Non-linear autoencoders for dimensionality reduction

Let us recapitulate

- General principle of autoencoding

$$\text{Distortion} = \sum_{i=1}^{R} \left(\mathbf{x}_i - \text{DECODE}[\text{ENCODE}(\mathbf{x}_i)]\right)^2$$

$$\text{Distortion} \rightarrow \min$$

- Discrete autoencoder, k-means: $R^p \underset{ENCODE}{\rightarrow}$ integer number $\underset{DECODE}{\rightarrow} R^p$
- Continuous linear autoencoder, PCA: $R^p \overset{linear}{\underset{ENCODE}{\rightarrow}}$ real number $\overset{linear}{\underset{DECODE}{\rightarrow}} R^p$

- Non-linear autoencoder, ? : $R^p \overset{non-linear}{\underset{ENCODE}{\rightarrow}} R^m \overset{non-linear}{\underset{DECODE}{\rightarrow}} R^p$

Non-linear PCA using autoassociative neural networks (Mark Kramer, AIChe, 1991)

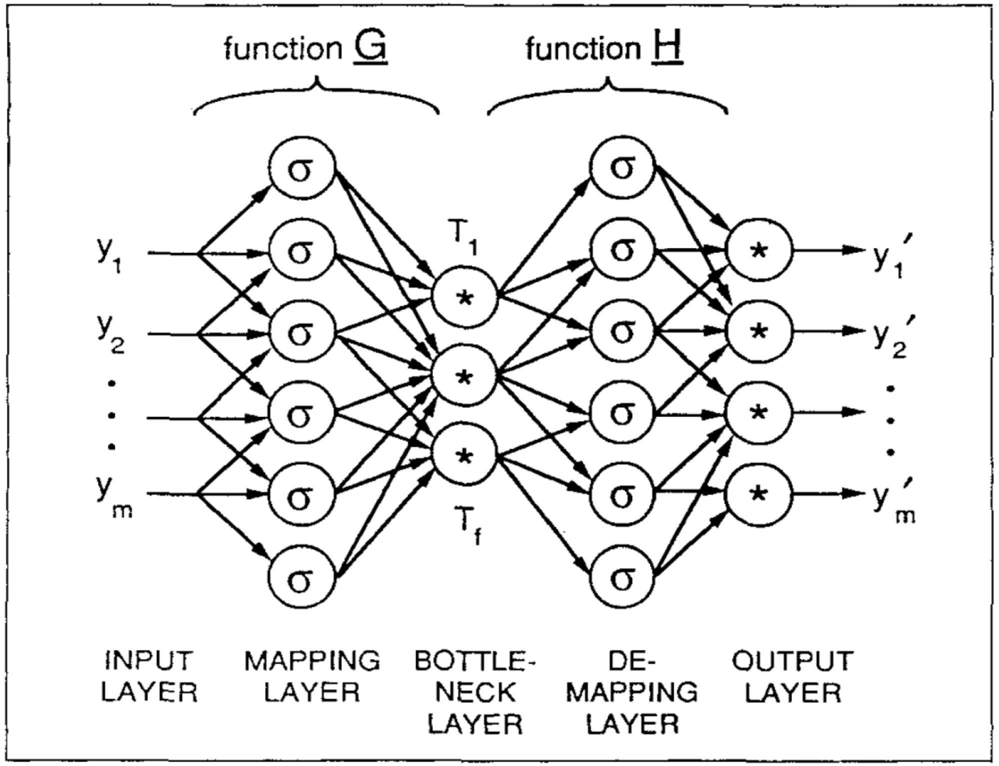
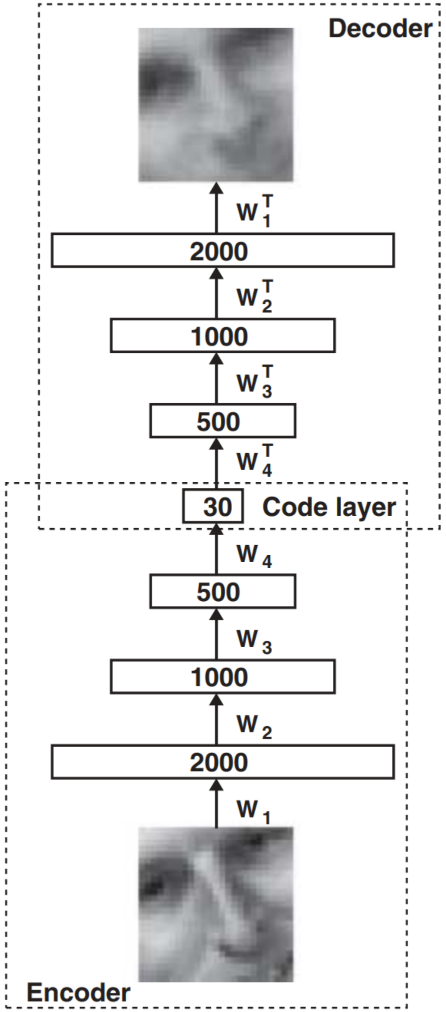


Figure 2. Network architecture for simultaneous determination of *f* nonlinear factors using an autoassociative network.

σ indicates sigmoidal nodes, * indicates sigmoidal or linear nodes.

