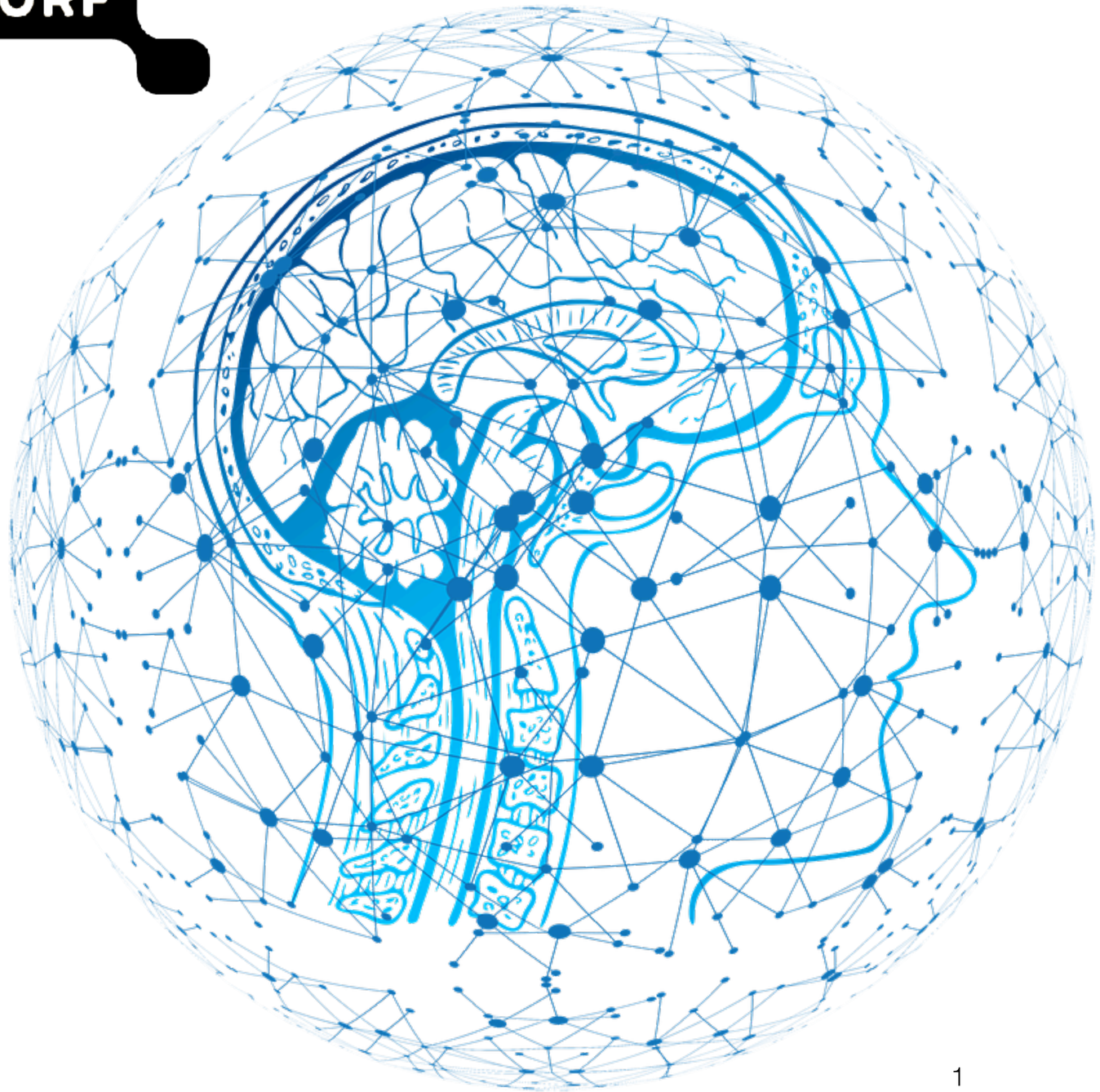
SURF



Introduction to Deep Learning

Variational Autoencoder

Maxwell Cai

maxwellcai.com

Move beyond associating inputs and outputs

Classification: mapping a distribution to a **certain** output

Happy cat

Angry cat

Relaxed cat

“Cats”

Generation: mapping an input to a probability **distribution**

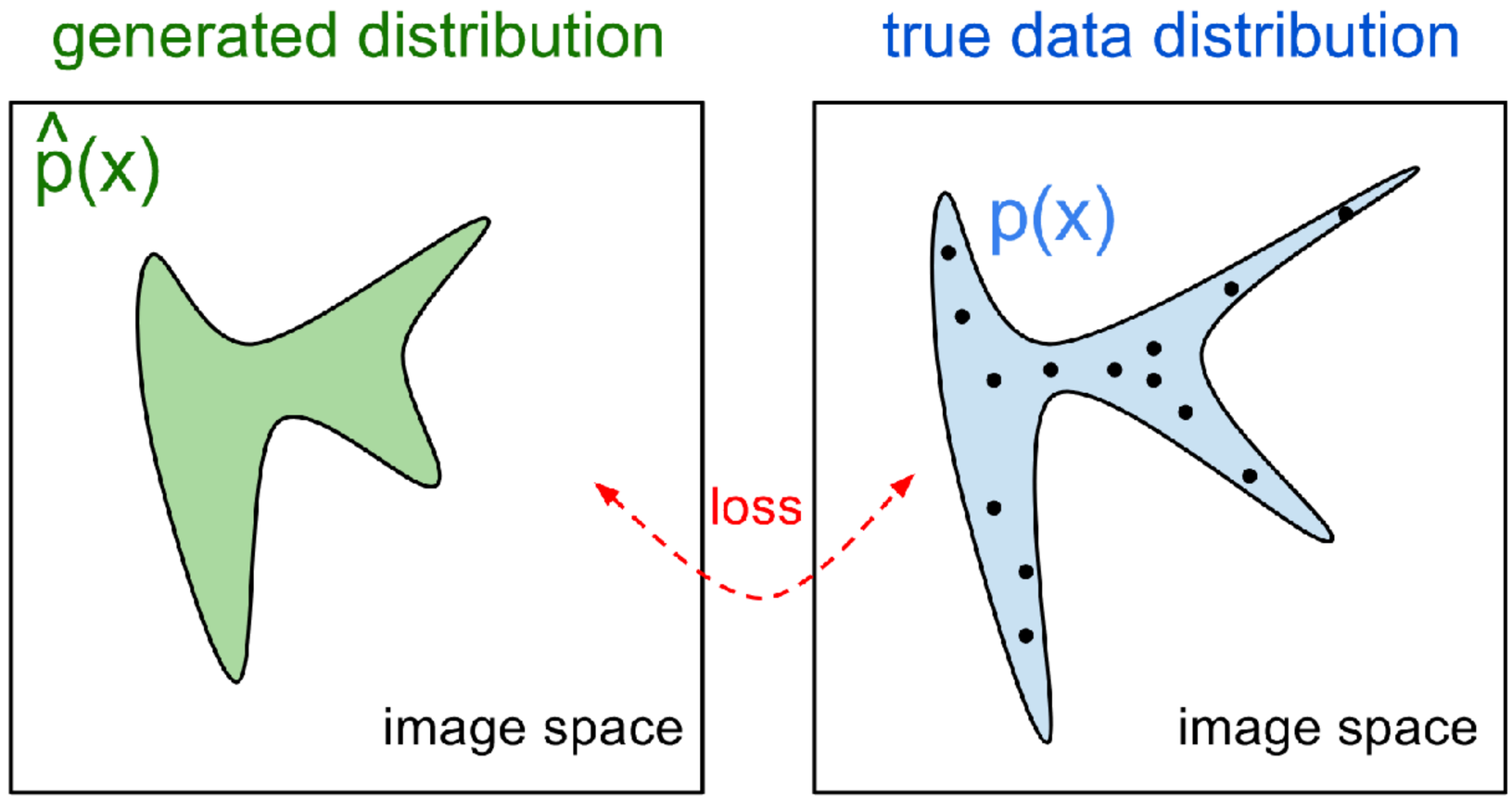
“Cats”

Happy cat

Angry cat

Relaxed cat

What is a Generative Model?

