A quick tour of deep generative models

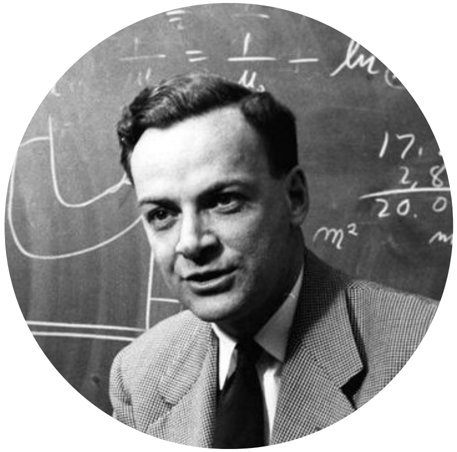
ML-IAP 2021

Prof. Gilles Louppe
g.louppe@uliege.be

A generative model is a probabilistic model p that can be used as a simulator of the data. Its purpose is to generate synthetic but realistic high-dimensional data

$$\mathbf{x} \sim p(\mathbf{x}; \theta),$$

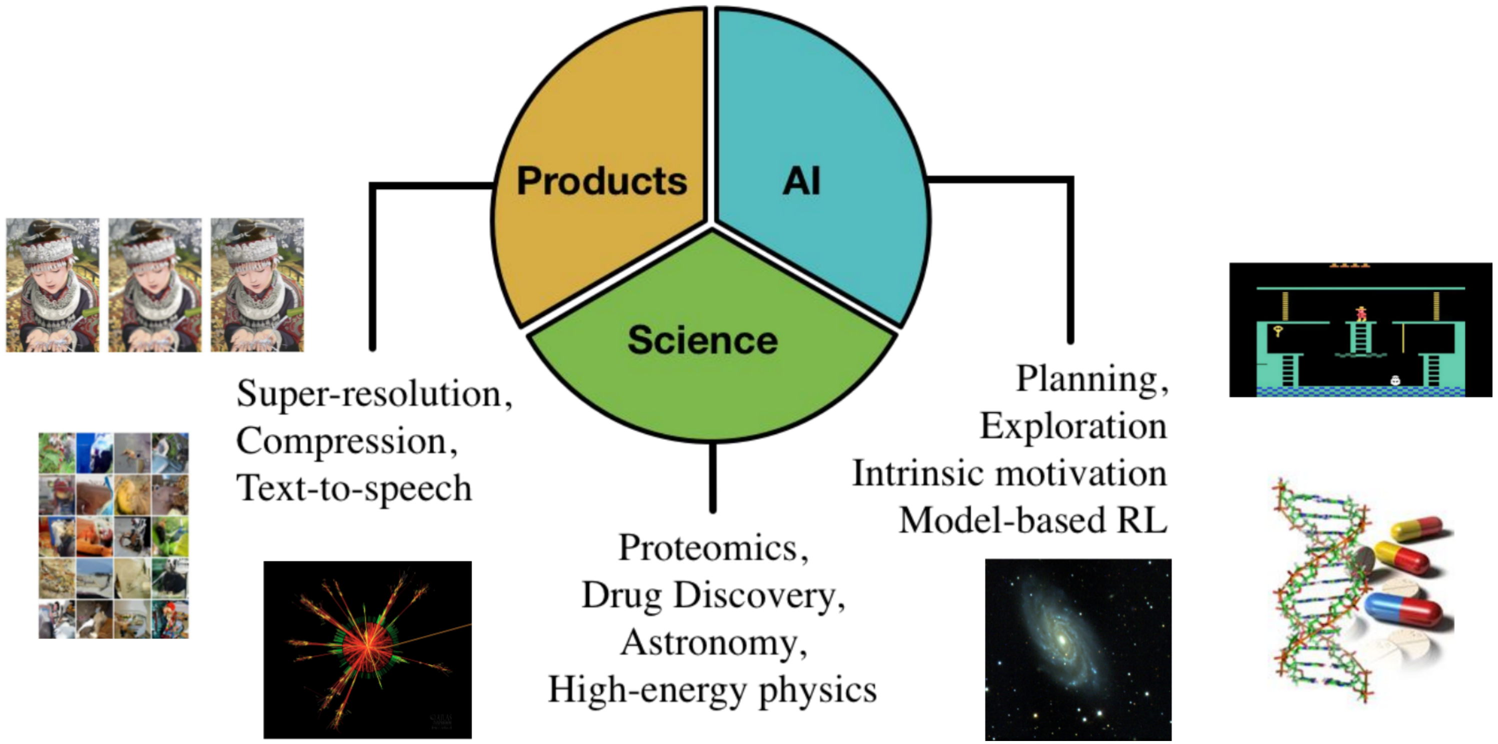
that is as close as possible from the unknown data distribution $p(\mathbf{x})$.



"What I cannot create, I do not understand."

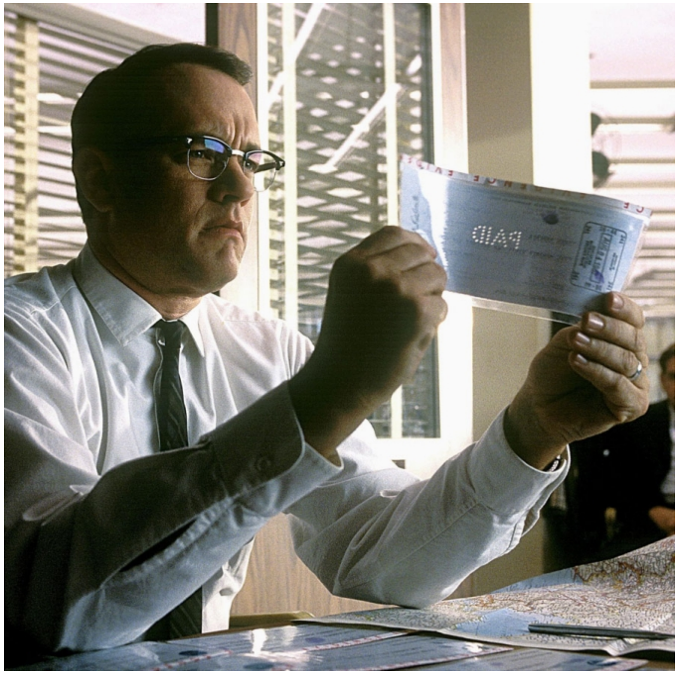
Richard Feynman

Why generative models

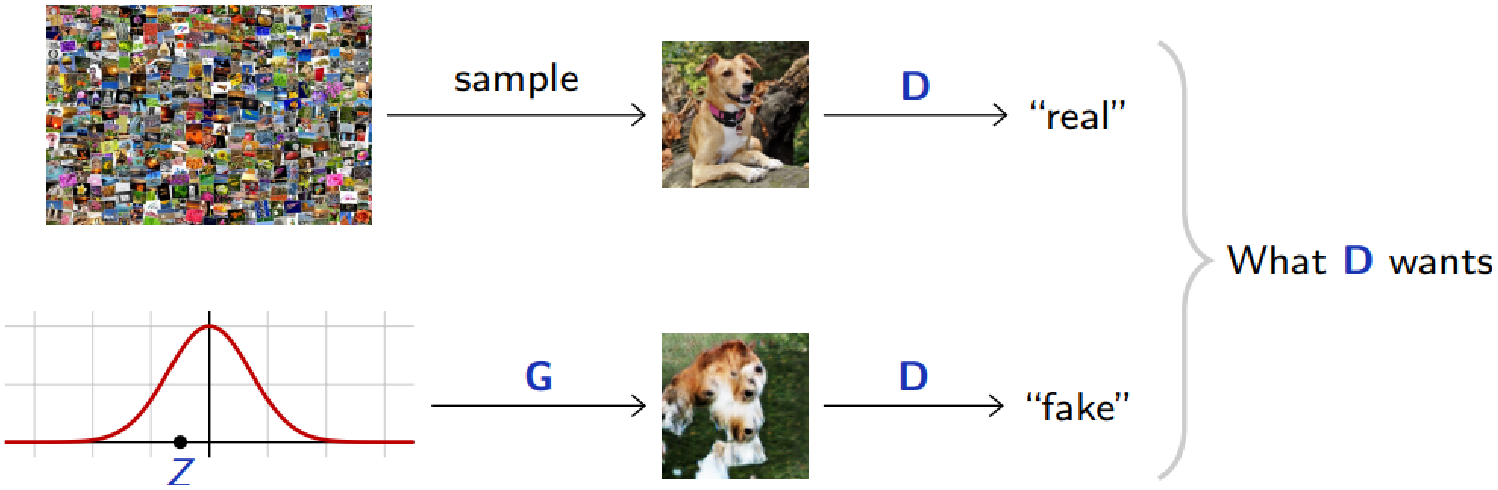


Generative models have a role in many important problems

Part I: Generative adversarial networks

SEP 07 1965
PAID

PAID

Generative adversarial networks



In generative adversarial networks (GANs), the task of learning a generative model is expressed as a two-player zero-sum game between two networks.

Architecture

The first network is a generator $g(\cdot;\theta) : \mathcal{Z} \to \mathcal{X}$, mapping a latent space equipped with a prior distribution $p(\mathbf{z})$ to the data space, thereby inducing a distribution

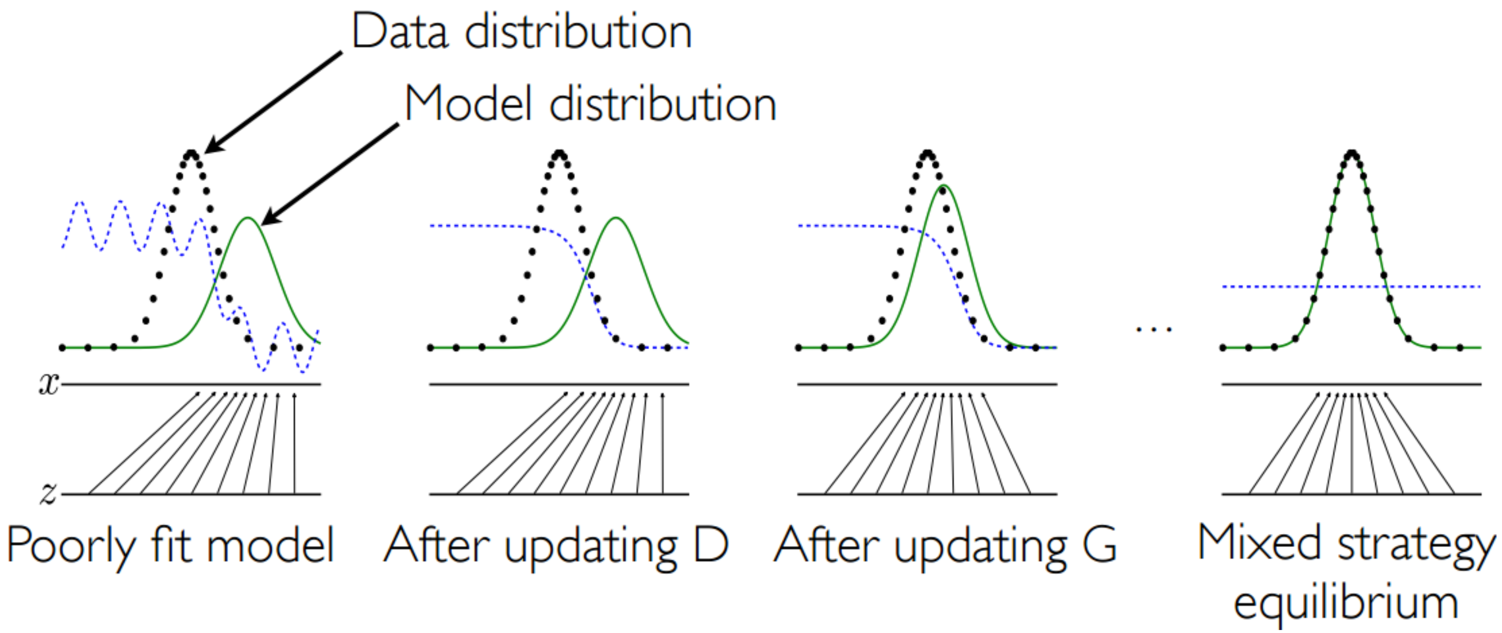
$$\mathbf{x} \sim q(\mathbf{x};\theta) \Leftrightarrow \mathbf{z} \sim p(\mathbf{z}), \mathbf{x} = g(\mathbf{z};\theta).$$

The second network $d(\cdot;\phi) : \mathcal{X} \to [0,1]$ is a classifier trained to distinguish between true samples $\mathbf{x} \sim p(\mathbf{x})$ and generated samples $\mathbf{x} \sim q(\mathbf{x};\theta)$.

