EECS 230 Deep Learning

Lecture 16: Variational Autoencoder

Some slides from Frank Noe and Pascal Poupart

Outline

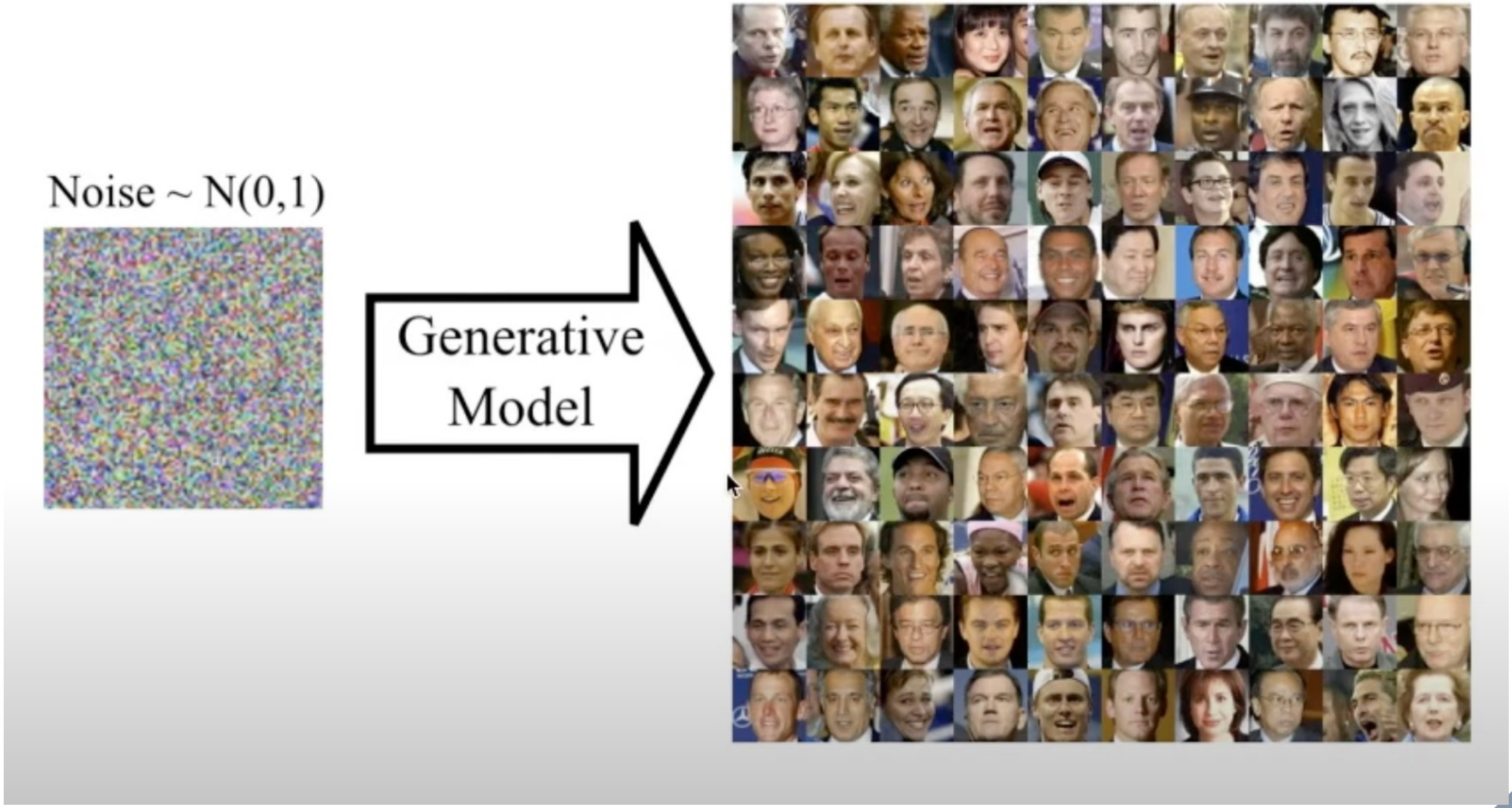
- Generative model
- Variational autoencoder
  - Autoencoder
  - Variational autoencoder