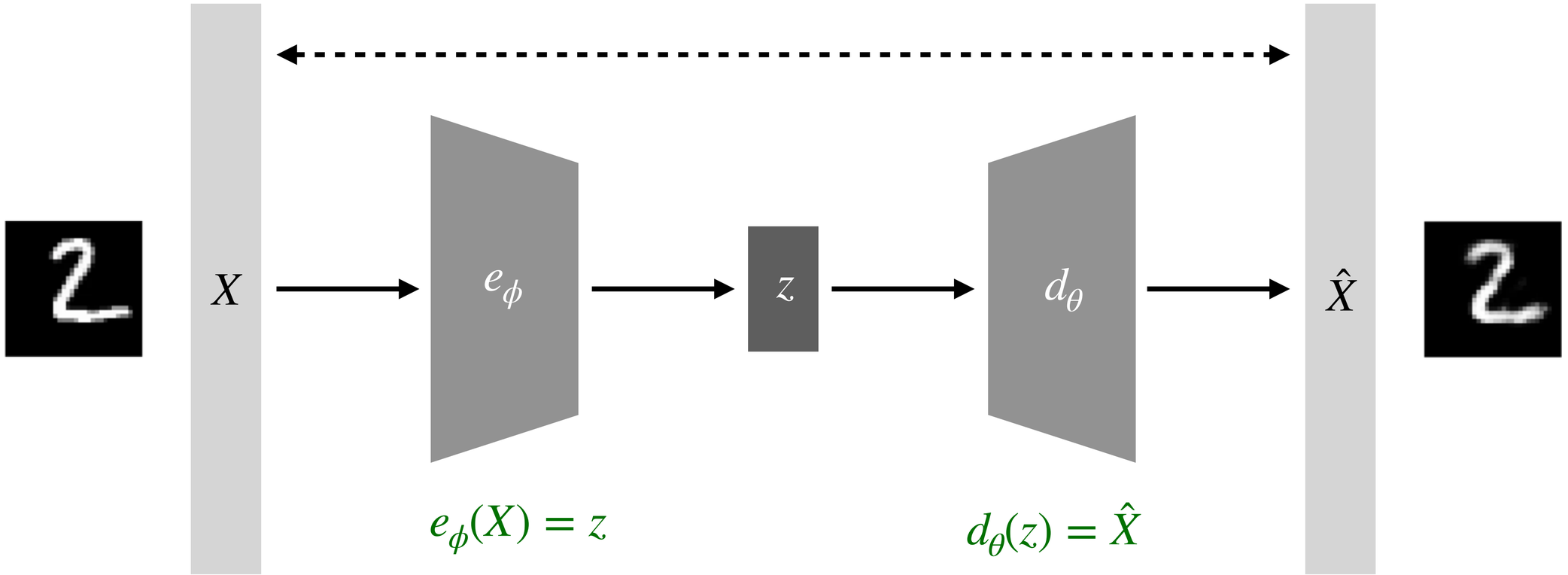


The underlying process of $p(x)$ is unknown (and usually intractable).

We approximate $p(x)$ with $\hat{p}_\theta(x)$.

Image: OpenAI

Autoencoder

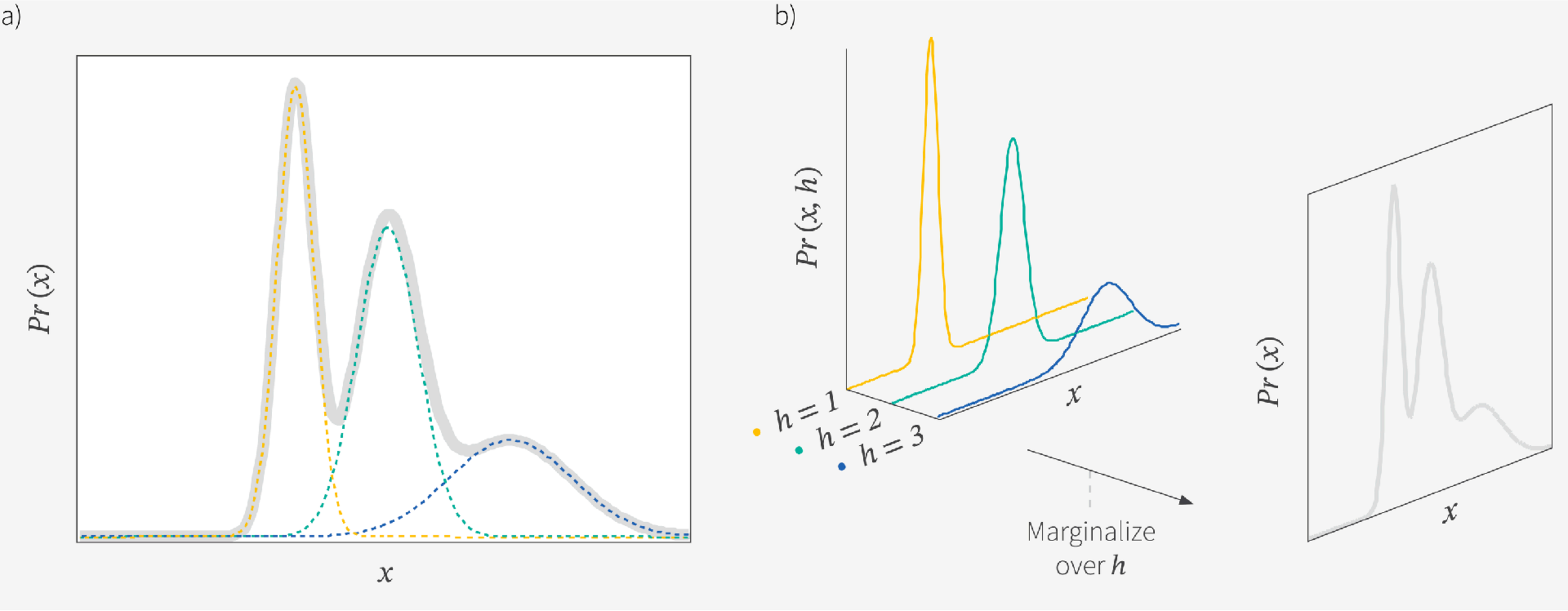


It looks like an autoencoder trivially copies X to $\hat{X}$, **but**:

- The **latent vector** z has lower dimension than the input X, and therefore information are **compressed**.
- A good autoencoder selects relevant information and **discards irrelevant information**.