Training

$$\min_{\theta} \max_{\phi} V(\phi, \theta) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[\log d(\mathbf{x}; \phi) \right] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \left[\log(1 - d(g(\mathbf{z}; \theta); \phi)) \right]$$

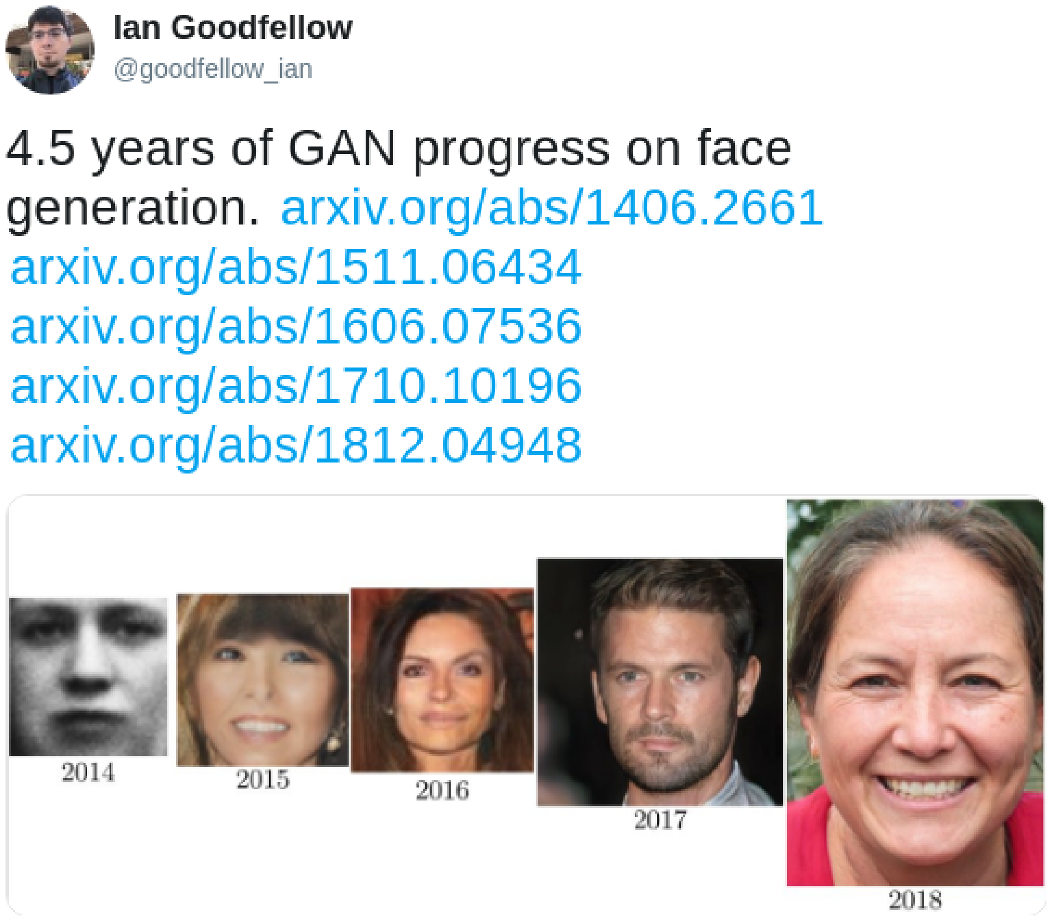
- For a fixed g, $V(\phi, \theta)$ is high if d is good at recognizing true from generated samples.
- If d is the best classifier given g, and if V is high, then this implies that the generator is bad at reproducing the data distribution.
- Conversely, g will be a good generative model if V is low when d is a perfect opponent.



—

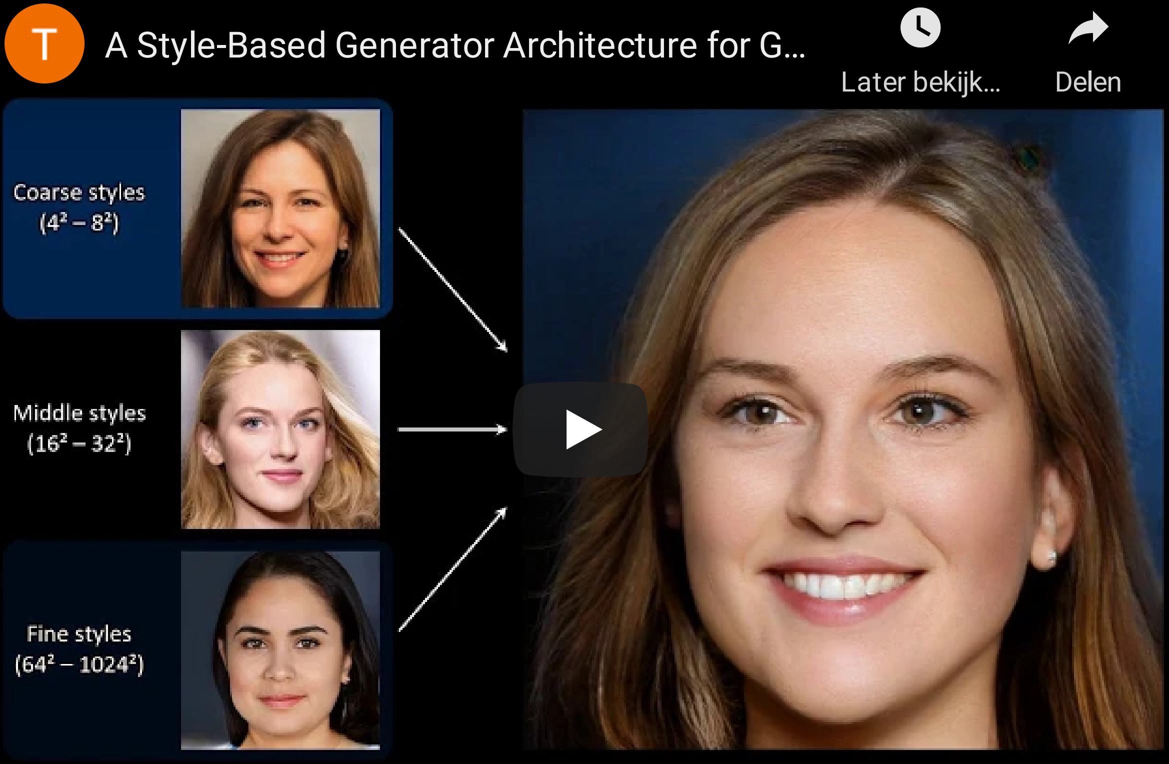
Credits: Goodfellow et al, Generative Adversarial Networks, 2014.

Examples

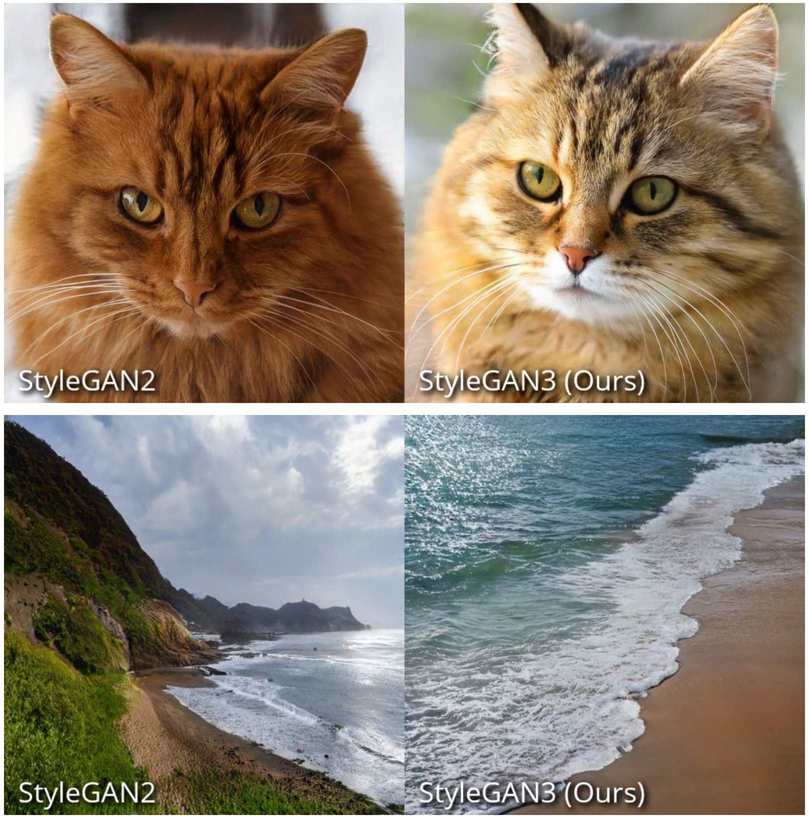


Ian Goodfellow
@goodfellow_ian

4.5 years of GAN progress on face generation. arxiv.org/abs/1406.2661 arxiv.org/abs/1511.06434 arxiv.org/abs/1606.07536 arxiv.org/abs/1710.10196 arxiv.org/abs/1812.04948



StyleGAN (v1) (Karras et al, 2018)



StyleGAN (v2, v3) (Karras et al, 2021)

