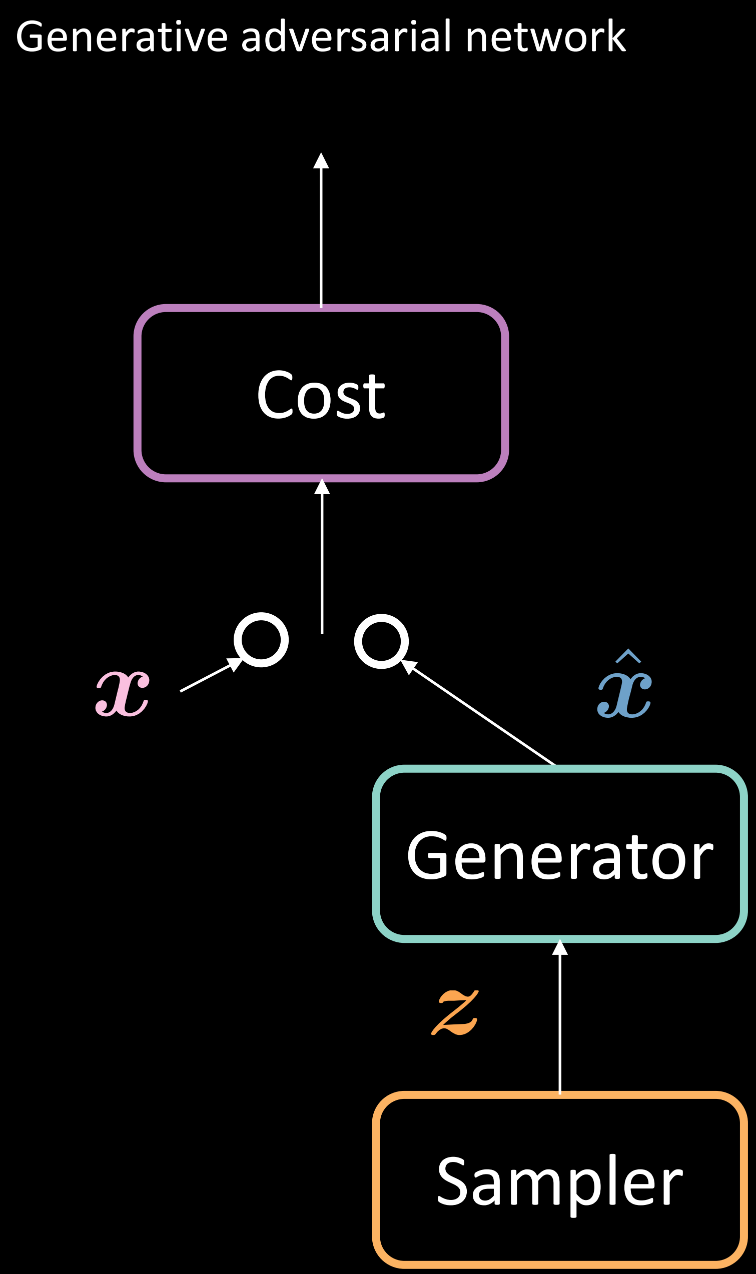




Generative adversarial nets

Unsupervised learning / Generative models


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

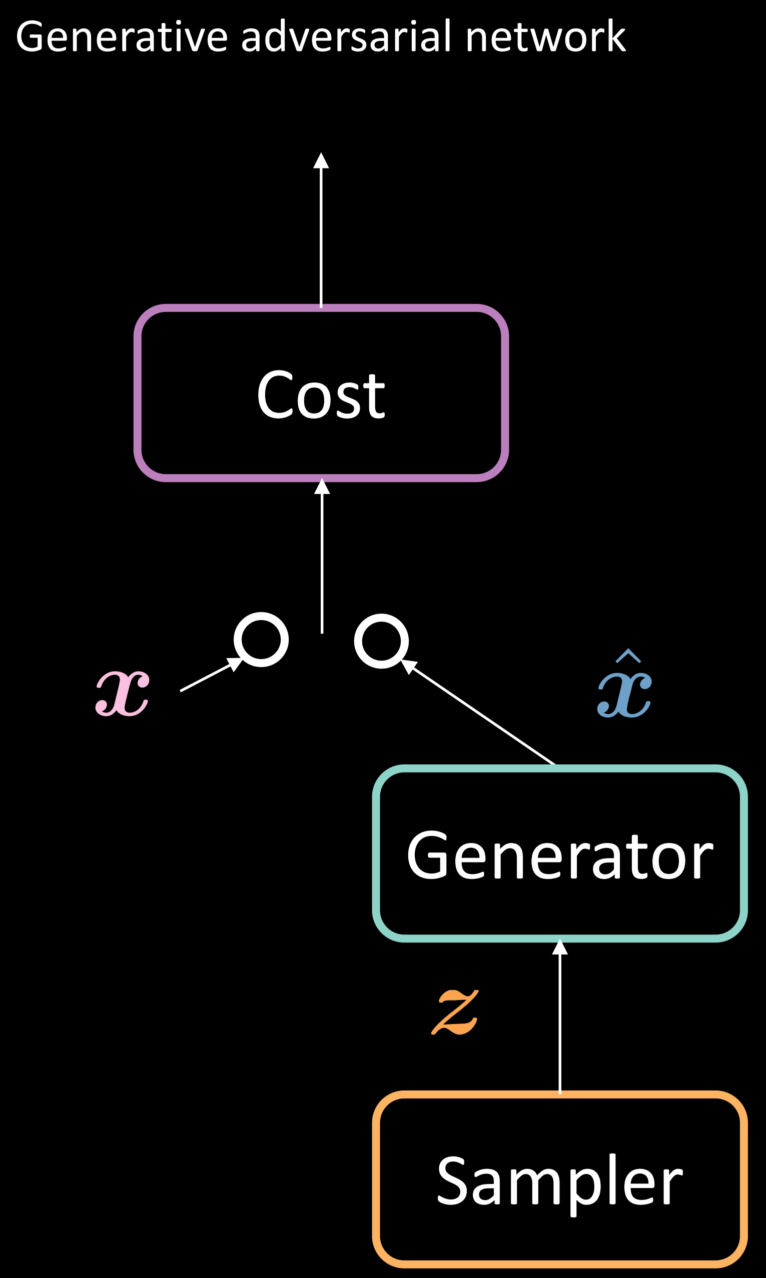

Generative adversarial network
Cost
x
$\hat{x}$
Generator
z
Sampler

$$\text{cost}, C : \mathbb{R}^n \to \mathbb{R}^+ \cup \{0\}$$

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

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

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



Training