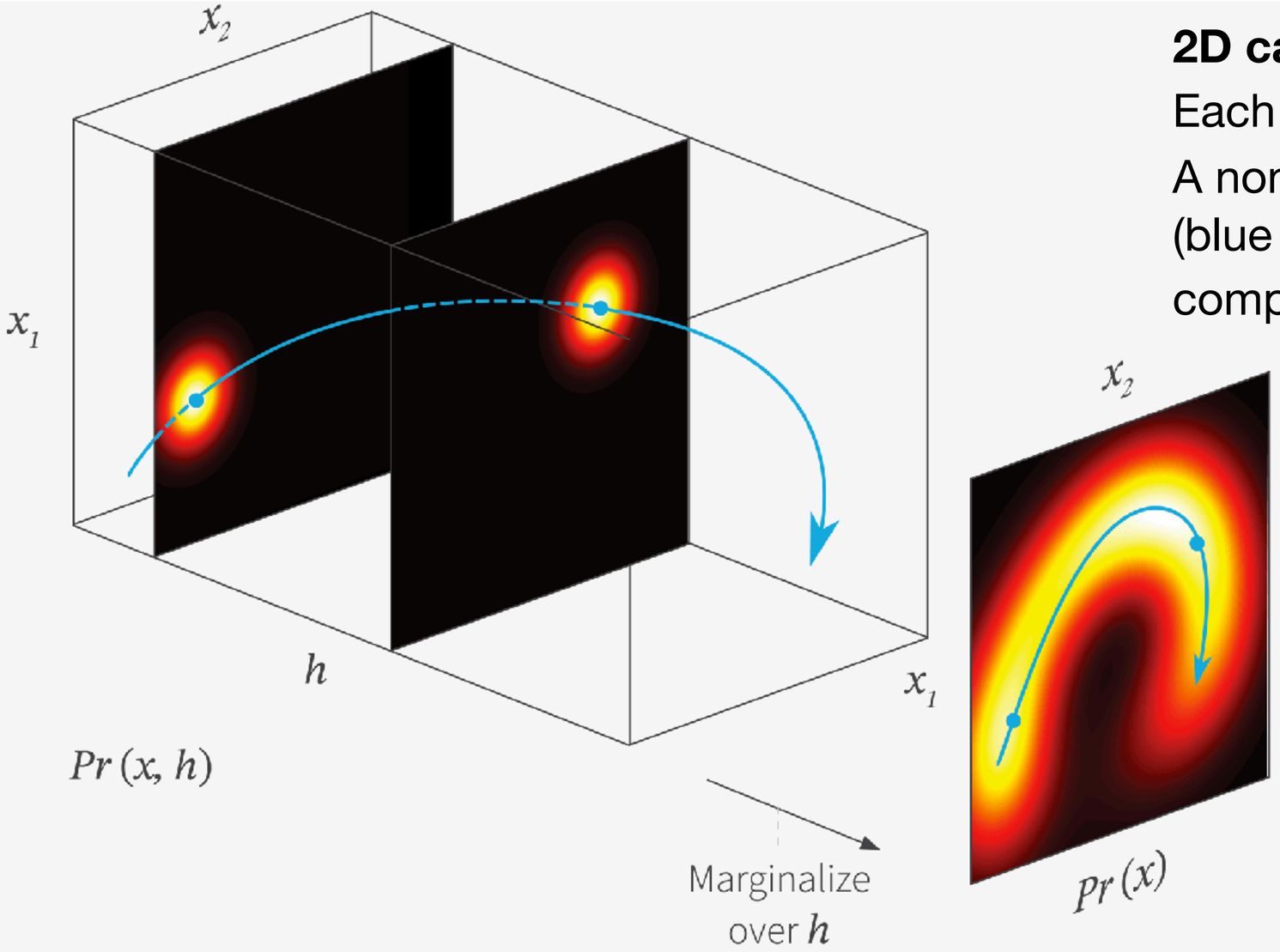
Hinton & Salakhutdinov (2006, Science)

Express complex distribution by marginalising simple distributions



$$p(x) = \sum_{i=1}^{k} p(\mathbf{x}, h = i) \quad \Rightarrow \quad p(x) = \int p(\mathbf{x}, \mathbf{h}) d\mathbf{h}$$

Credit: Borealis AI



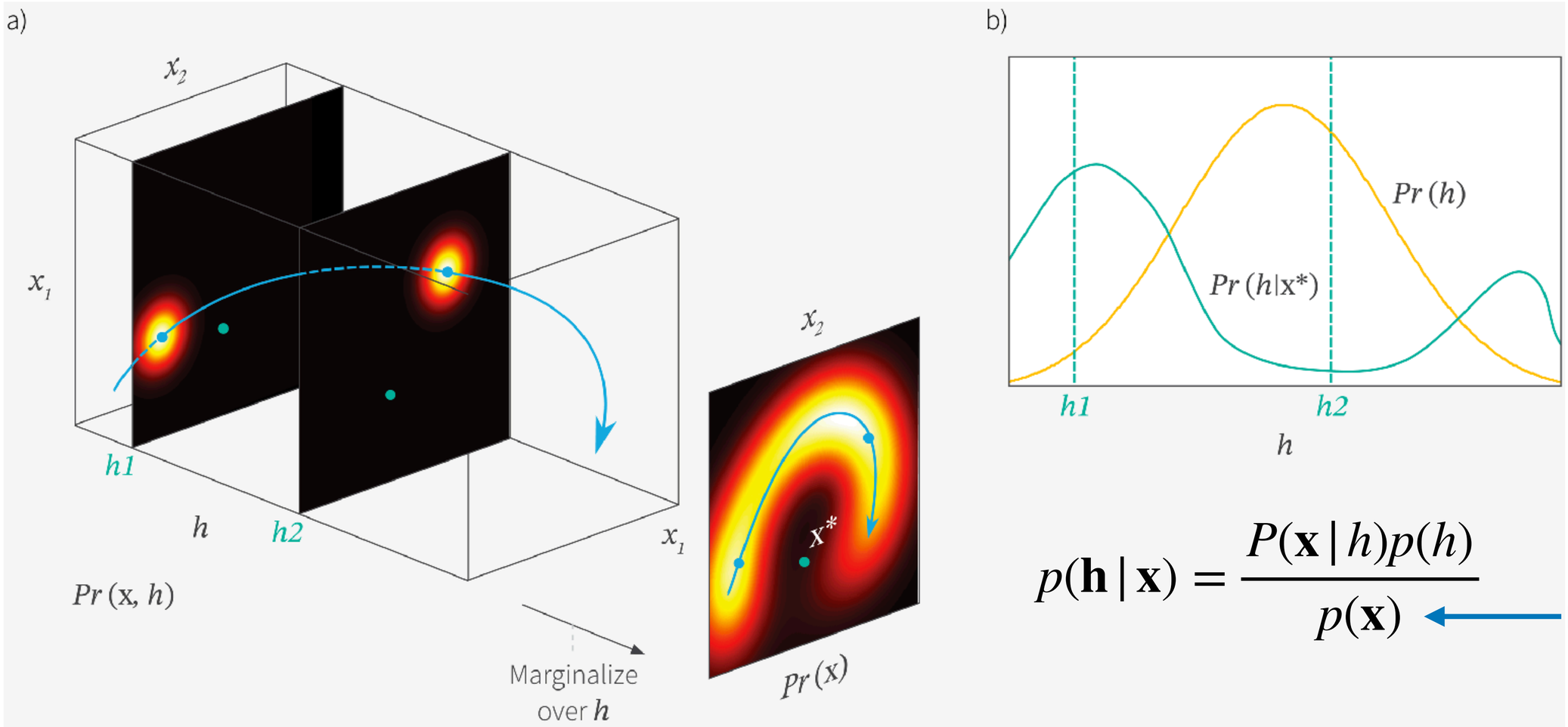
2D case:

Each value of h is a spherical gaussian

A non-linear combination described by $f(h, \theta)$ (blue curve) allows us to approximate a complex distribution $p(x)$.

Image credit: Borealis AI

Computing the posterior



$$p(\mathbf{h} \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid h)p(h)}{p(\mathbf{x})}$$ ← Intractable

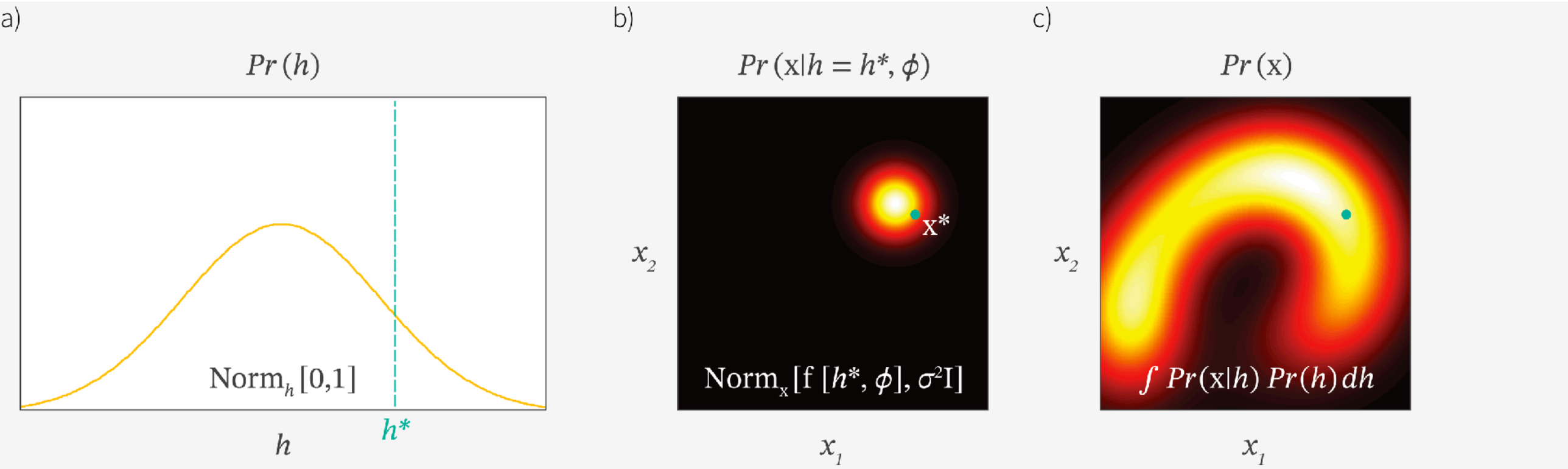
$p(\mathbf{x} \mid \mathbf{h}, \theta)$ tells us how to compute the observed distribution given hidden variable $\mathbf{h}$.