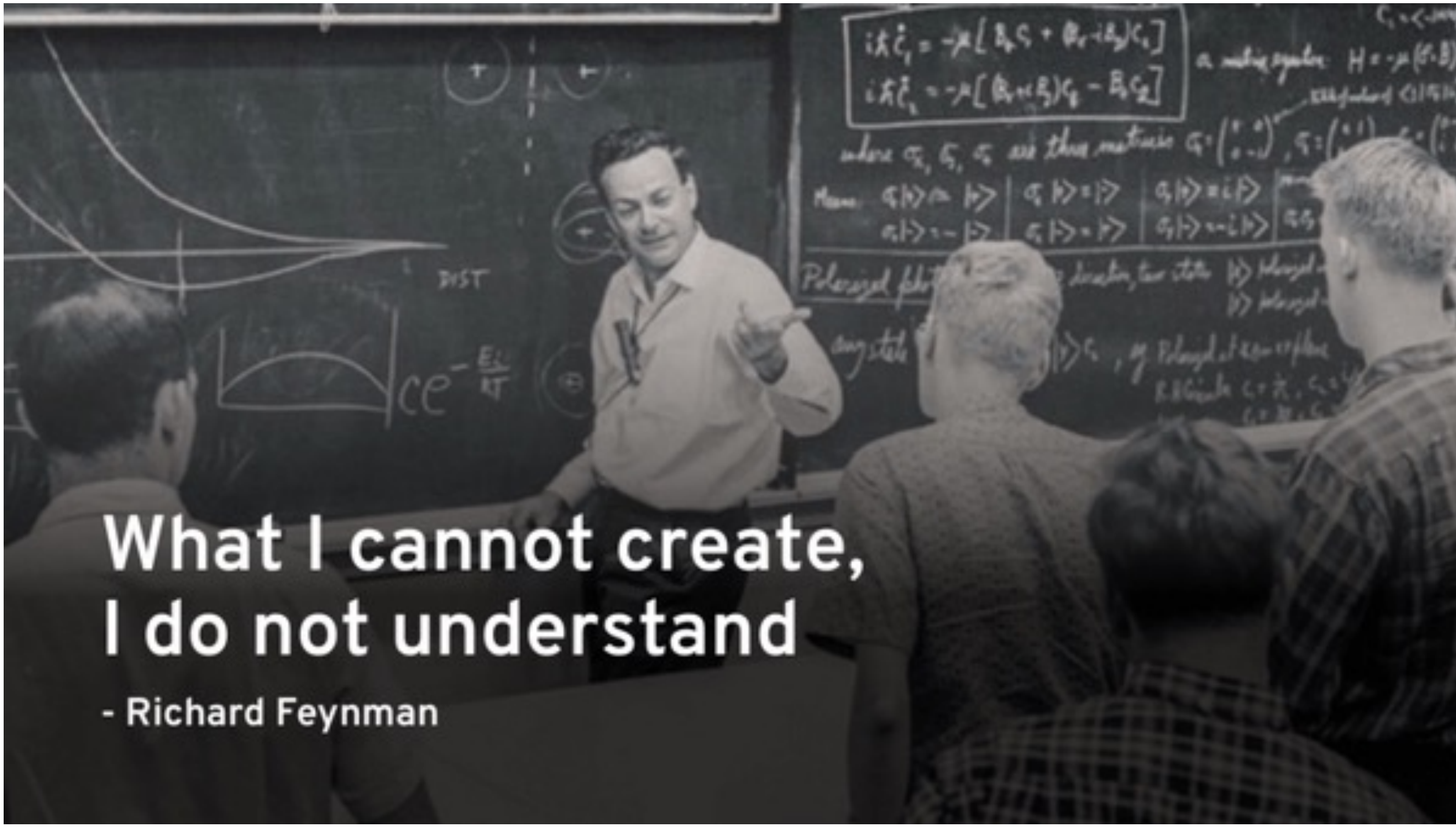
Generative model

So far…

- Discriminative model $P(y|x)$
  - Given input data x, predict y
  - E.g., classification, regression
- Generative model
  - Model data distribution P(x)
  - Sample from P(x) to generate new data

Generative model




What I cannot create,
I do not understand
- Richard Feynman

Types of generative neural networks

- ❑Boltzmann machines
- ❑Sigmoid belief networks
- ❑**Variational autoencoders (inference net + generator net)**
- ❑**Generative adversarial networks (generator net + discriminator net)**
- ❑Normalizing flows
- ❑**Diffusion Models**
- ❑…