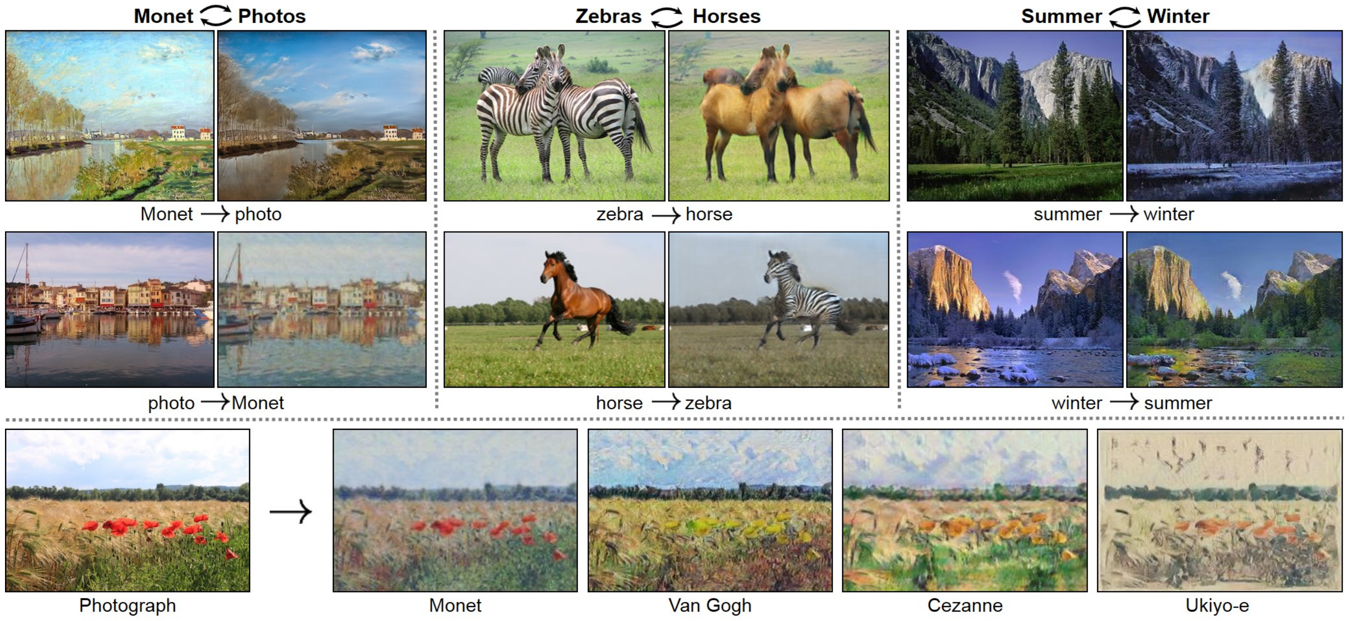
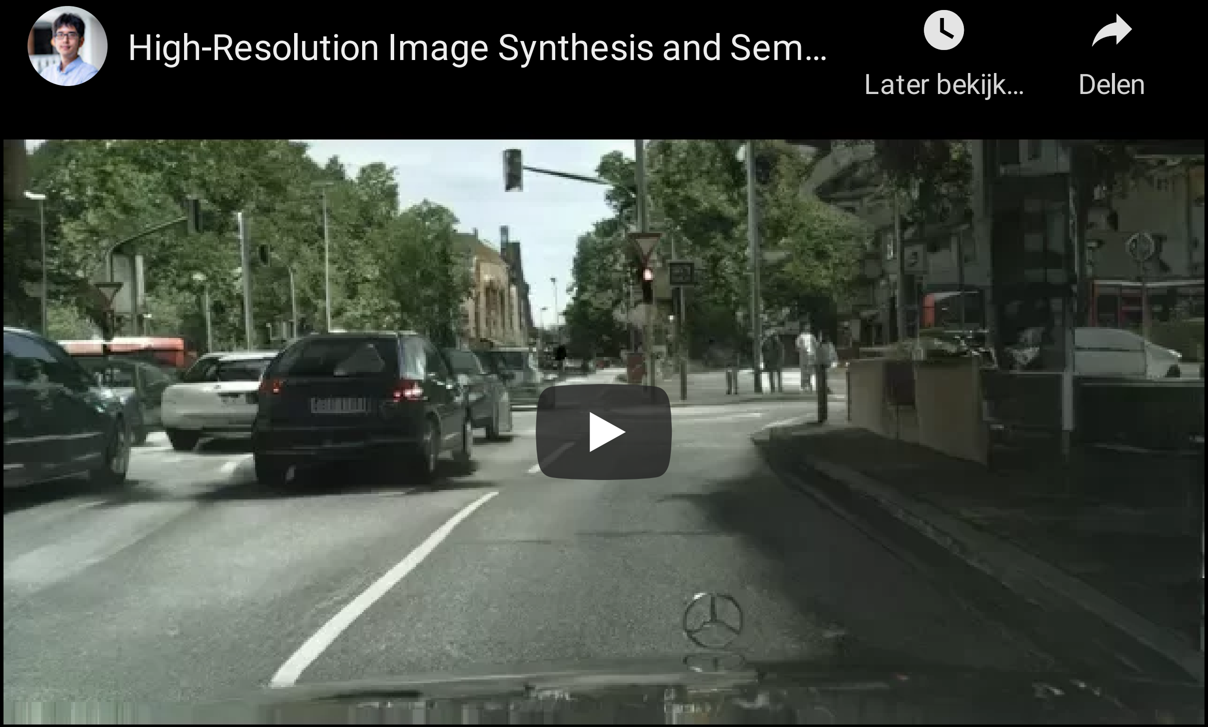
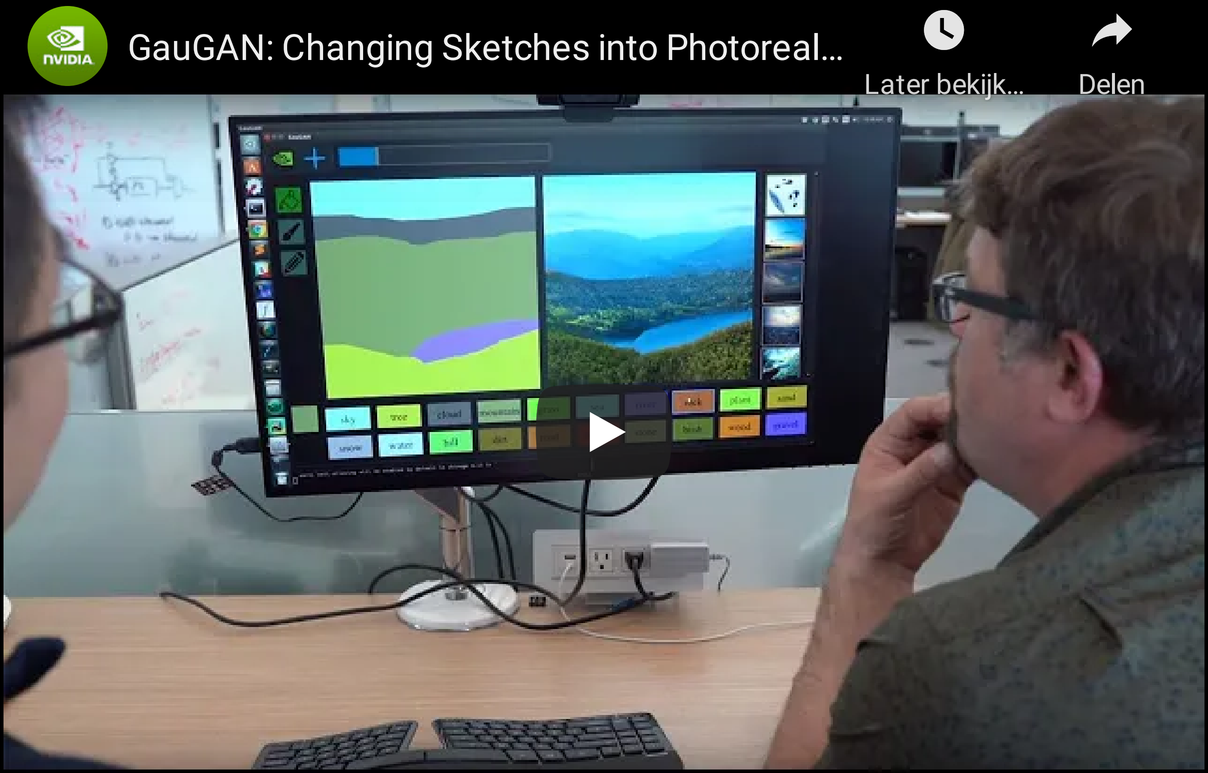


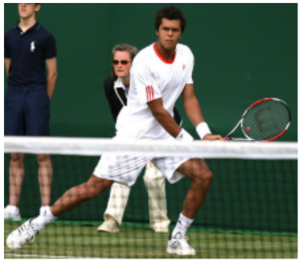
Image-to-image translation (Zhu et al, 2017)

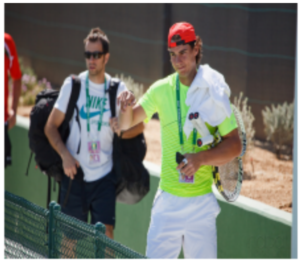


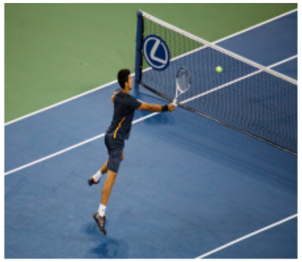
High-resolution image synthesis (Wang et al, 2017)

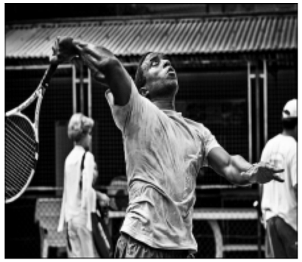


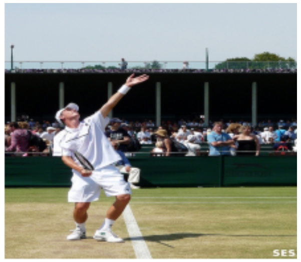
GauGAN: **Changing sketches into photorealistic masterpieces** (NVIDIA, 2019)

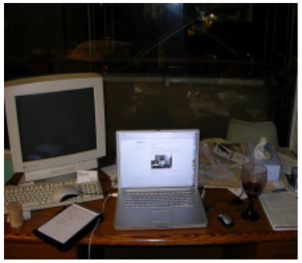

a tennis player gets ready to return a serve

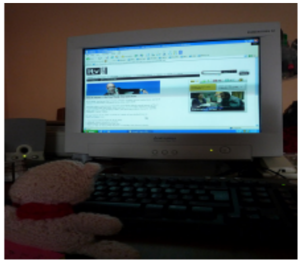

two men dressed in costumes and holding tennis rackets

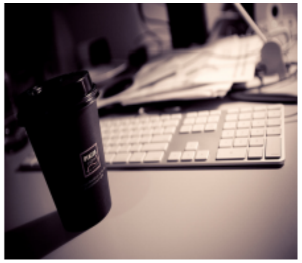

a tennis player hits the ball during a match

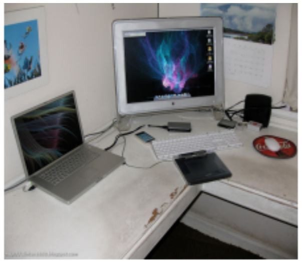

a male tennis player in action on the court

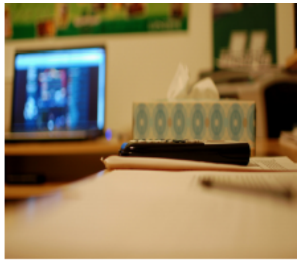

a man in white is about to serve a tennis ball


a laptop and a desktop computer sit on a desk


a person is working on a computer screen


a cup of coffee sitting next to a laptop


a laptop computer sitting on top of a desk next to a


a picture of a computer on a desk

Captioning (Shetty et al, 2017)