However, we are also interested in what $\mathbf{h}$ is responsible for $\mathbf{x}$.

Therefore, we are interested in $p(\mathbf{h} \mid \mathbf{x})$.

Image credit: Borealis AI

Generation



Draw $\mathbf{h}^*$ from the prior $p(\mathbf{h})$

Plug $\mathbf{h}^*$ into $p(\mathbf{x} \mid \mathbf{h}, \phi)$

Repeat many times until an approximation of $p(\mathbf{x})$ is properly constructed

Image credit: Borealis AI

Evaluating the likelihood

The likelihood is:

$$
\begin{aligned}
p(\mathbf{x}) &= \int p(\mathbf{x}, \mathbf{h} \,|\, \phi) d\mathbf{h} \\
&= \int p(\mathbf{x} \,|\, \mathbf{h}, \phi) p(\mathbf{h}) d\mathbf{h} \\
&= \int \mathcal{N}\left[f(\mathbf{h}, \phi), \sigma^2 \mathbf{I}\right] \mathcal{N}(\mathbf{0}, \mathbf{1}) d\mathbf{h}
\end{aligned}
$$

More frequently the **log likelihood** is used.

Problem is: There is no close form for this integral.

Since we can't directly compute $p(x)$, can we at least **constrain its range?**

Jensen's Inequality

For a concave function $g[\bullet]$, Jensen's inequality states that

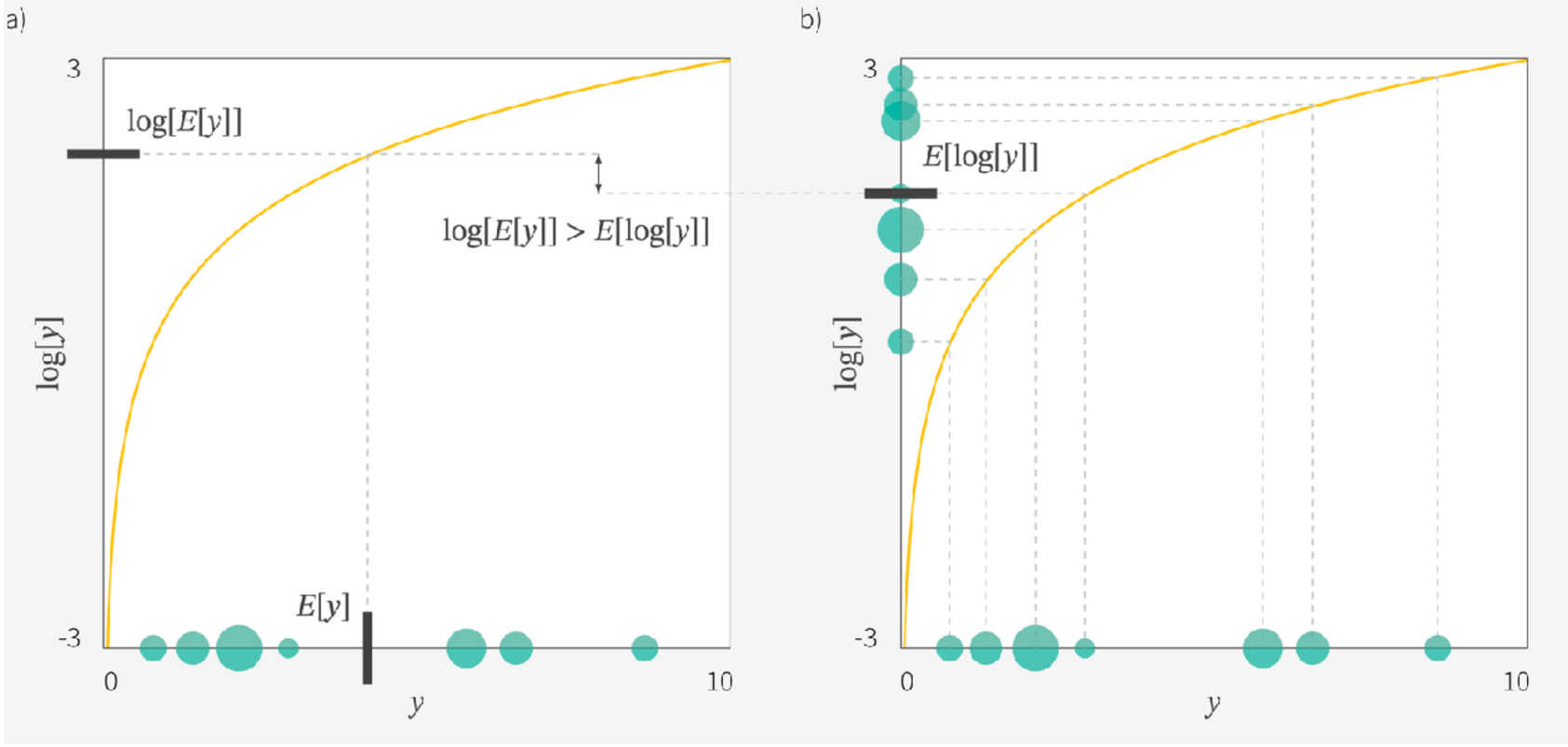
$$g\left[\mathbf{E}(y)\right] \geq \mathbf{E}\left[g(y)\right]$$

Since $\log$ is a concave function, so

$$\log\left[\mathbf{E}(y)\right] \geq \mathbf{E}\left[\log(y)\right]$$

Which means

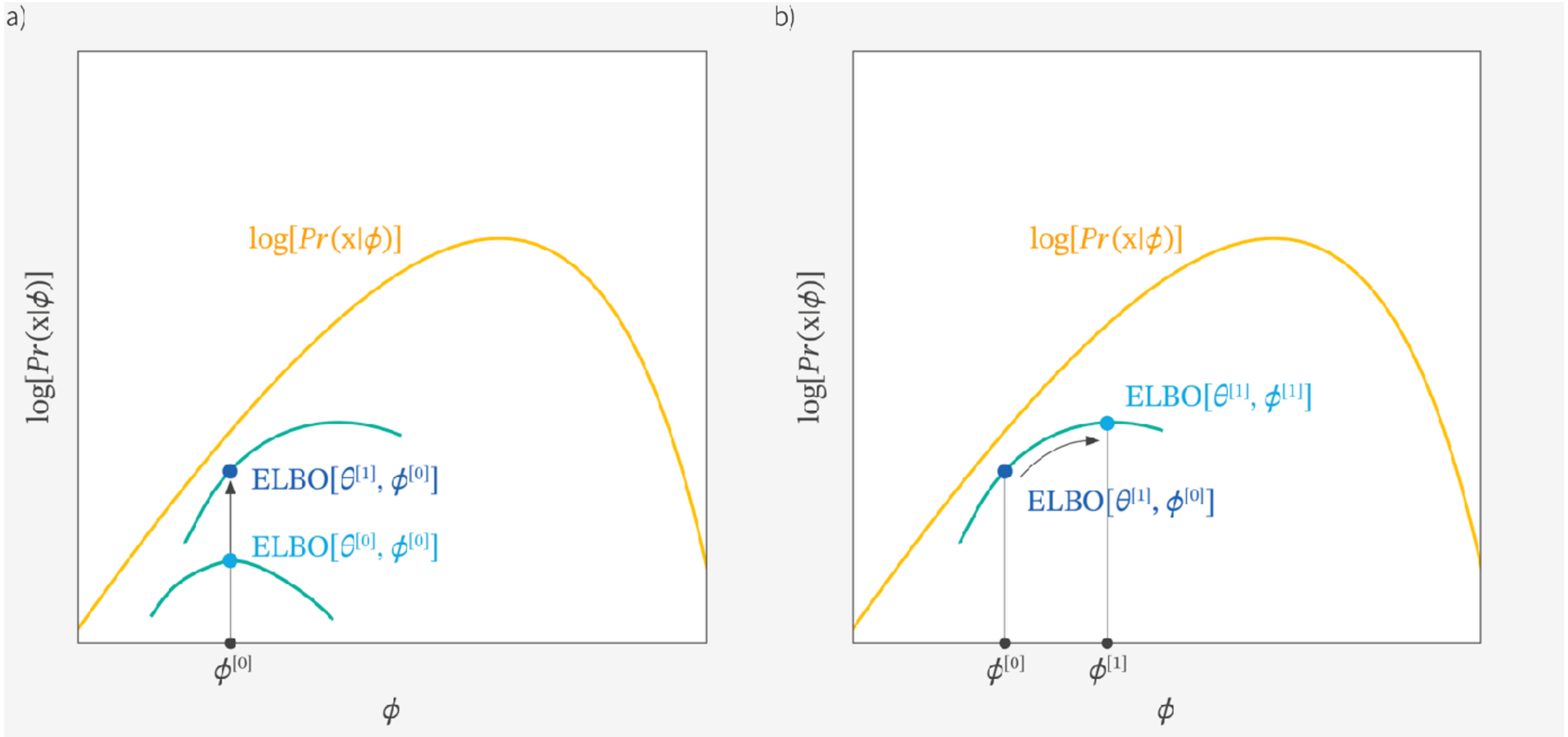
$$\log\left[\int p(y)dy\right] \leq \int p(y)\log(y)dy$$