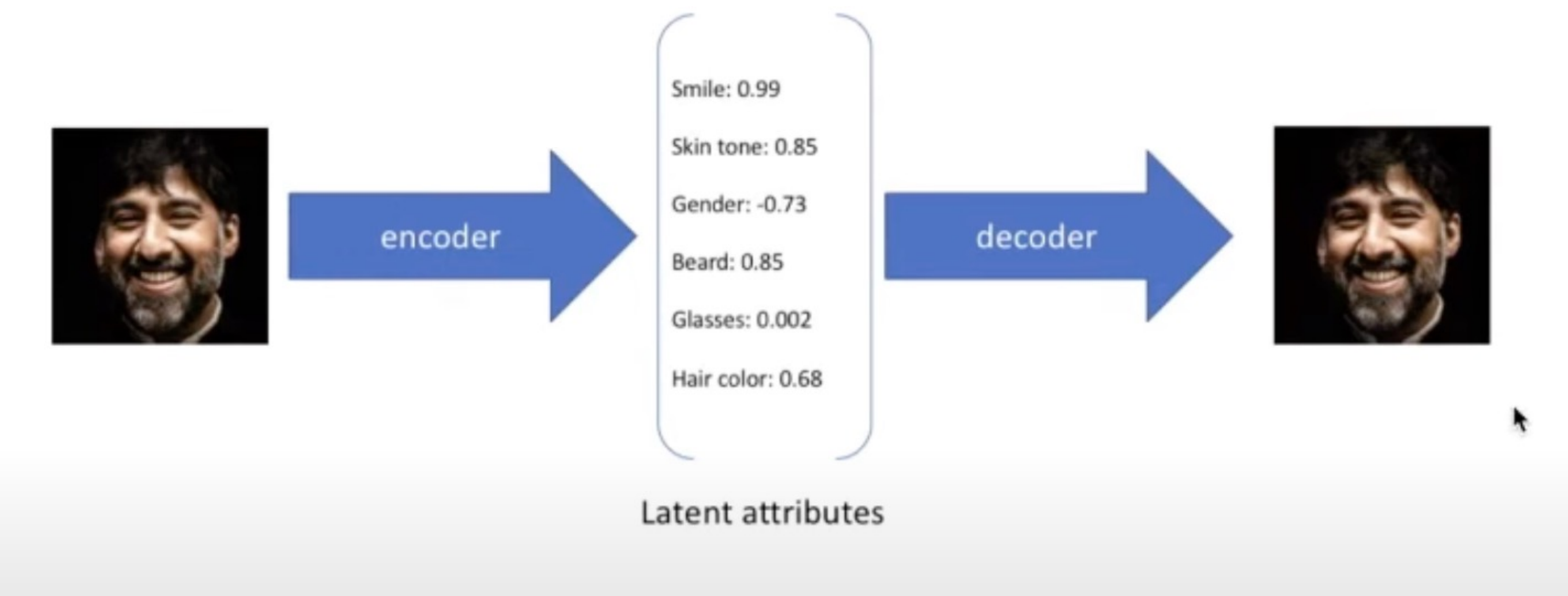
Variational Autoencoder

Autoencoder

- Special type of feed forward network for
  - Compression
  - Denoising
  - Sparse representation
  - Data generation

Autoencoder

- ❑Encoder: f( )
- ❑Decoder: g( )
- ❑Autoencoder: g(f(x)) = x



Latent variable encodes "essential" information about input data

Linear autoencoder

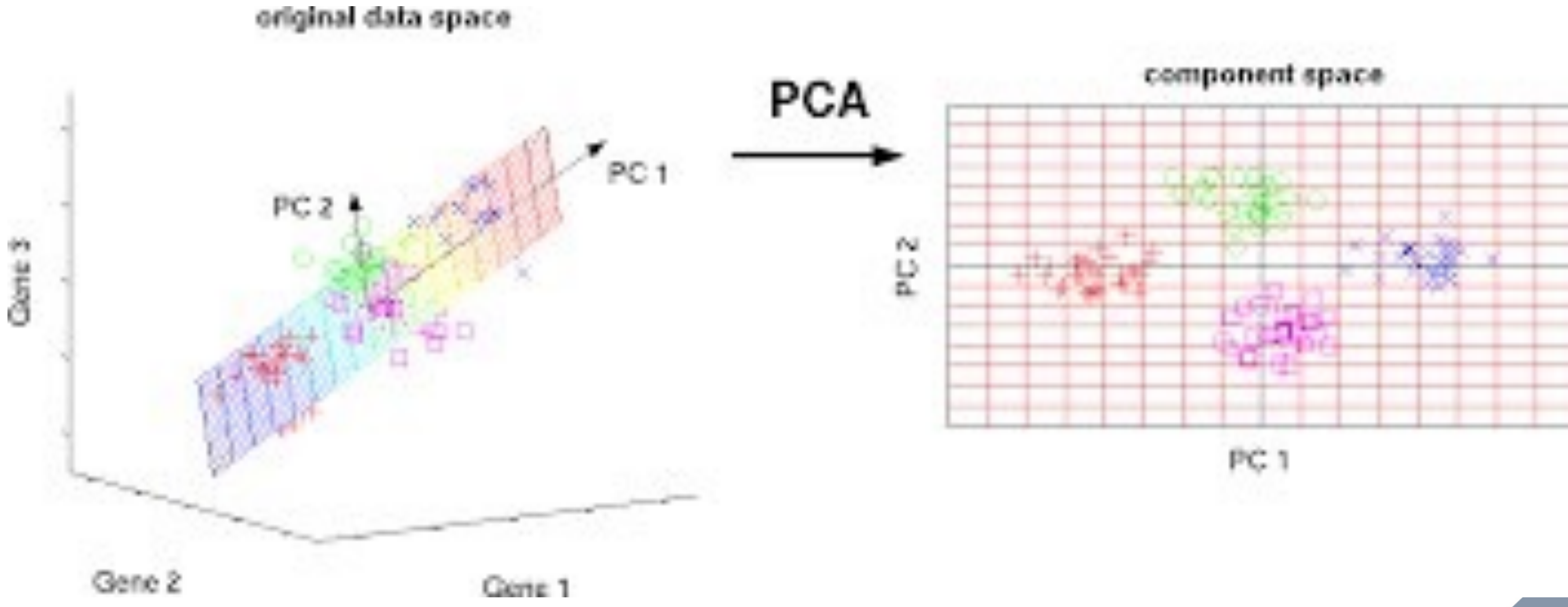
- Objective: find weights W_f and W_g that minimize reconstruction error

$$\min_{\boldsymbol{W}} \frac{1}{2} \sum_{n} \left|\left| \boldsymbol{W}_g \boldsymbol{W}_f \boldsymbol{x}_n - \boldsymbol{x}_n \right|\right|_2^2$$