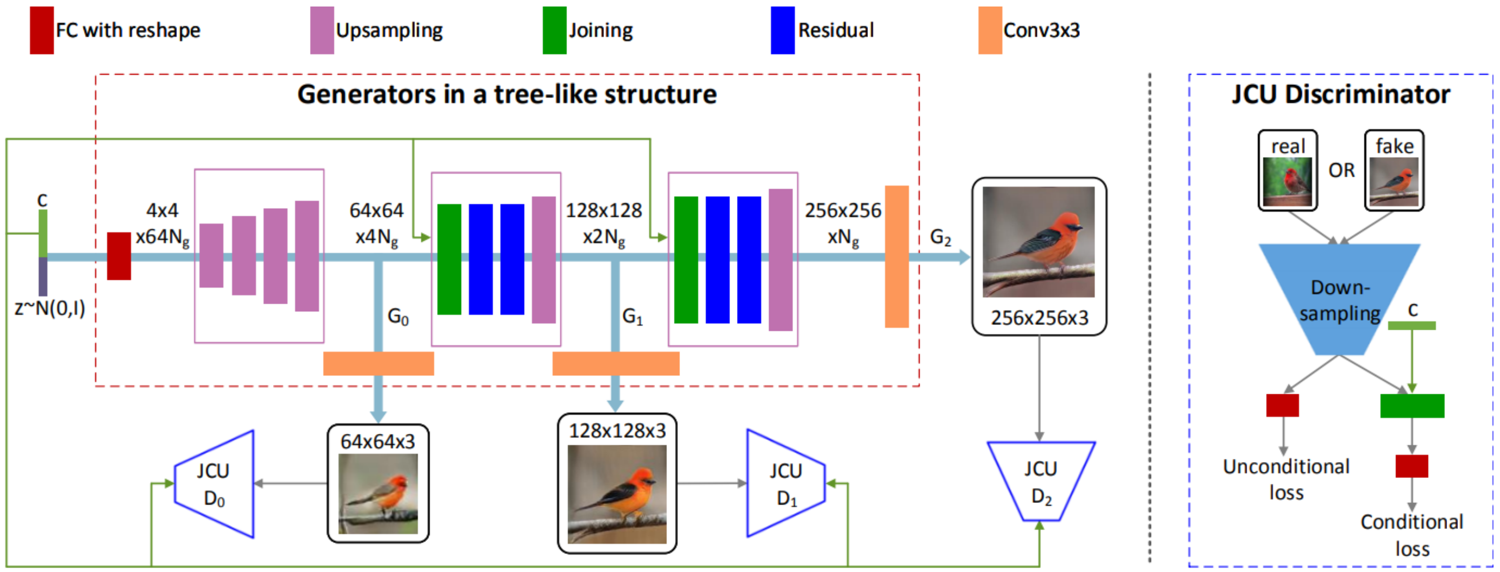


Fig. 2: The overall framework of our proposed StackGAN-v2 for the conditional image synthesis task. c is the vector of conditioning variables which can be computed from the class label, the text description, *etc.*. N_g and N_d are the numbers of channels of a tensor.

Text-to-image synthesis (Zhang et al, 2017)

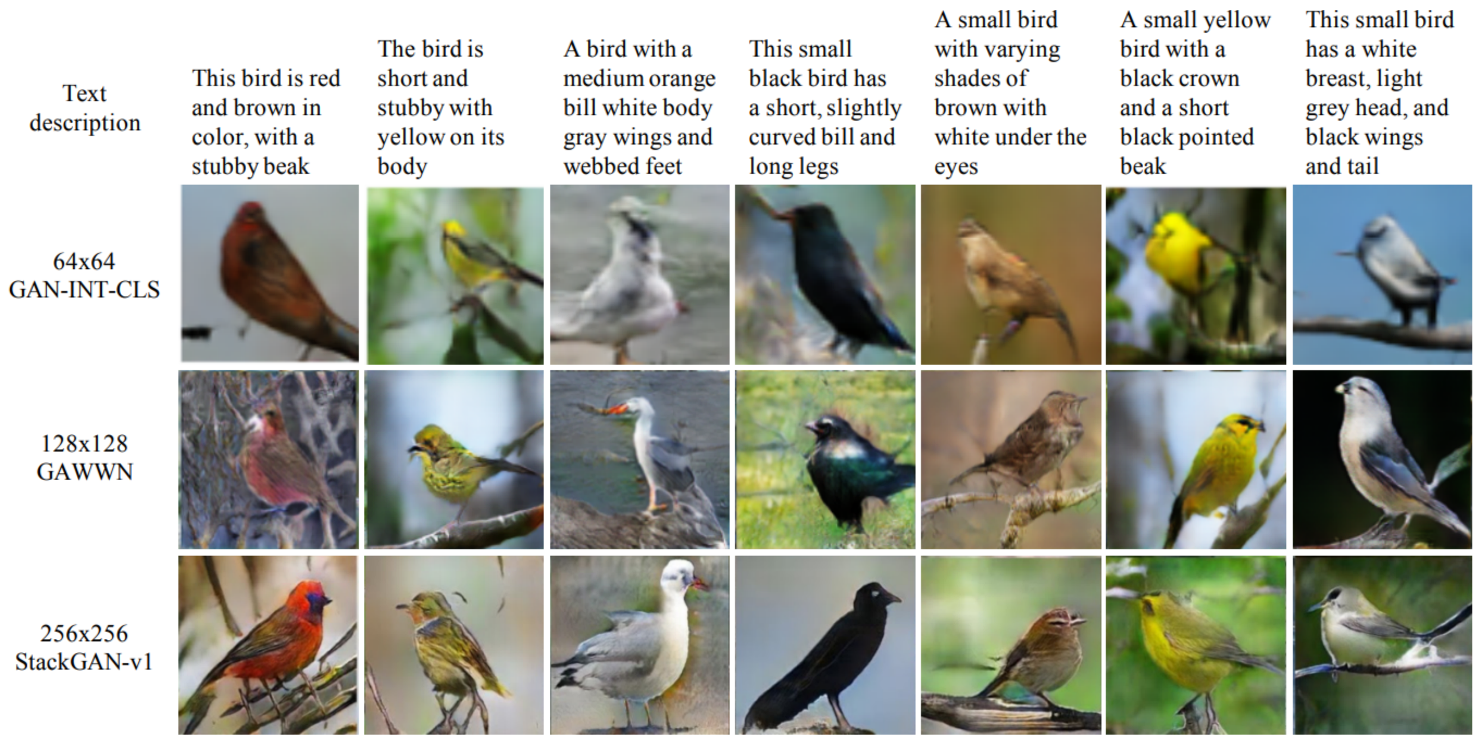

Text description
This bird is red and brown in color, with a stubby beak
The bird is short and stubby with yellow on its body
A bird with a medium orange bill white body gray wings and webbed feet
This small black bird has a short, slightly curved bill and long legs
A small bird with varying shades of brown with white under the eyes
A small yellow bird with a black crown and a short black pointed beak
This small bird has a white breast, light grey head, and black wings and tail
64x64 GAN-INT-CLS
128x128 GAWWN
256x256 StackGAN-v1

Fig. 3: Example results by our StackGAN-v1, GAWWN [29], and GAN-INT-CLS [31] conditioned on text descriptions from CUB test set.

(Zhang et al, 2017)

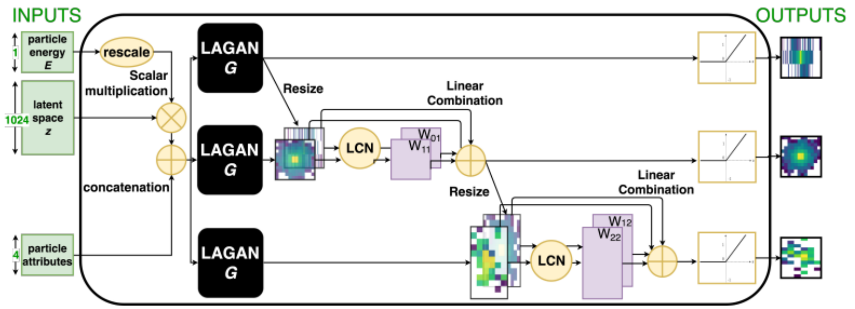


Figure 8.37: Composite conditional CaloGAN generator G, with three LAGAN-like streams connected by attentional layer-to-layer dependence.

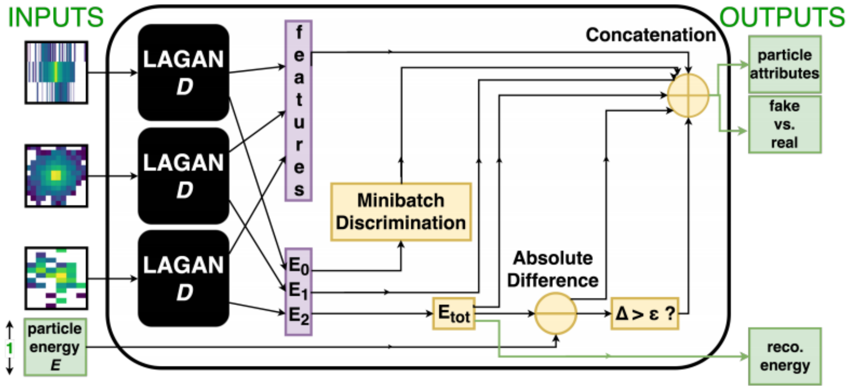
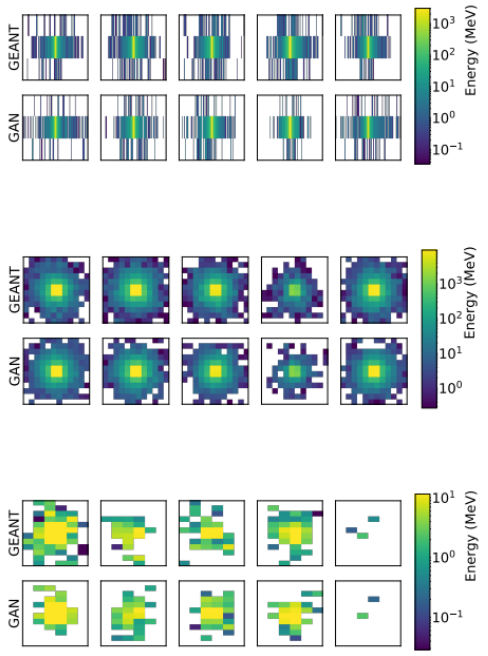


Figure 8.38: Composite conditional CaloGAN discriminator D, with three LAGAN-like streams and additional domain-specific energy calculations included in the final feature space.