Evidence Lower Bound (ELBO)

If $q_\theta(\mathbf{h})$ is an arbitrary PDF depending on parameters θ,

$$\log\left[p_\phi(x)\right] = \log\left[\int p_\phi(\mathbf{x},\mathbf{h})d\mathbf{h}\right] = \log\left[\int q_\theta(\mathbf{h})\frac{p_\phi(\mathbf{x},\mathbf{h})}{q(\mathbf{h})}d\mathbf{h}\right] \geq \underbrace{\int q_\theta(\mathbf{h})\log\left[\frac{p_\phi(\mathbf{x},\mathbf{h})}{q_\theta(\mathbf{h})}d\mathbf{h}\right]}_{\text{ELBO}}$$



By adjusting the ELBO parameters θ, it is possible to find a reasonable approximation to the original PDF.

Image credit: Borealis AI

Evidence Lower Bound (ELBO)

$$
\begin{aligned}
\mathrm{ELBO}[\theta, \phi] &= \int q_\theta(\mathbf{x}\,|\,\mathbf{h}) \log \left[\frac{p_\phi(\mathbf{x}, \mathbf{h})}{q_\theta(\mathbf{x}\,|\,\mathbf{h})} \right] d\mathbf{h} \\
&= \int q_\theta(\mathbf{x}\,|\,\mathbf{h}) \log \left[\frac{p_\phi(\mathbf{h}\,|\,\mathbf{x}) p_\phi(\mathbf{x})}{q_\theta(\mathbf{x}\,|\,\mathbf{h})} \right] d\mathbf{h} \\
&= \int q_\theta(\mathbf{x}\,|\,\mathbf{h}) \log \Big[p_\phi(\mathbf{x}) \Big] d\mathbf{h} + \int q_\theta(\mathbf{x}\,|\,\mathbf{h}) \log \left[\frac{p_\phi(\mathbf{h}\,|\,\mathbf{x})}{q_\theta(\mathbf{x}\,|\,\mathbf{h})} \right] d\mathbf{h} \\
&= \log[p_\phi(\mathbf{x})] + \int q_\theta(\mathbf{x}\,|\,\mathbf{h}) \log \left[\frac{p_\phi(\mathbf{h}\,|\,\mathbf{x})}{q_\theta(\mathbf{x}\,|\,\mathbf{h})} \right] d\mathbf{h} \\
&= \log[p_\phi(\mathbf{x})] - \mathbf{D}_{\mathrm{KL}} \Big[q_\theta(\mathbf{x}\,|\,\mathbf{h}) \,||\, p_\phi(\mathbf{h}\,|\,\mathbf{x}) \Big]
\end{aligned}
$$

Reconstruction loss

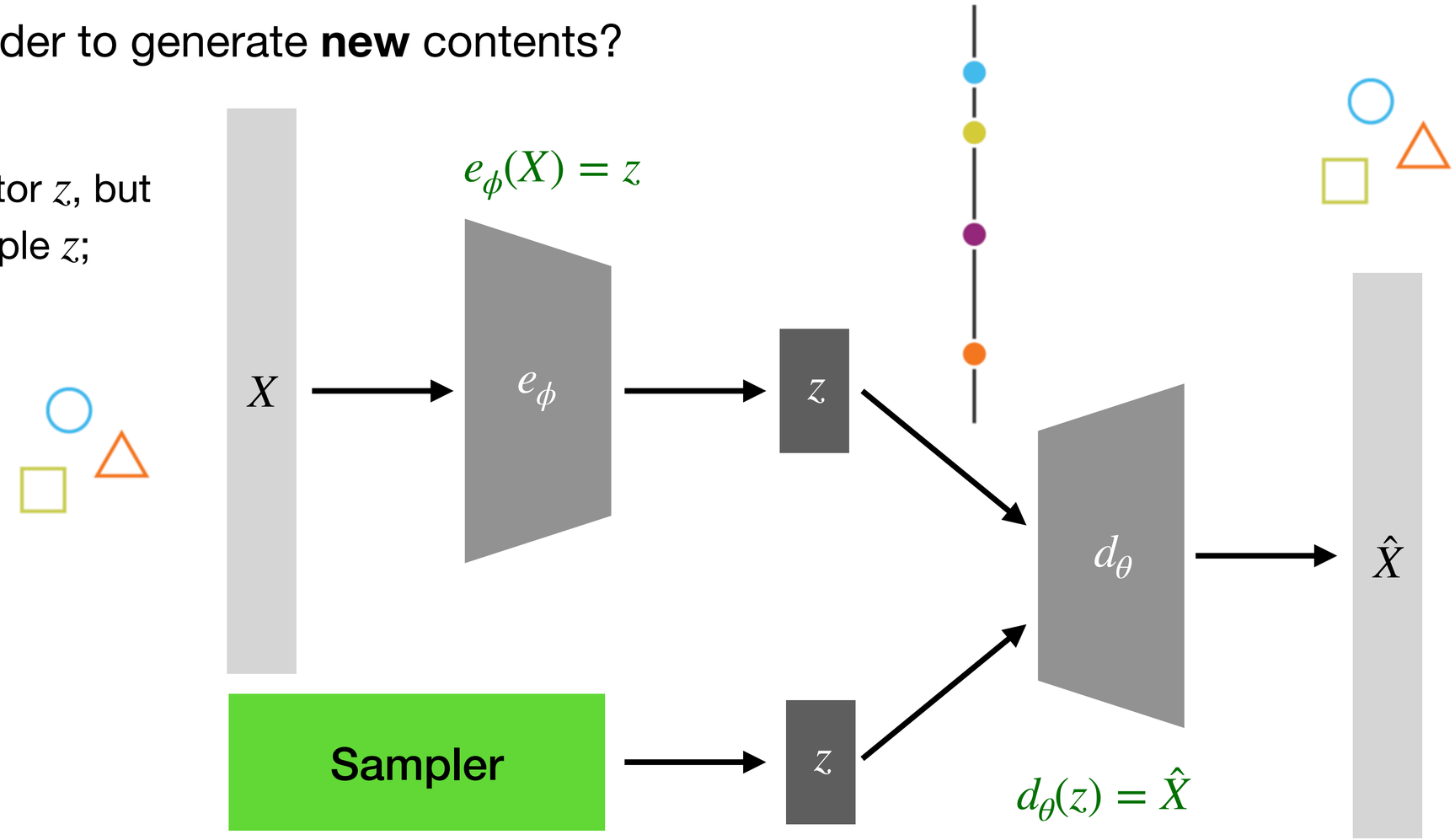
Tightness of the bound (i.e., ELBO gap)

Autoencoder

How can we let an autoencoder to generate **new** contents?