- When using Euclidean norm (i.e., squared loss), solution is the same as principal component analysis (PCA)

Recap: principle component analysis

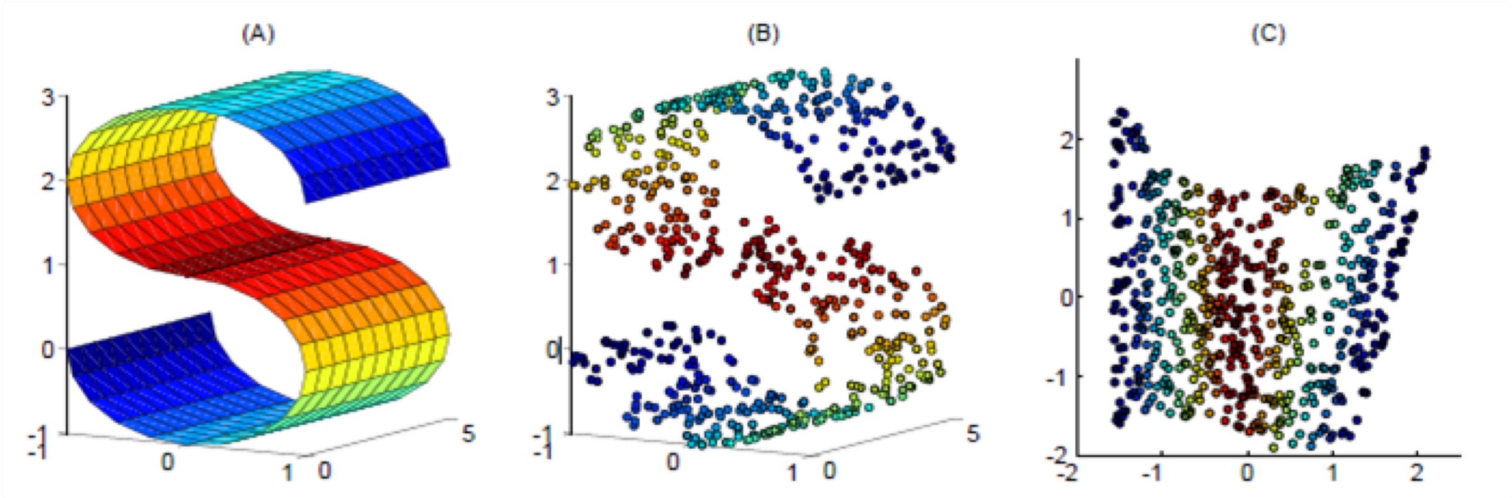
- Components with maximum variance



Non-linear autoencoder

- f( ) and g( ) are both non-linear functions

$$\min_{W} \frac{1}{2} \sum_{n} \left\| g\left(f\left(\boldsymbol{x}_n; \boldsymbol{W}_f\right); \boldsymbol{W}_g\right) - \boldsymbol{x}_n \right\|_2^2$$



Sparse representation

- When more hidden nodes than inputs, use regularization to constrain autoencoder
- Example: force hidden nodes to be sparse

$$\min_{\boldsymbol{W}} \frac{1}{2} \sum_n \left\| g(f(\boldsymbol{x}_n; \boldsymbol{W}_f); \boldsymbol{W}_g) - \boldsymbol{x}_n \right\|_2^2 + \mathrm{c}\, \underbrace{nnz\left(f(\boldsymbol{x}_n; \boldsymbol{W}_f)\right)}_{\textbf{Sparse hidden nodes}}$$

where $nnz\left(f(\boldsymbol{x}_n; \boldsymbol{W}_f)\right)$ is the number of non-zero entries in the vector produced by f.

- Aporoximate objective: L_1 regularization

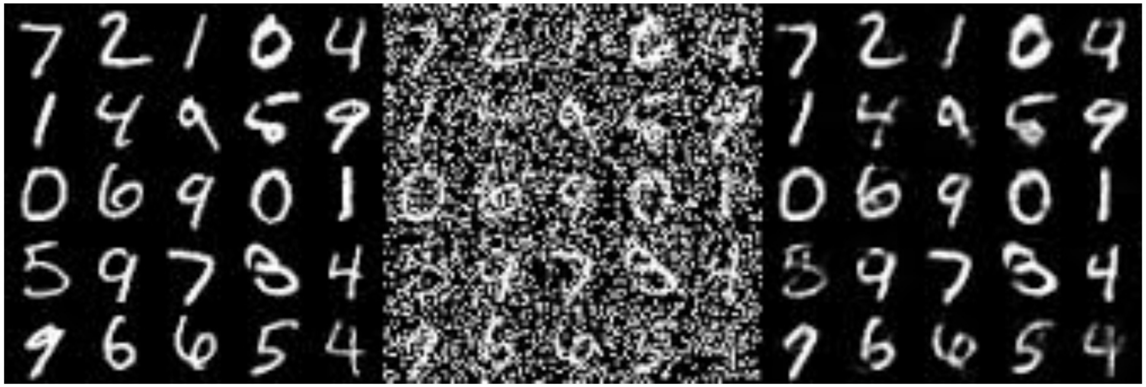
$$\min_{\boldsymbol{W}} \frac{1}{2} \sum_n \left\| g(f(\boldsymbol{x}_n; \boldsymbol{W}_f); \boldsymbol{W}_g) - \boldsymbol{x}_n \right\|_2^2 + \mathrm{c} \left\| f(\boldsymbol{x}_n; \boldsymbol{W}_f) \right\|_1$$