Learning particle physics (Paganini et al, 2017)

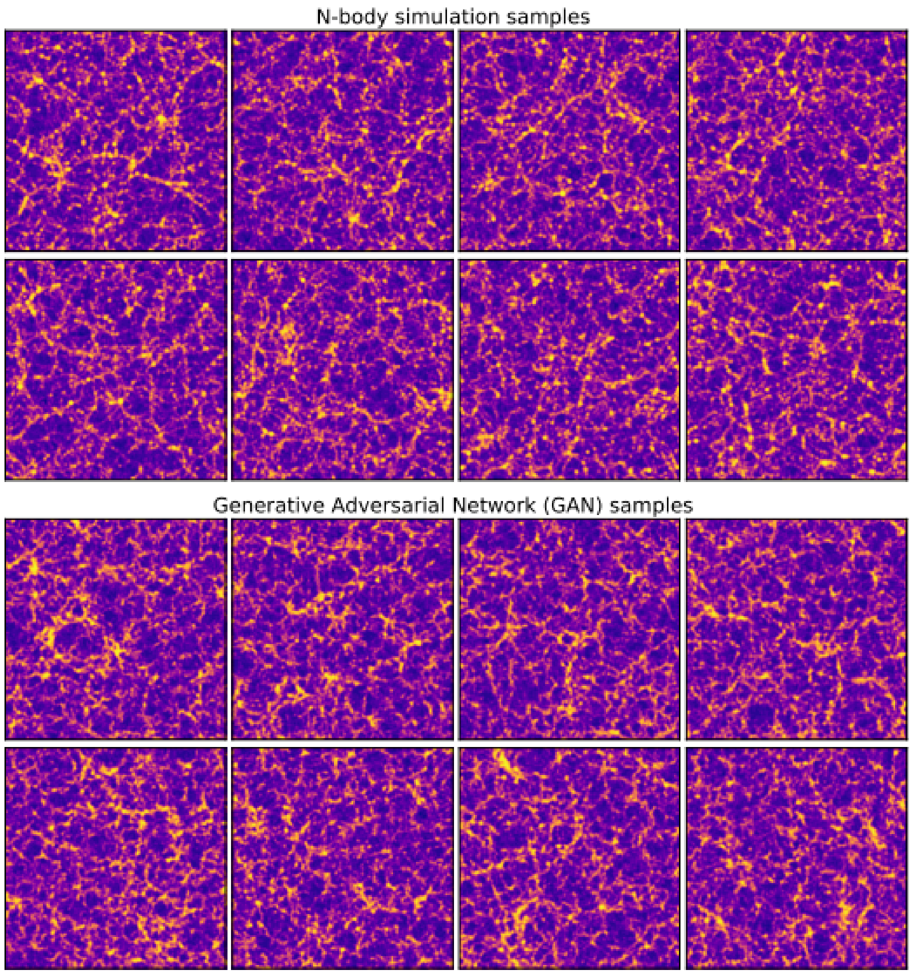
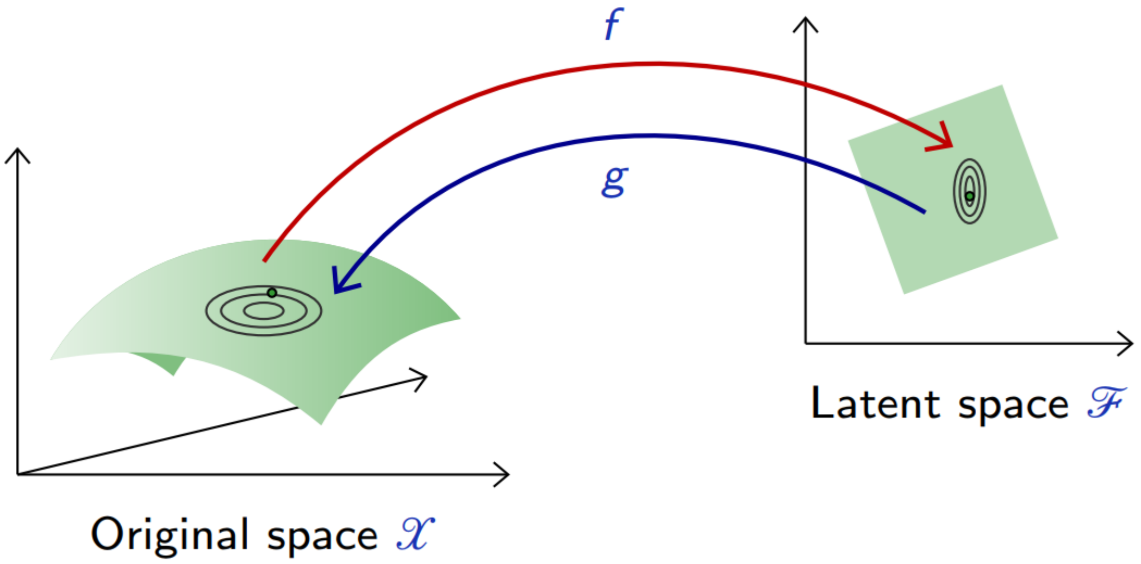

N-body simulation samples
Generative Adversarial Network (GAN) samples

Figure 1: Samples from N-body simulation and from GAN for the box size of 500 Mpc. Note that the transformation in Equation 3.1 with $a = 20$ was applied to the images shown above for better clarity.

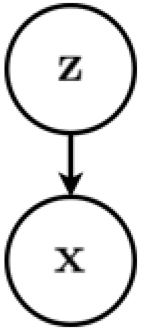
Learning cosmological models (Rodriguez et al, 2018)

Part II: Variational auto-encoders


f
g
Latent space $\mathscr{F}$
Original space $\mathscr{X}$

Variational auto-encoders

A variational auto-encoder is a deep latent variable model where:



- The prior $p(\mathbf{z})$ is prescribed, and usually chosen to be Gaussian.
- The likelihood $p(\mathbf{x}|\mathbf{z};\theta)$ is parameterized with a generative network NN_θ (or decoder) that takes as input $\mathbf{z}$ and outputs parameters $\phi = \text{NN}_\theta(\mathbf{z})$ to the data distribution. E.g.,

$$\mu, \sigma = \text{NN}_\theta(\mathbf{z})$$
$$p(\mathbf{x}|\mathbf{z};\theta) = \mathcal{N}(\mathbf{x}; \mu, \sigma^2\mathbf{I})$$

- The approximate posterior $q(\mathbf{z}|\mathbf{x};\varphi)$ is parameterized with an inference network NN_φ (or encoder) that takes as input $\mathbf{x}$ and outputs parameters $\nu = \text{NN}_\varphi(\mathbf{x})$ to the approximate posterior. E.g.,

$$\mu, \sigma = \text{NN}_\varphi(\mathbf{x})$$
$$q(\mathbf{z}|\mathbf{x};\varphi) = \mathcal{N}(\mathbf{z}; \mu, \sigma^2\mathbf{I})$$

We can use variational inference to jointly optimize the generative and the inference networks parameters θ and φ.

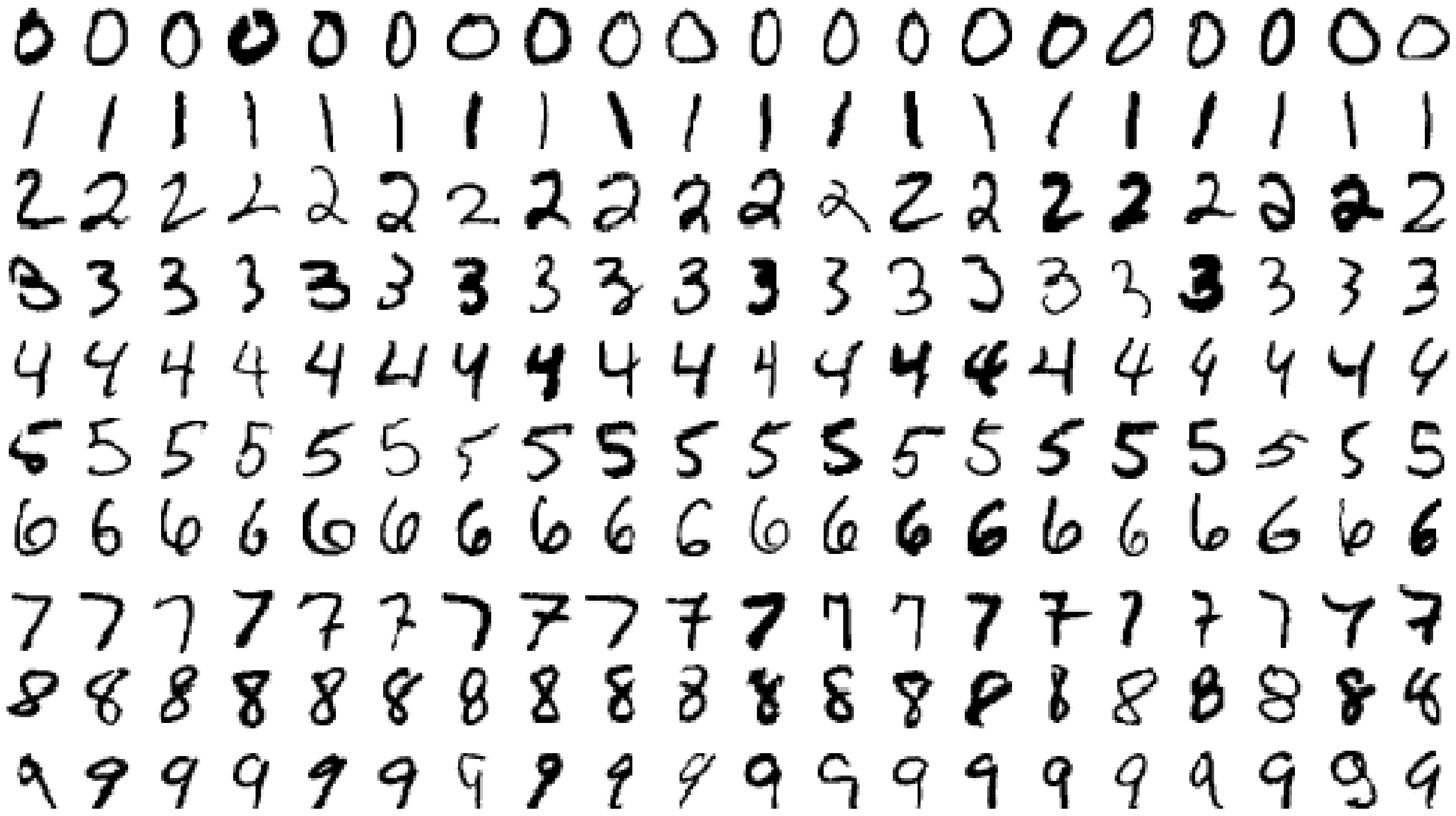
We want

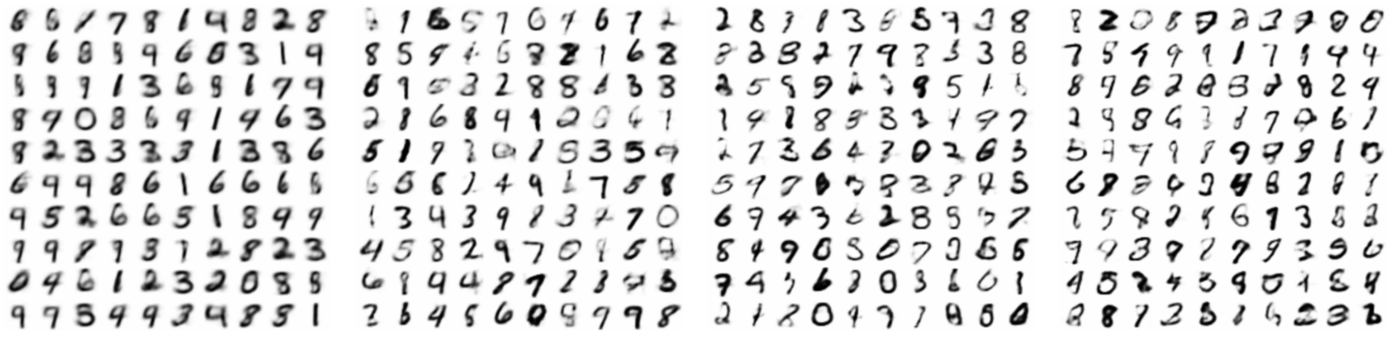
$$\begin{aligned}\theta^*, \varphi^* &= \arg\max_{\theta,\varphi} \text{ELBO}(\mathbf{x}; \theta, \varphi) \\ &= \arg\max_{\theta,\varphi} \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[\log p(\mathbf{x}, \mathbf{z}; \theta) - \log q(\mathbf{z}|\mathbf{x}; \varphi)\right] \\ &= \arg\max_{\theta,\varphi} \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\varphi)}\left[\log p(\mathbf{x}|\mathbf{z}; \theta)\right] - \text{KL}(q(\mathbf{z}|\mathbf{x}; \varphi)||p(\mathbf{z})).\end{aligned}$$

- Given some generative network θ, we want to put the mass of the latent variables, by adjusting φ, such that they explain the observed data, while remaining close to the prior.
- Given some inference network φ, we want to put the mass of the observed variables, by adjusting θ, such that they are well explained by the latent variables.