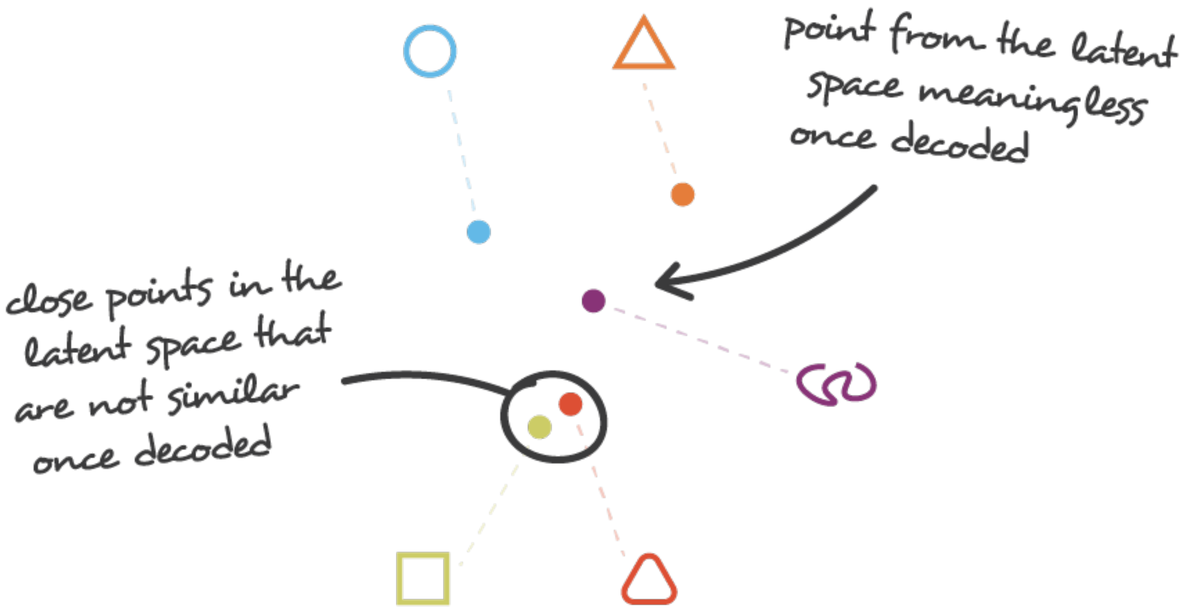
We can sample the latent vector z, but

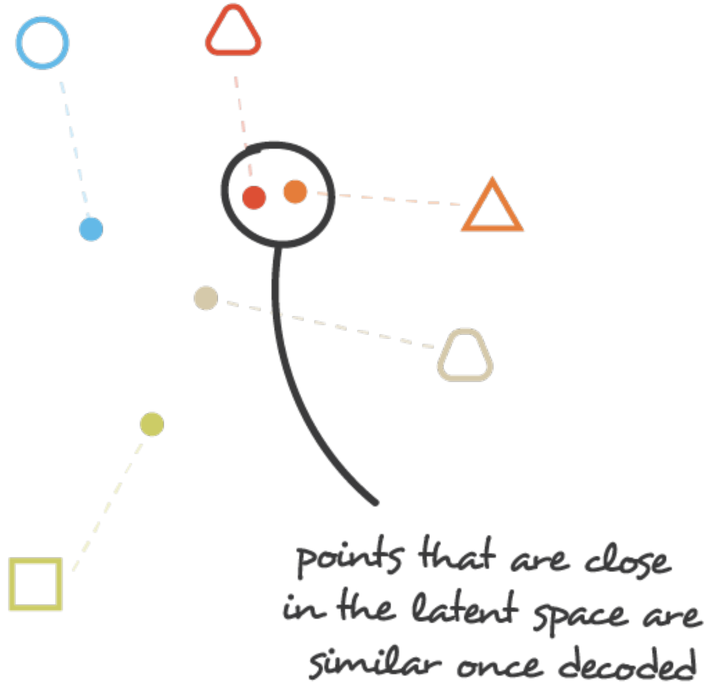
- We don't know how to sample z;
- z can be very irregular.



We need to **regularize** the distribution (i.e., arrangement) of the latent space!

Latent Space Distribution





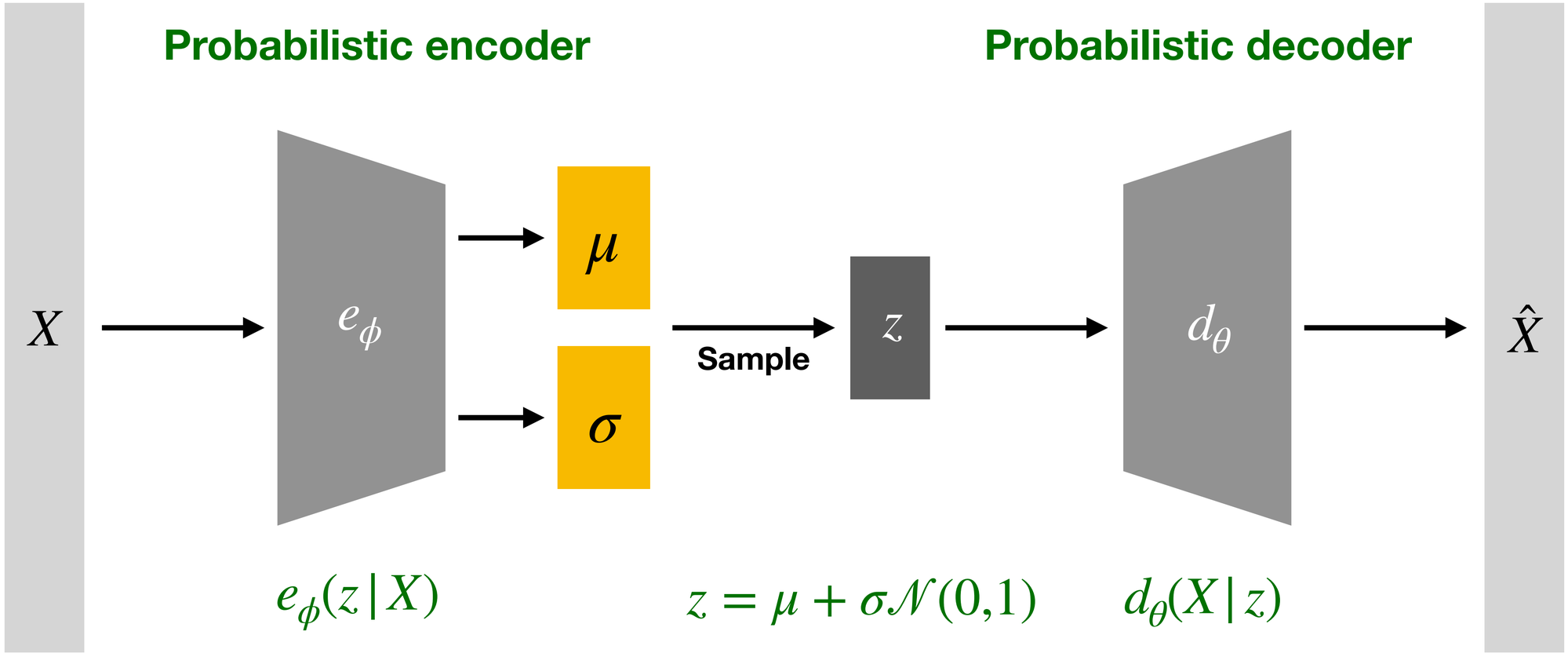
irregular latent space

regular latent space

A regular latent space allows us to do **interpolation**.
New contents are generated **by interpolating** the latent space.