Denoising autoencoder

- Consider noisy version $\tilde{x}$ of the input x

$$\min_{\boldsymbol{W}} \frac{1}{2} \sum_{n} \left\| g(f(\tilde{\boldsymbol{x}}_n; \boldsymbol{W}_f); \boldsymbol{W}_g) - \boldsymbol{x}_n \right\|_2^2 + \mathrm{c} \left\| f(\tilde{\boldsymbol{x}}_n; \boldsymbol{W}_f) \right\|_1$$

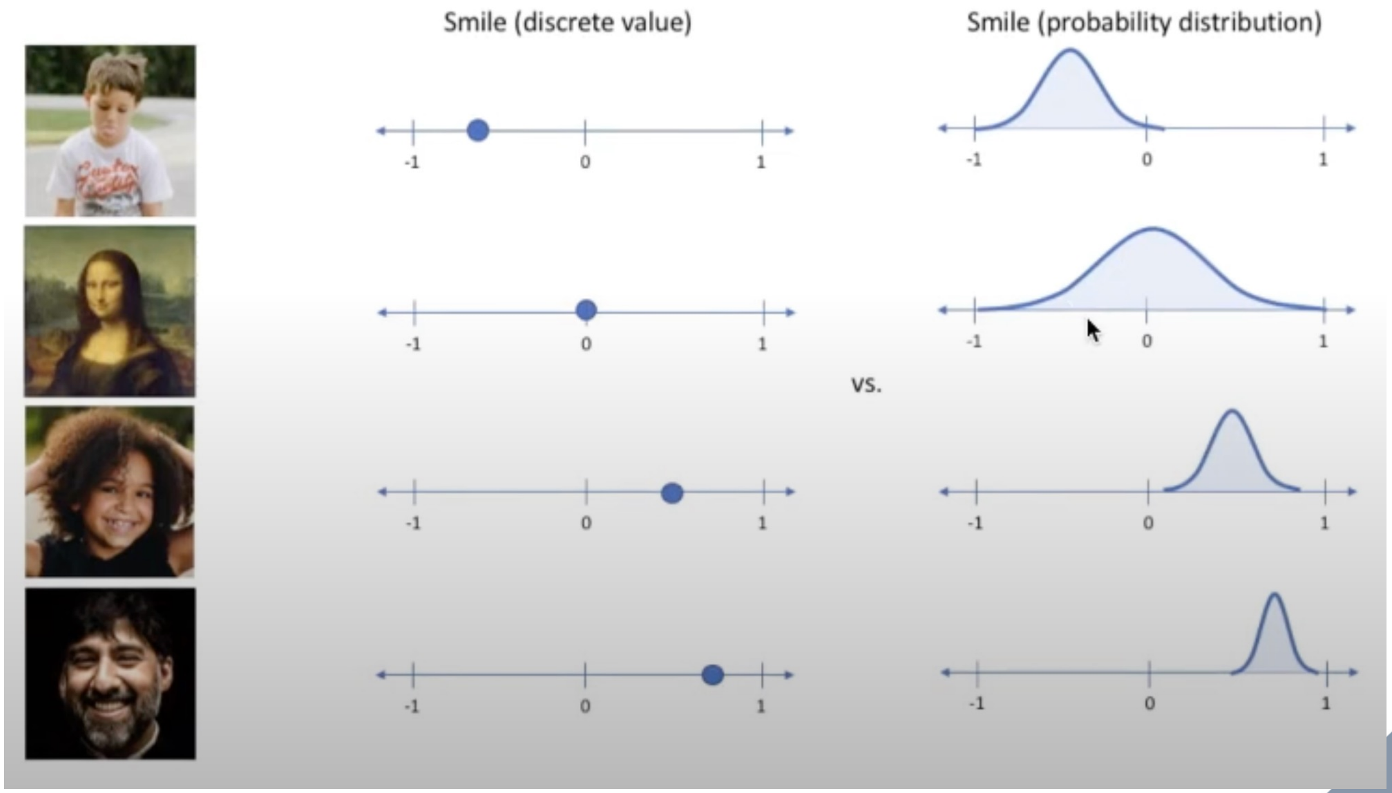
$\boldsymbol{x}$ | $\tilde{\boldsymbol{x}}$ | $g(f(\tilde{\boldsymbol{x}}))$