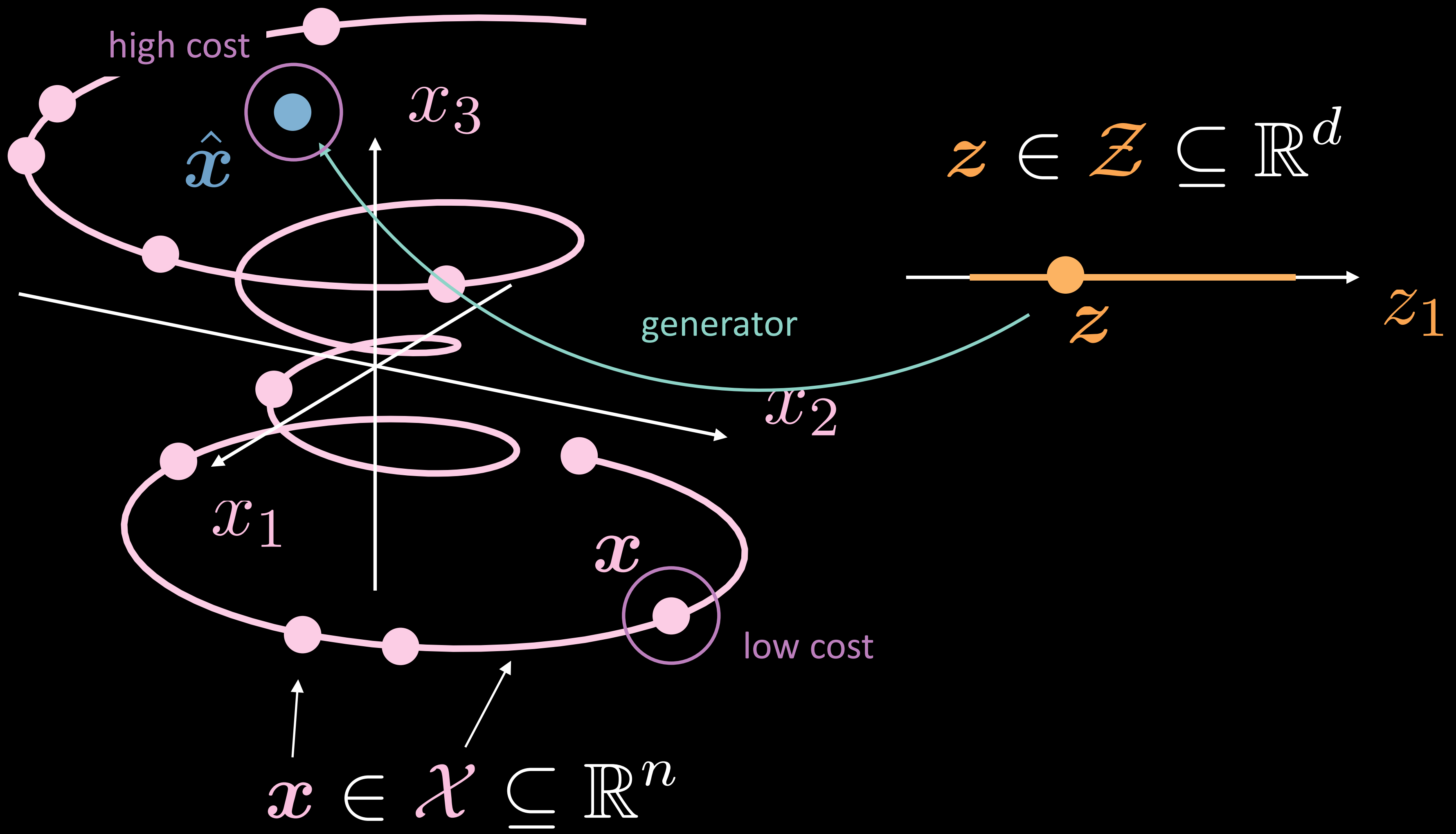
$$\mathcal{L}_C(\boldsymbol{x}, \boldsymbol{z}) = C(\boldsymbol{x}) + [m - C(G(\boldsymbol{z}))]^+$$
$$\mathcal{L}_G(\boldsymbol{z}) = C(G(\boldsymbol{z}))$$

Possible choice of $C(\boldsymbol{x})$:

$$C(\boldsymbol{x}) = \|\mathrm{Dec}(\mathrm{Enc}(\boldsymbol{x})) - \boldsymbol{x}\|^2$$

Zhao, Mathieu, LeCun (2016) Energy-based generative adversarial network

Generative adversarial network



Major pitfalls

- Vanishing gradients
- Mode collapse
- Unstable convergence