Image: Joseph Rocca (TowardDataScience)

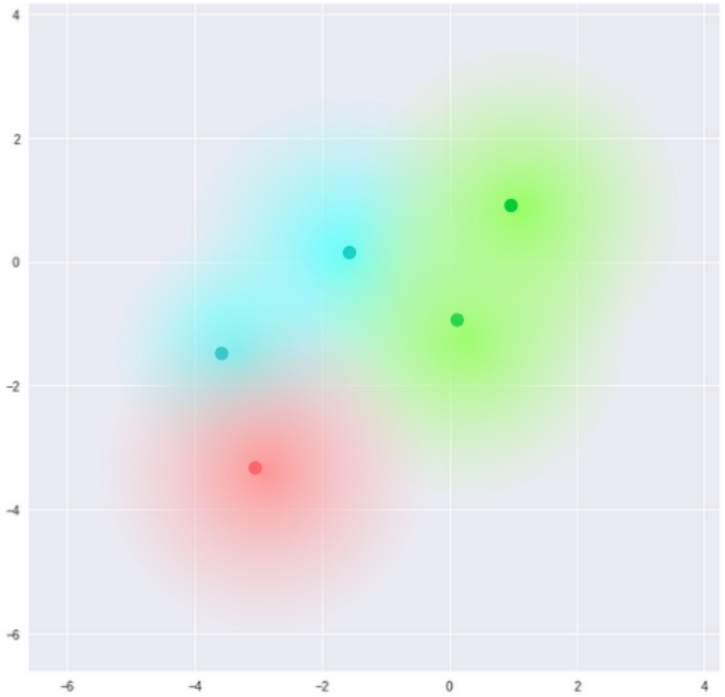
Variational Autoencoder



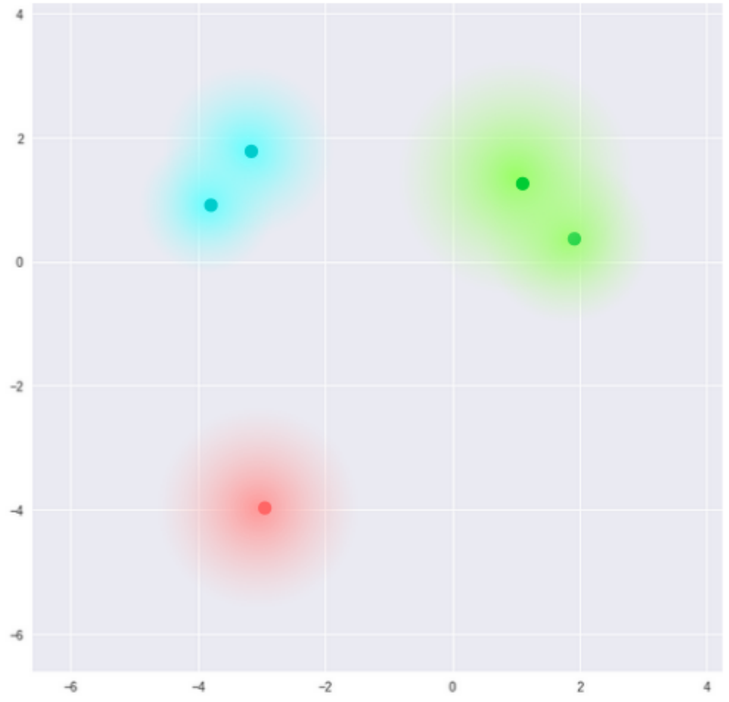
Kingma & Welling (2013, arXiv:1312.6114)

Latent space: tight, smooth, and complete

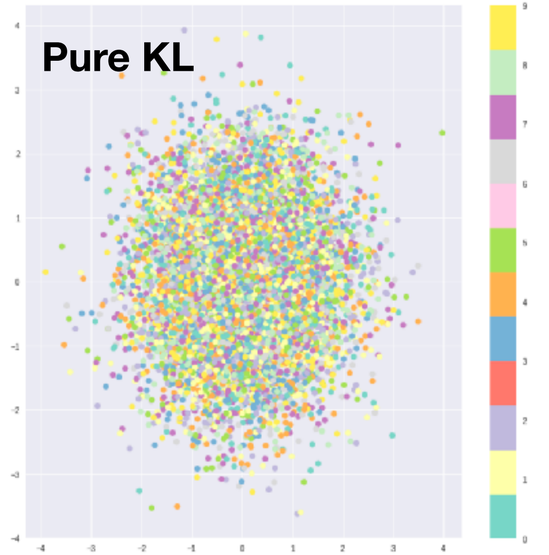
Without constraining μ and σ:

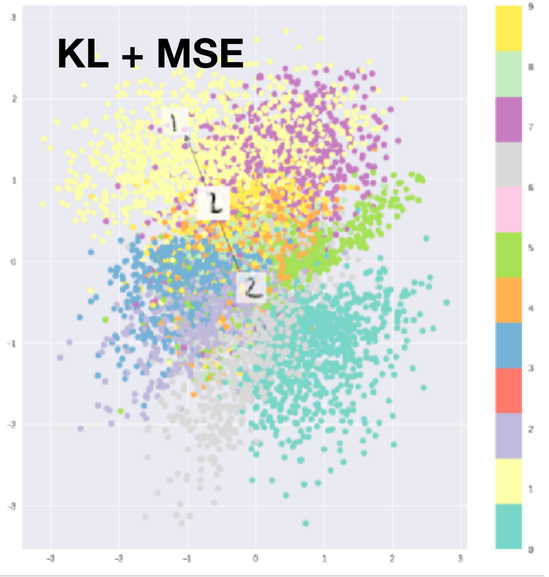


What we require



What we may inadvertently end up with

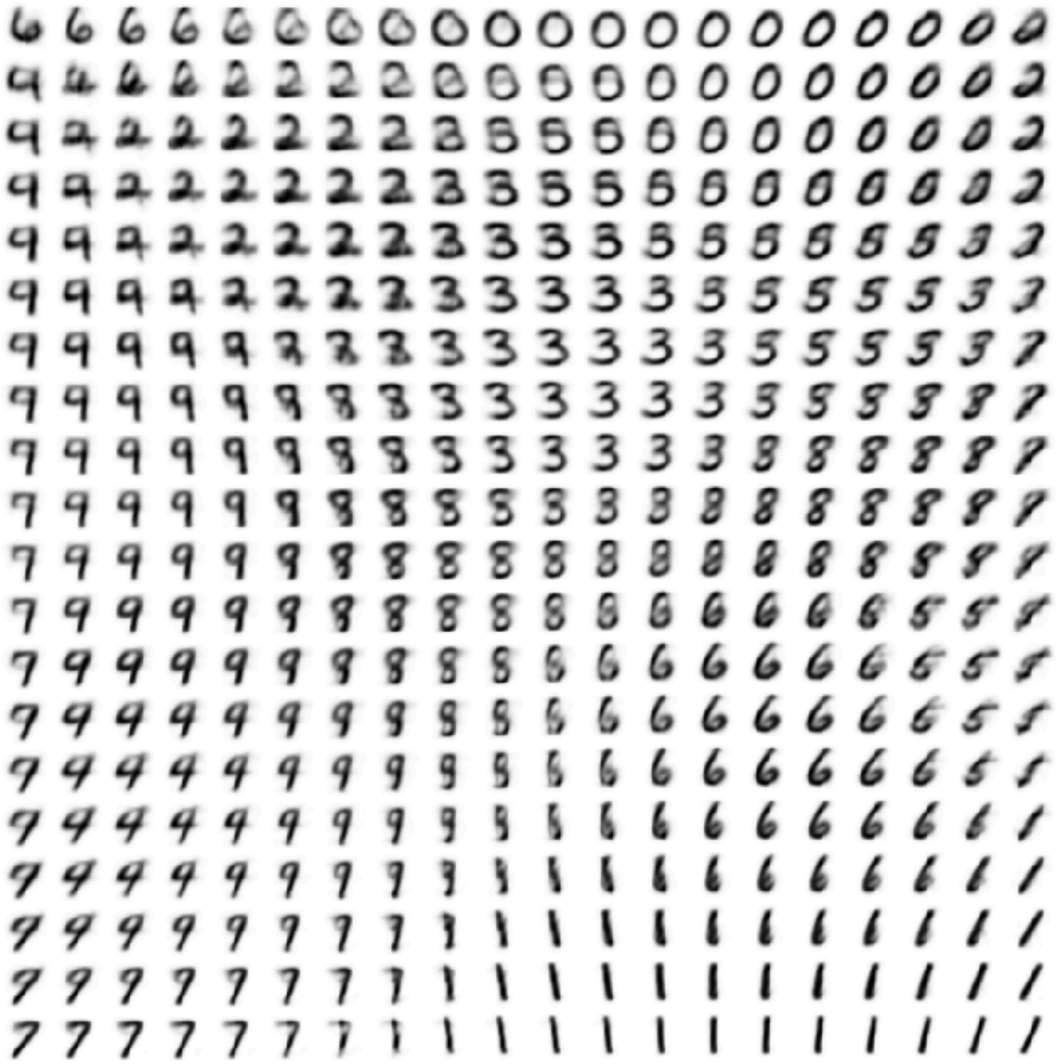


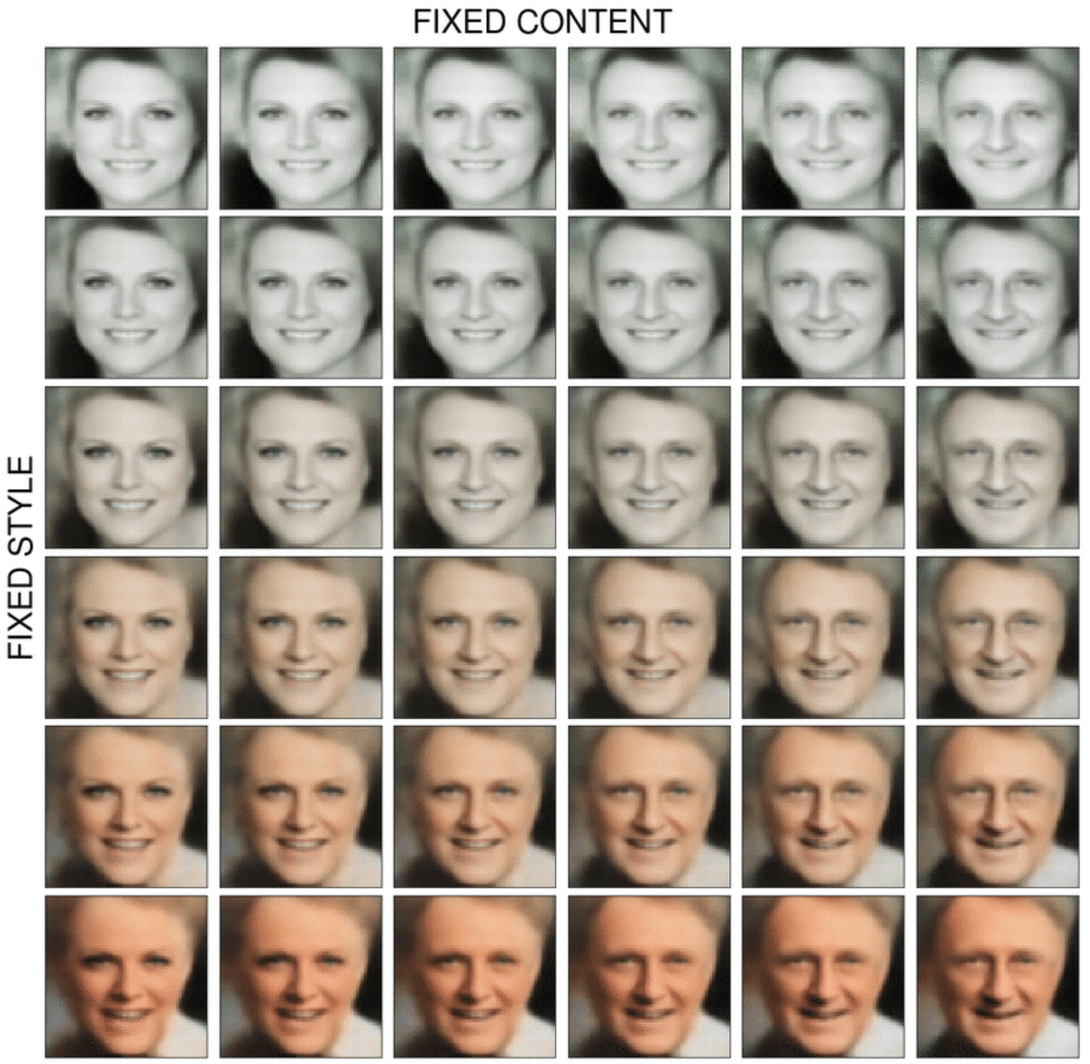


Solution:

- Use **KL divergence** to force clusters to get **close to each other**;
- Use **reconstruction losses** (e.g., MSE) to **avoid** clusters from **overlapping**.

Latent vector interpolation





Bouchacourt, Tomioka & Nowozin (2017).

Application: VAE for Drug Discovery

Proposing candidate molecules by interpolating the latent space.

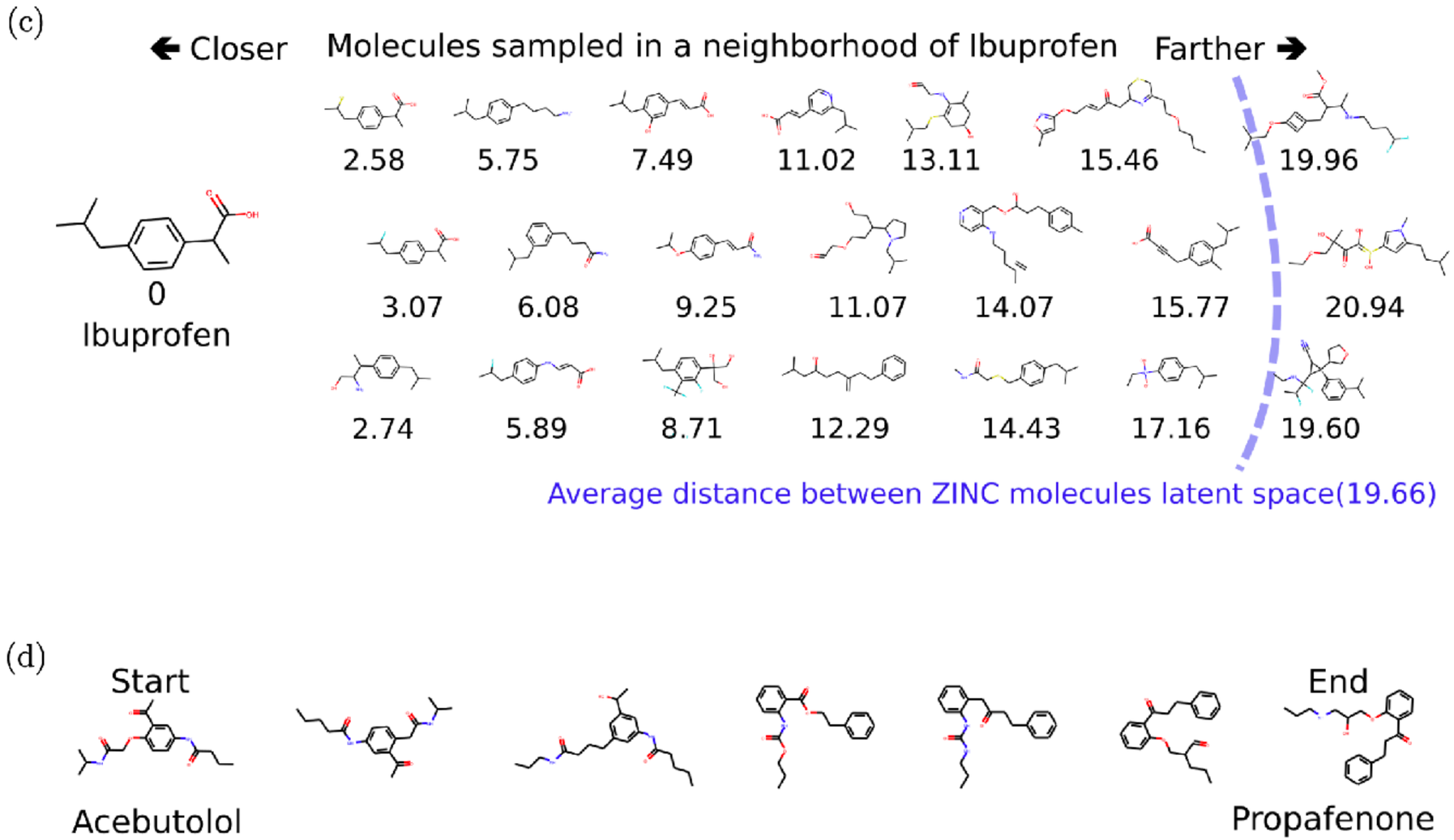


Fig.2 of Gómez-Bombarelli et al. (2016, arXiv:1610.02415)