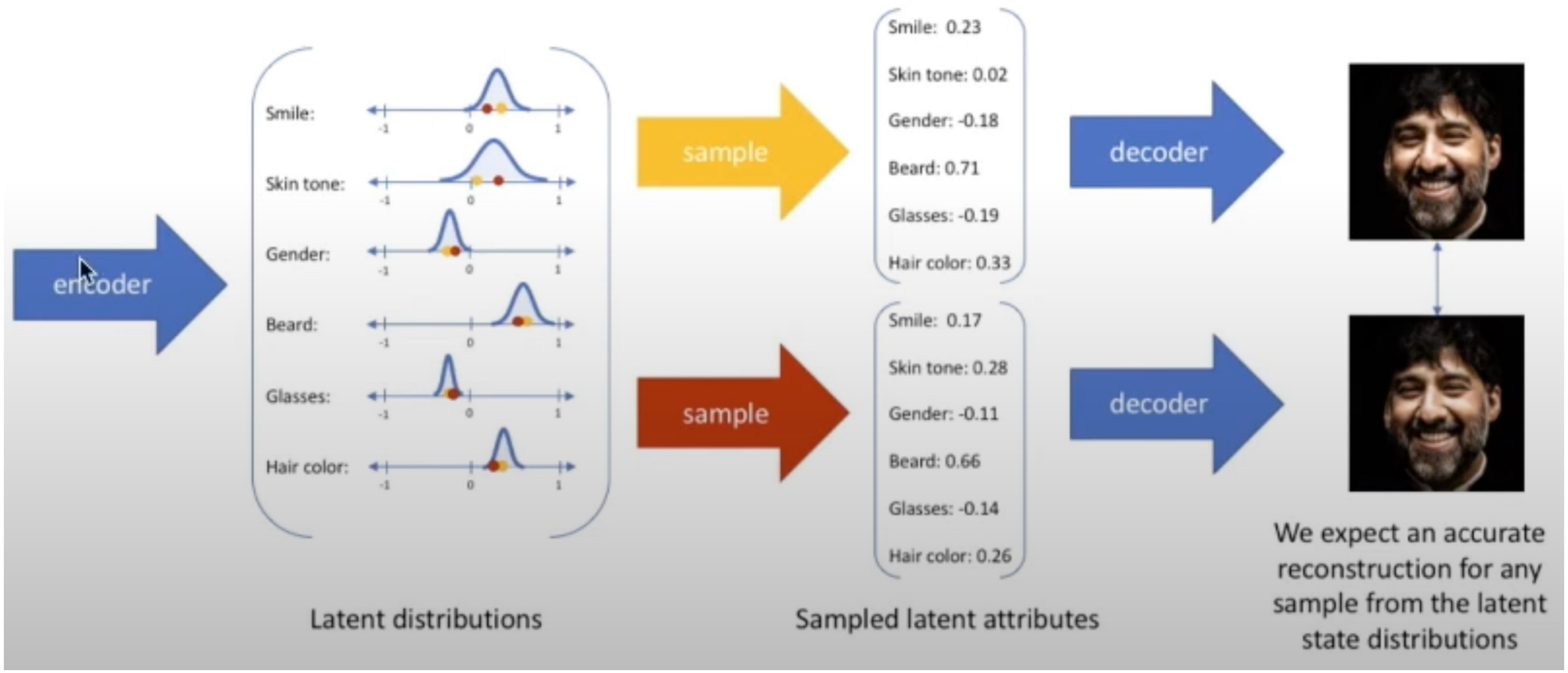
original | perturbed | reconstructed

Variational/Probabilistic autoencoder

- Instead of a single value for each attribute, represent each attribute as a range of values
- VAE: Describe latent attribute in probabilistic terms



Variational/Probabilistic autoencoder



Variational/Probabilistic autoencoder

- Let f() and g() represent conditional distributions
  - f: $Pr(h|x; w_f)$
  - g: $Pr(x|h; w_g)$
- The decoder g() can be treated as a generative model
  - First sample h from Pr(h)
  - Then sample x from $Pr(x|h; w_g)$

Variational autoencoder

- Idea: train encoder $\Pr(h|x; w_f)$ to approach a simple and fixed distribution, e.g., $N(h; 0, I)$

$$\max_{\boldsymbol{W}} \sum_n \log \Pr\left(\boldsymbol{x_n}; \boldsymbol{W_f}, \boldsymbol{W_g}\right) - \underbrace{c\, KL\left(\Pr\left(\boldsymbol{h} \middle| \boldsymbol{x}_n; \boldsymbol{W_f}\right) || N(\boldsymbol{h}; \boldsymbol{0}, \boldsymbol{I})\right)}_{\substack{\text{Kullback-Leibler divergence} \\ \text{Distance measure for distributions}}}$$

Variational Autoencoder Likelihood

- How to compute $\Pr(x_n; W_f, W_g)$?