Examples

Consider as data $\mathbf{d}$ the MNIST digit dataset:

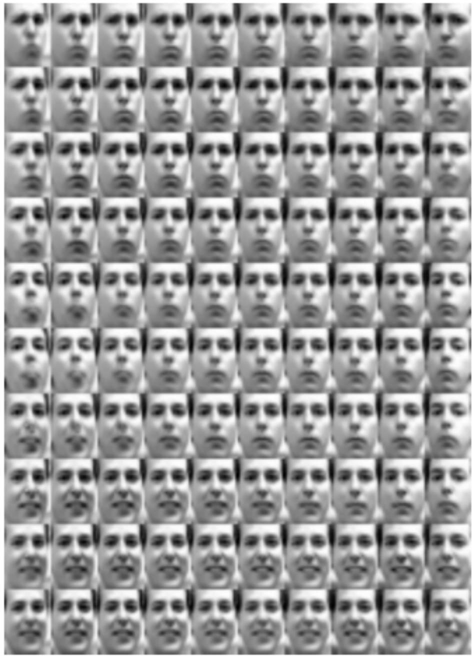




(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

(Kingma and Welling, 2013)


(a) Learned Frey Face manifold

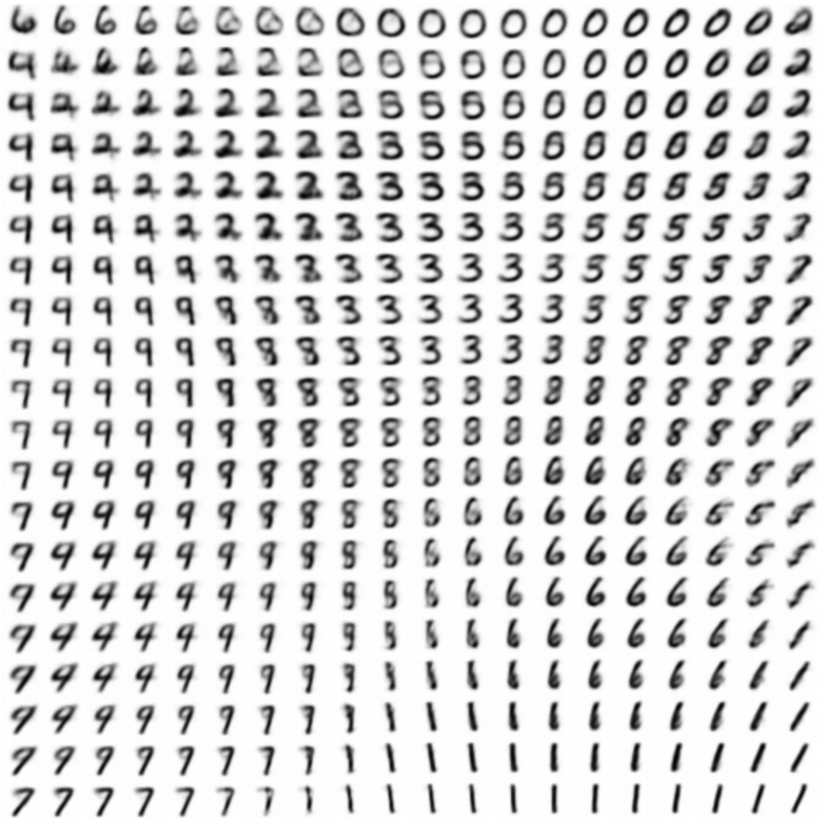
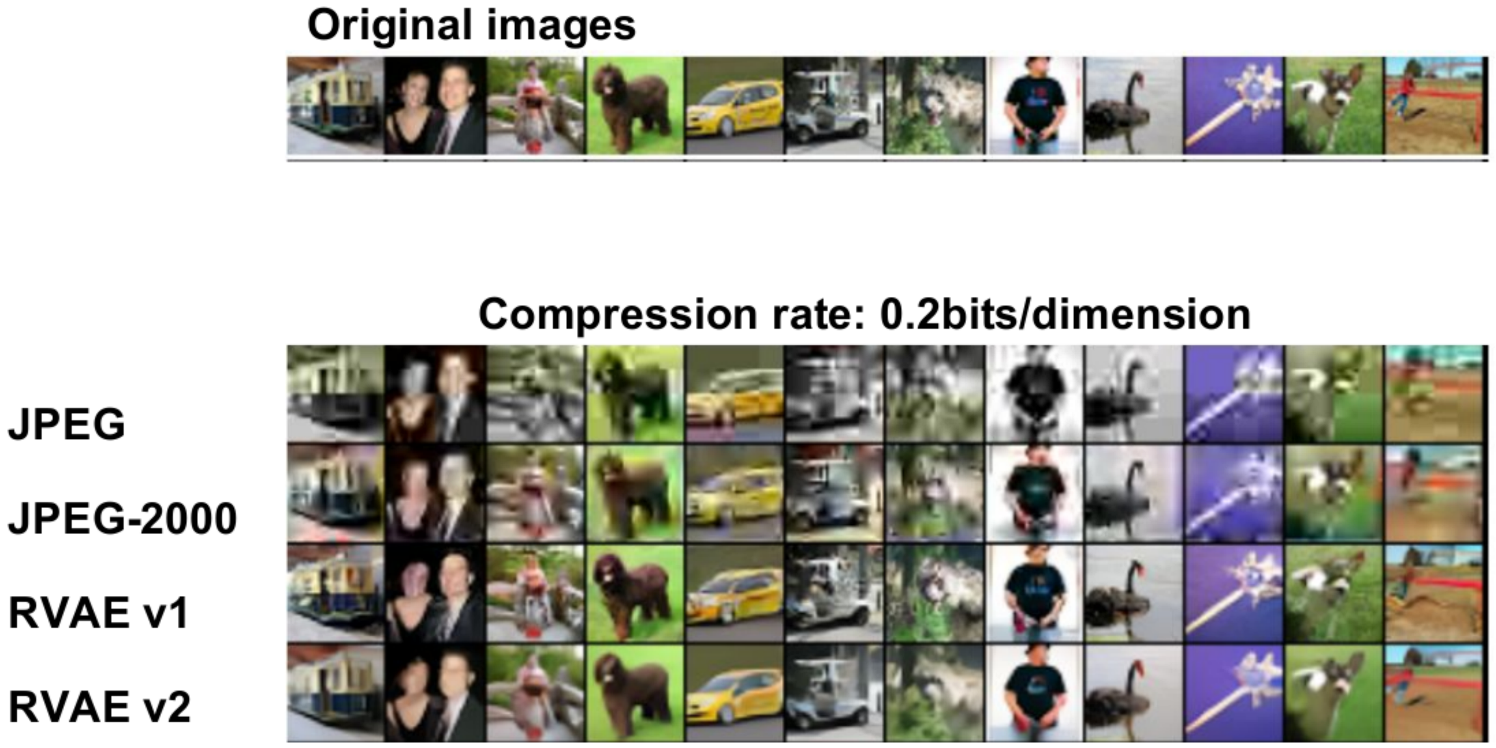

(b) Learned MNIST manifold

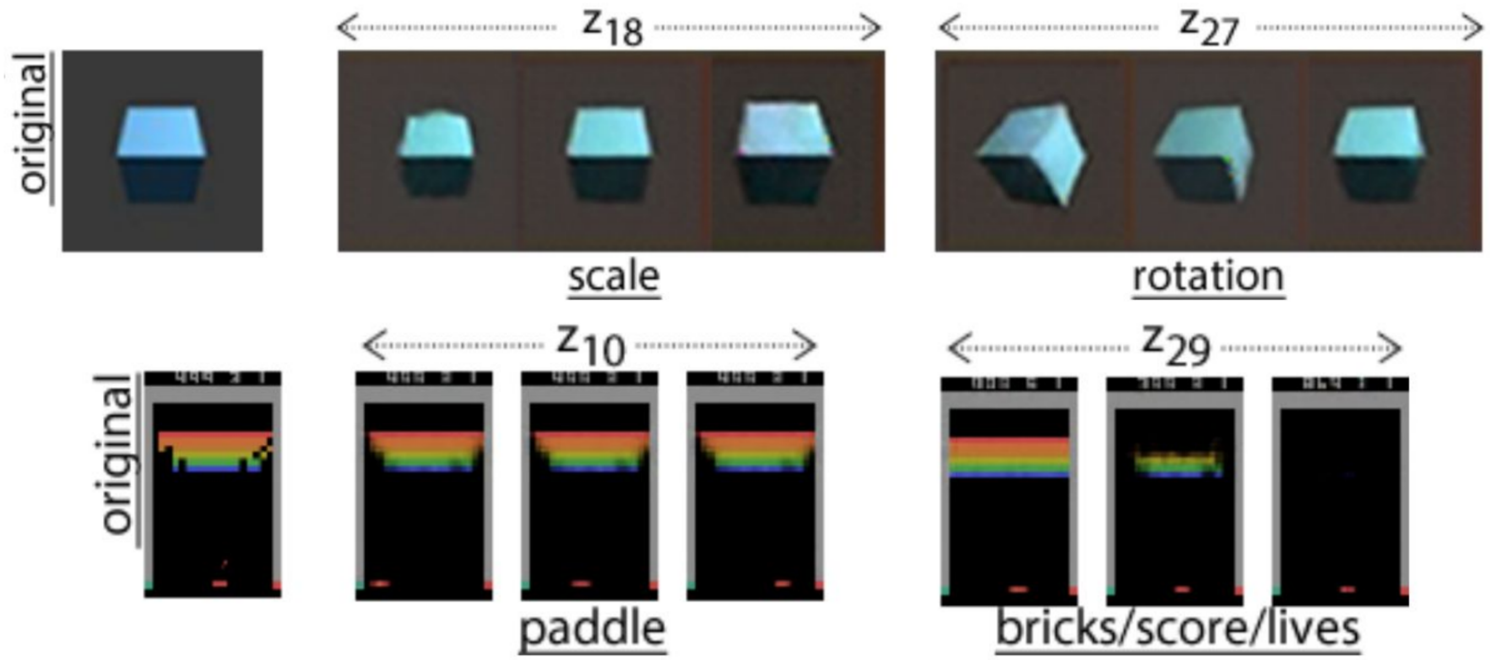
Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

(Kingma and Welling, 2013)