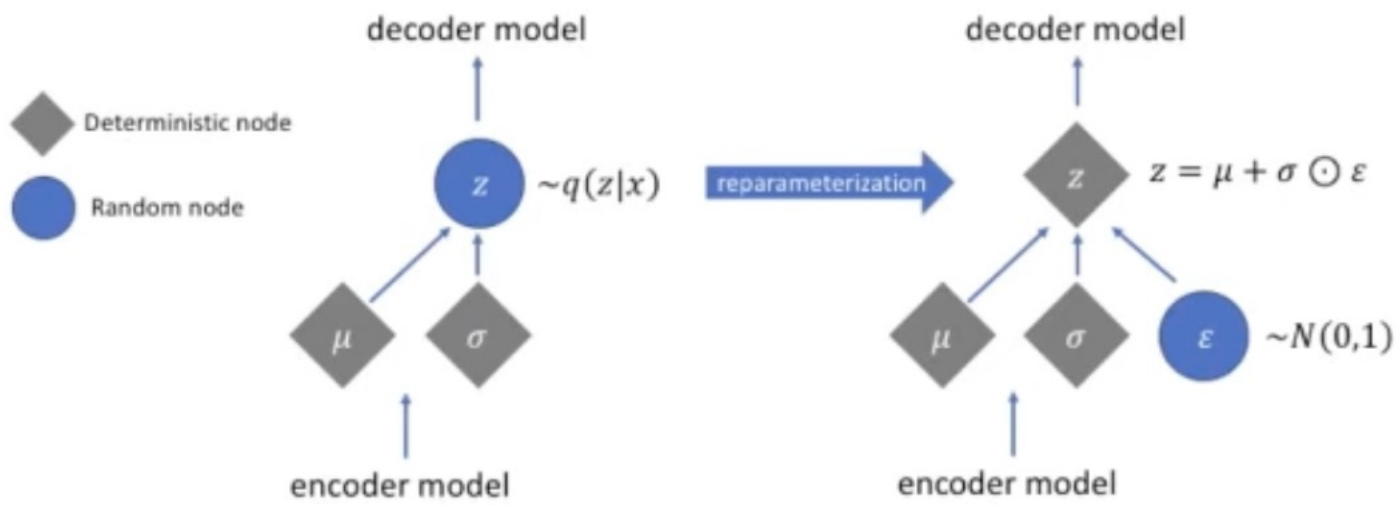
$$\Pr(\boldsymbol{x}_n; \boldsymbol{W}_f, \boldsymbol{W}_g) = \int_{\boldsymbol{h}} \Pr(\boldsymbol{x}_n | \boldsymbol{h}; \boldsymbol{W}_g) \Pr(\boldsymbol{h} | \boldsymbol{x}_n; \boldsymbol{W}_f)\, d\boldsymbol{h}$$

- Obtain mean and variance of $\Pr(h)$ by neural network

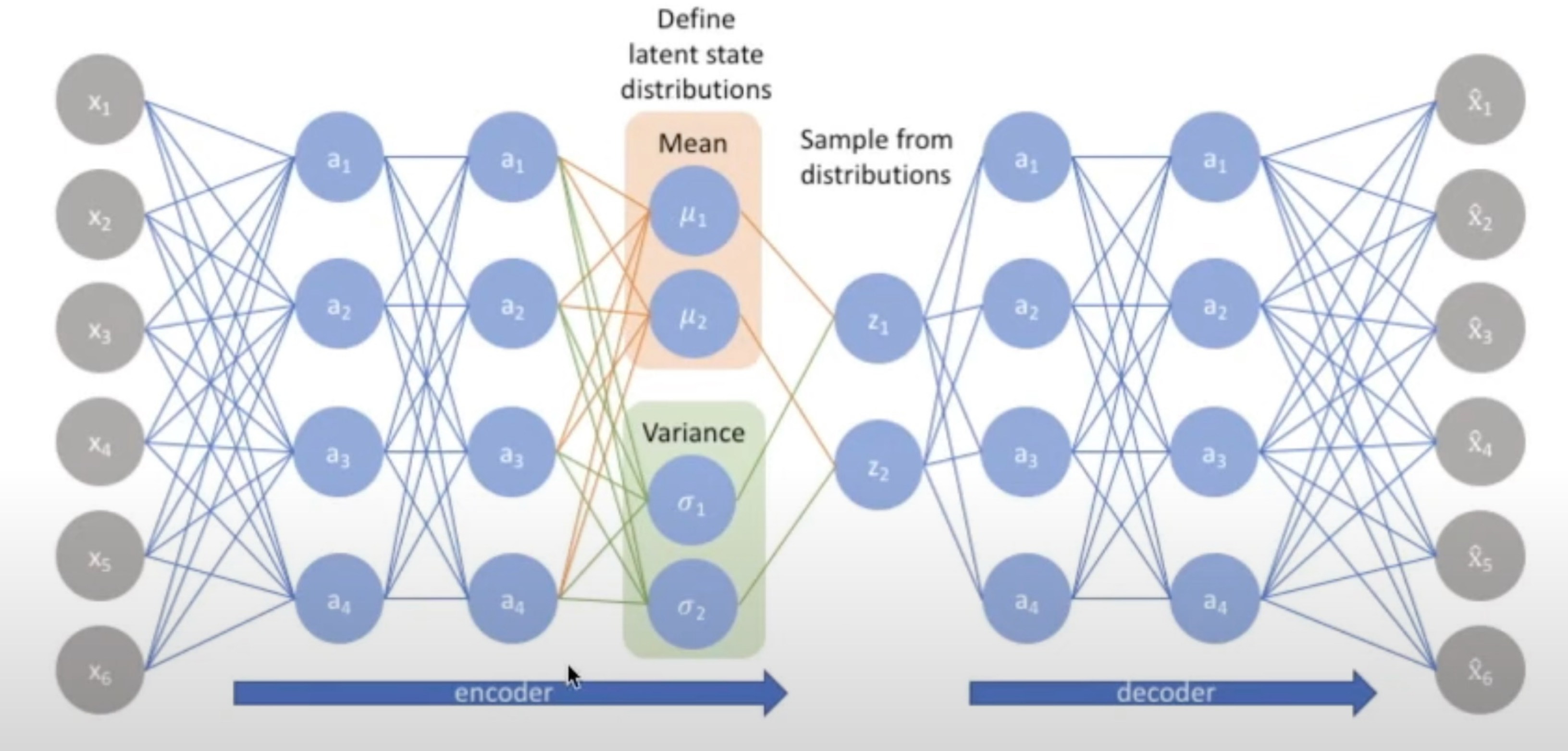
$$\Pr(\boldsymbol{h} | \boldsymbol{x}_n; \boldsymbol{W}_f) = N(\boldsymbol{h}; \mu_n(\boldsymbol{x}_n; \boldsymbol{W}_f), \sigma_n(\boldsymbol{x}_n; \boldsymbol{W}_f)\boldsymbol{I})$$

UCMERCED

Reparameterization trick

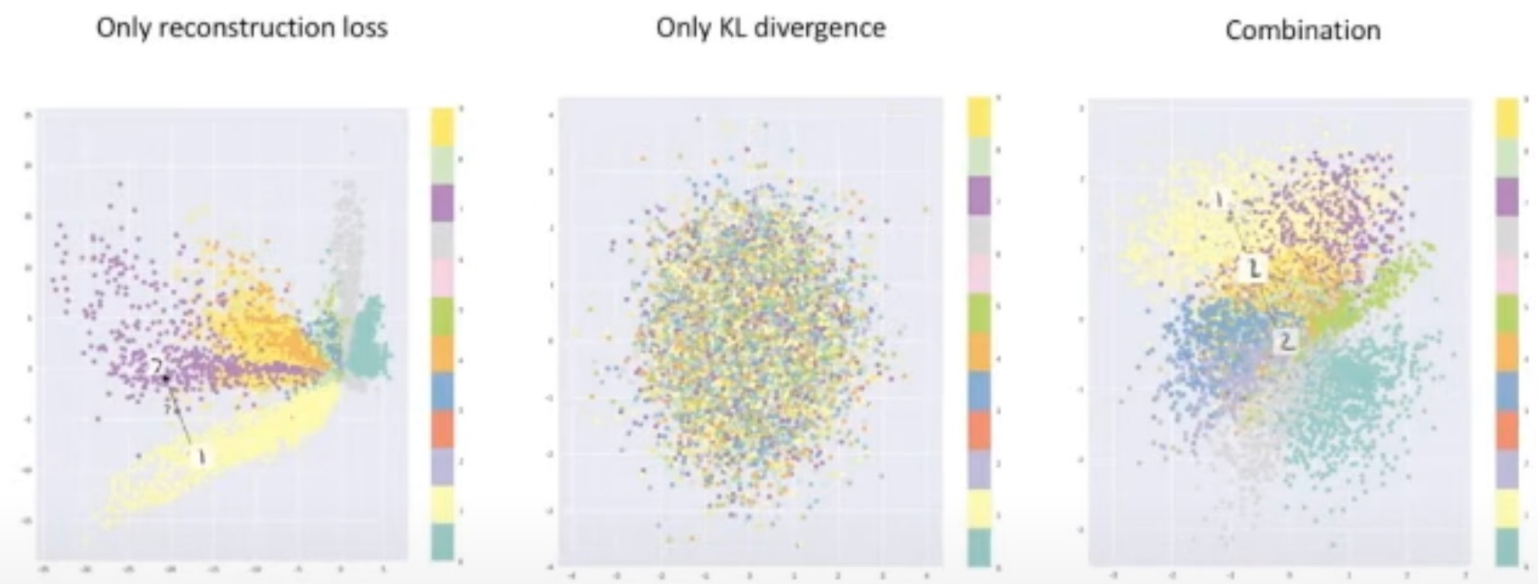


VAE implementation



MNIST VAE

$$\max_{\boldsymbol{W}} \sum_{n} \log \Pr\left(\boldsymbol{x_n}; \boldsymbol{W_f}, \boldsymbol{W_g}\right) - \underbrace{c\, KL\left(\Pr\left(\boldsymbol{h} \middle| \boldsymbol{x}_n; \boldsymbol{W_f}\right) || N\left(\boldsymbol{h}; \boldsymbol{0}, \boldsymbol{I}\right)\right)}_{\text{Kullback-Leibler divergence}}$$



Examples from VAE