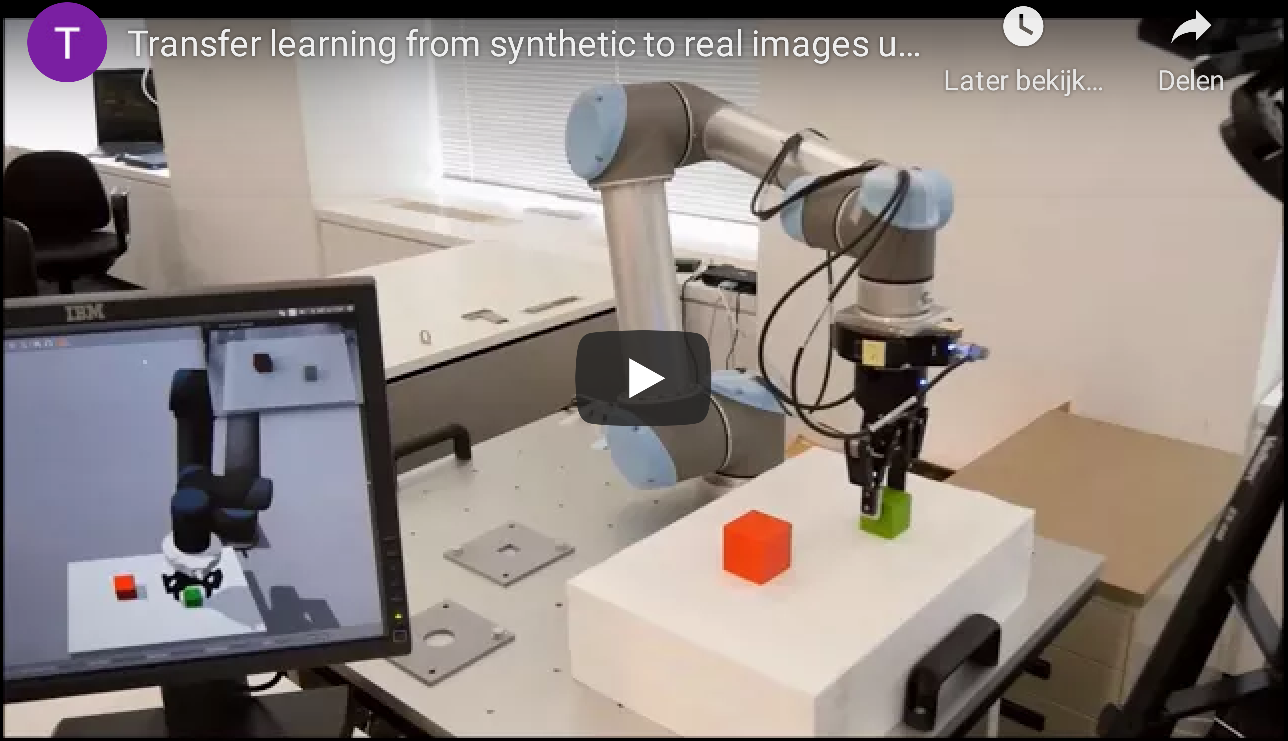
Random walks in latent space (Vahdat and Kautz, 2020).



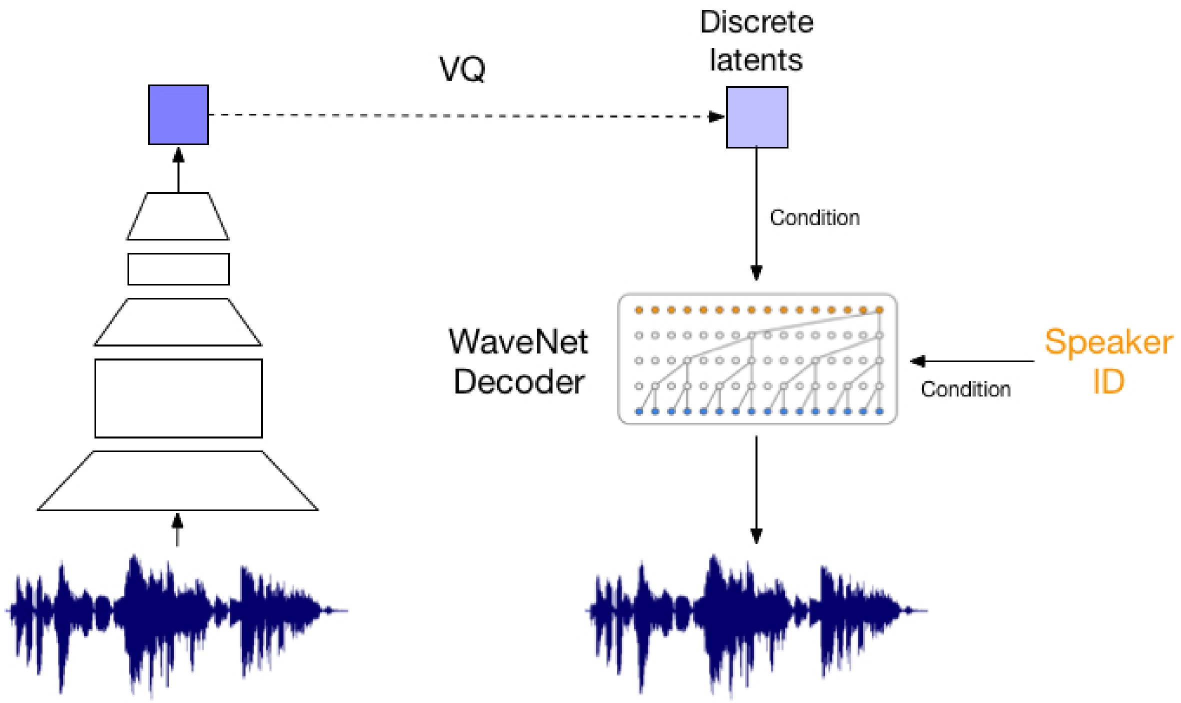
Hierarchical **compression of images and other data**, e.g., in video conferencing systems (Gregor et al, 2016).



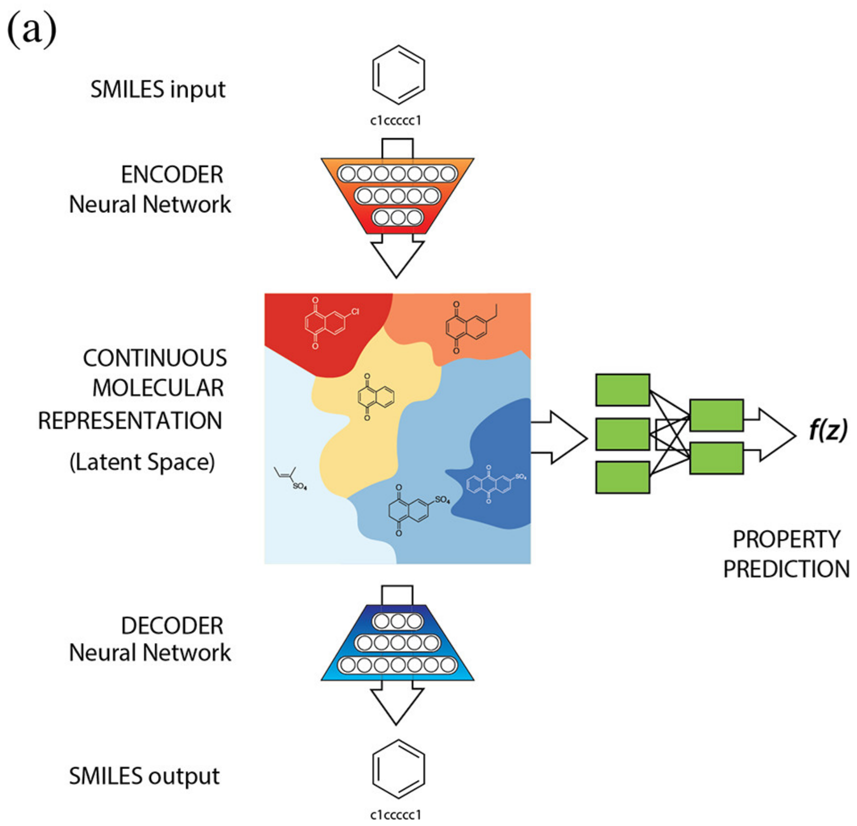
Understanding the factors of variation and invariances (Higgins et al, 2017).

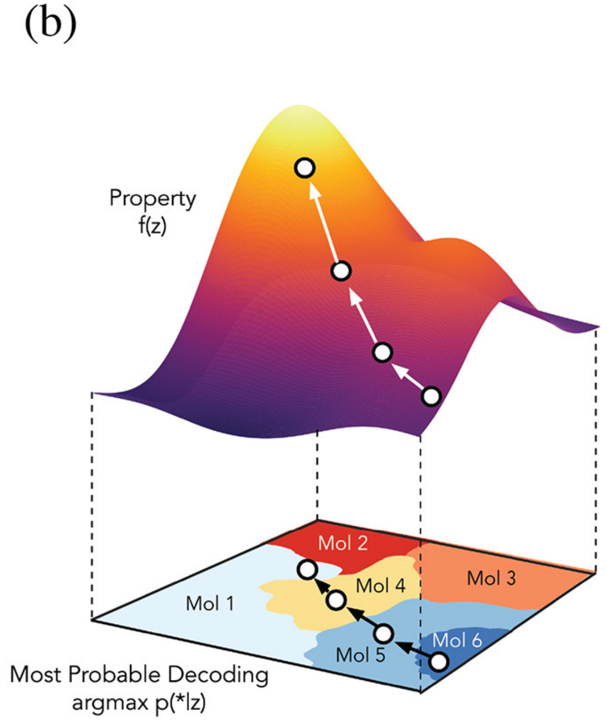


Bridging the **simulation-to-reality** gap (Inoue et al, 2017).



Voice style transfer [demo] (van den Oord et al, 2017).

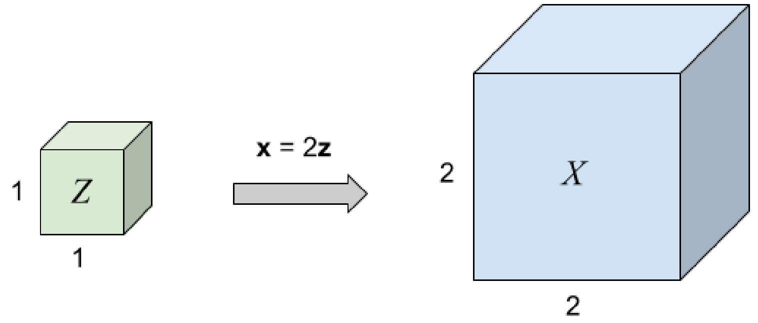




Design of new molecules with desired chemical properties (Gomez-Bombarelli et al, 2016).

Part III: Normalizing flows

Change of variables theorem

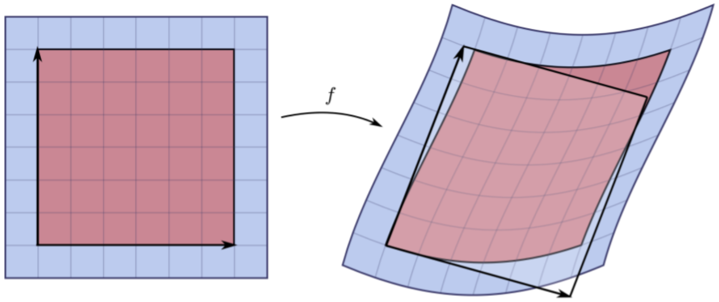


Assume $p(\mathbf{z})$ is a uniformly distributed unit cube in $\mathbb{R}^3$ and $\mathbf{x} = f(\mathbf{z}) = 2\mathbf{z}$.

Since the total probability mass must be conserved,

$$p(\mathbf{x} = f(\mathbf{z})) = p(\mathbf{z})\frac{V_{\mathbf{z}}}{V_{\mathbf{x}}} = p(\mathbf{z})\frac{1}{8},$$

where $\frac{1}{8} = \left|\det\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}\right|^{-1}$ represents the determinant of the linear transformation f.



What if f is non-linear?

- The Jacobian $J_f(\mathbf{z})$ of $\mathbf{x} = f(\mathbf{z})$ represents the infinitesimal linear transformation in the neighborhood of $\mathbf{z}$
- If the function is a bijective map, then the mass must be conserved locally.

Therefore, the local change of density yields

$$p(\mathbf{x} = f(\mathbf{z})) = p(\mathbf{z}) \left|\det J_f(\mathbf{z})\right|^{-1}.$$

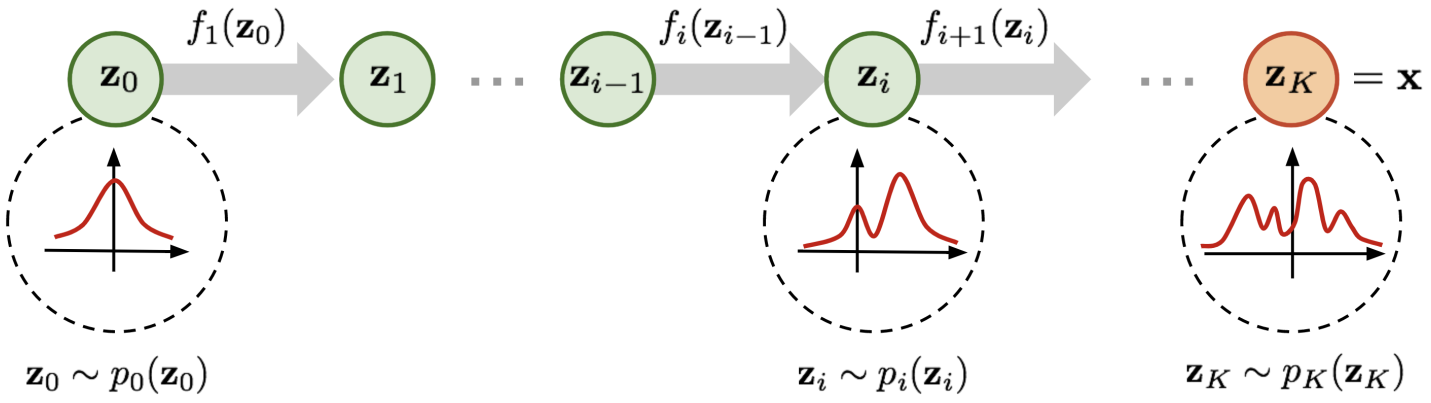
Similarly, for $g = f^{-1}$, we have

$$p(\mathbf{x}) = p(\mathbf{z} = g(\mathbf{x})) \left|\det J_g(\mathbf{x})\right|.$$

Normalizing flows

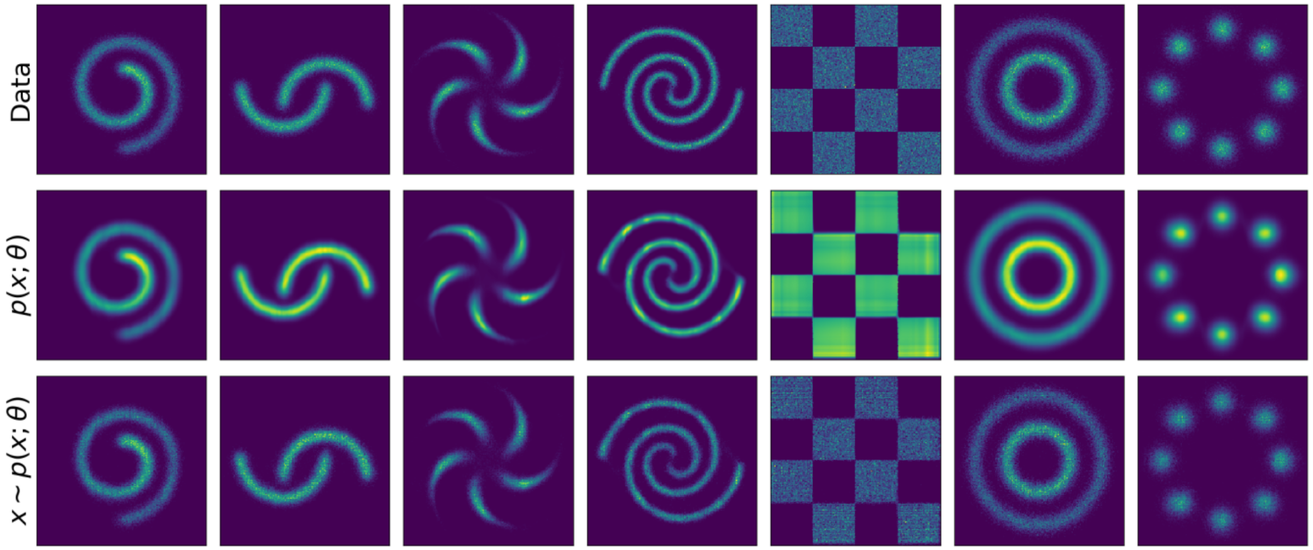
A normalizing flow is a change of variable f parameterized by an invertible neural network that transforms a base distribution $p(\mathbf{z})$ into $p(\mathbf{x})$. Formally,

- f is a composition $f = f_K \circ ... \circ f_1$, where each f_k is an invertible neural transformation
- $g_k = f_k^{-1}$
- $\mathbf{z}_k = f_k(\mathbf{z}_{k-1})$, with $\mathbf{z}_0 = \mathbf{z}$ and $\mathbf{z}_K = \mathbf{x}$
- $p(\mathbf{z}_k) = p(\mathbf{z}_{k-1} = g_k(\mathbf{z}_k)) \left|\det J_{g_k}(\mathbf{z}_k)\right|$



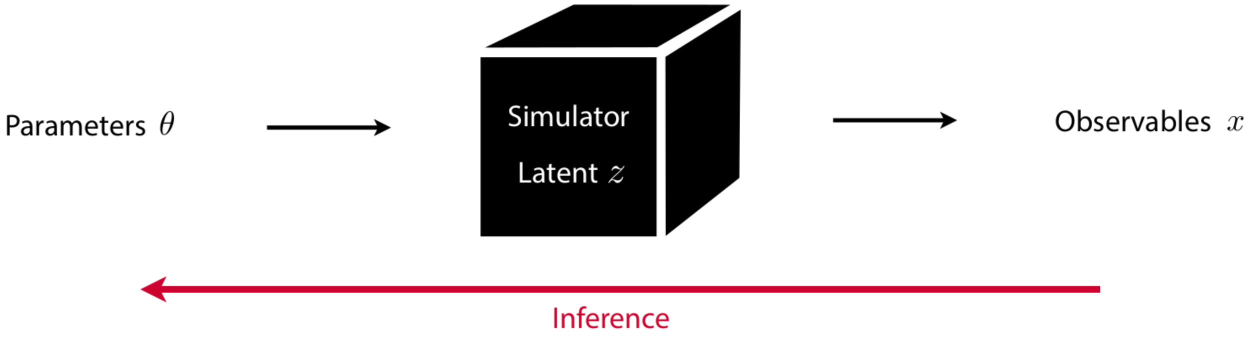
—
Image credits: Lilian Weng, 2018.

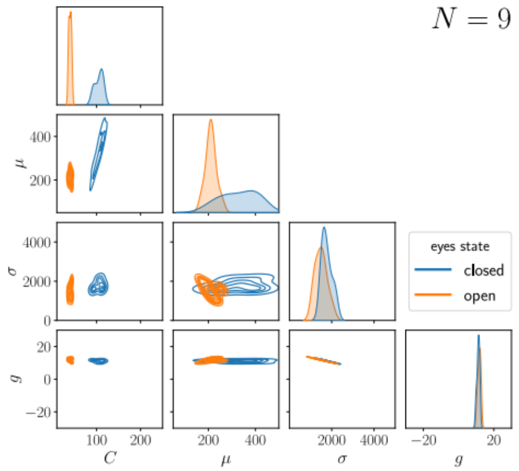
Examples

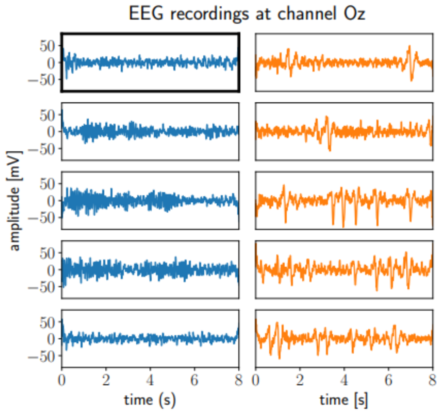


Normalizing flows shine in applications for **density estimation**, where access to the density function is required (Wehenkel and Louppe, 2019).

Normalizing flows are at the core of neural posterior estimation algorithms for **simulation-based inference**.







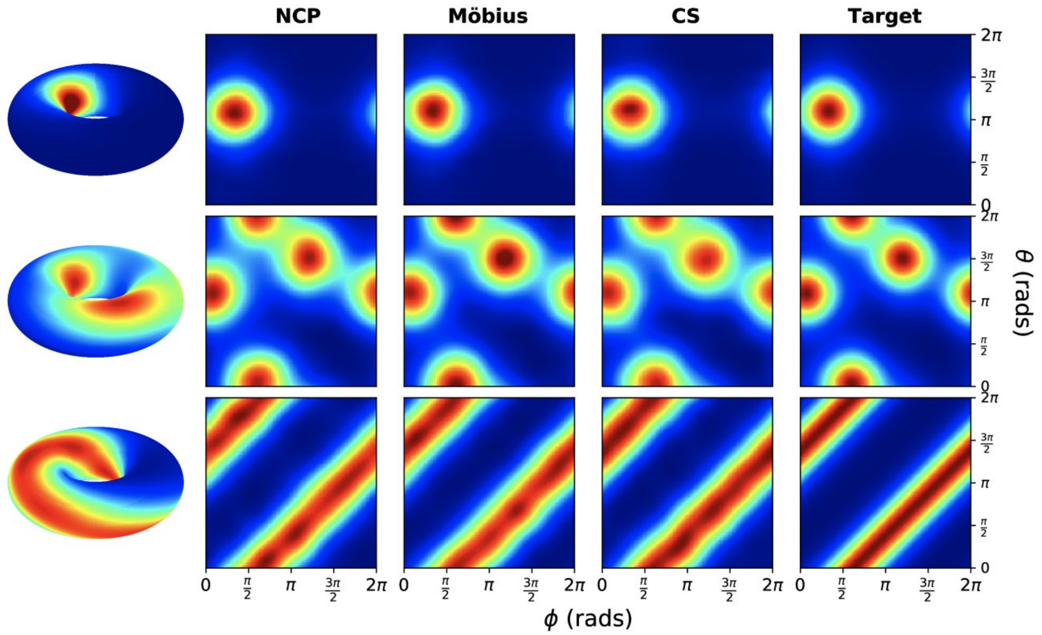


Figure 3. Learned densities on $\mathbb{T}^2$ using NCP, Möbius and CS flows. Densities shown on the torus are from NCP.

Density estimation on **manifolds** (Rezende et al, 2020).

The end.