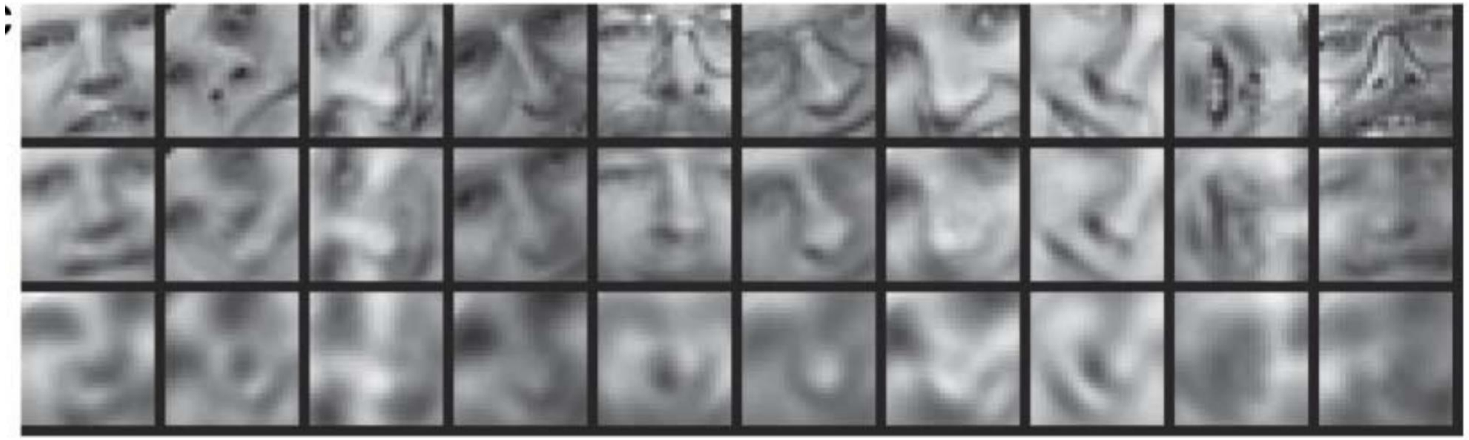
Reducing the Dimensionality of Data with Neural Networks (G.E.Hinton&R.R.Salakhutdinov, Science 2006)



Original

ANN autoencoder with 30 neurons

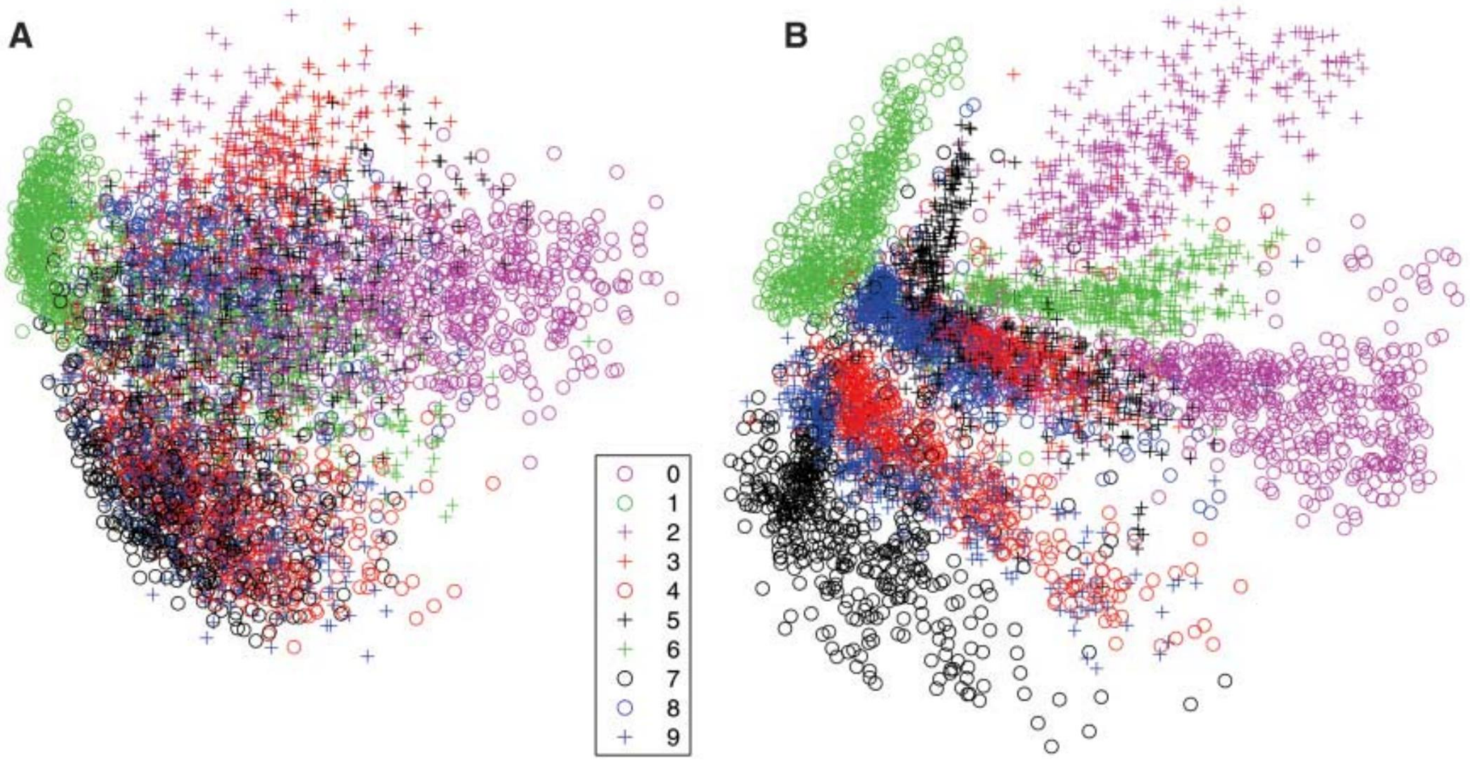
PCA with 30 dimensions



Reducing the Dimensionality of Data with Neural Networks (G.E.Hinton&R.R.Salakhutdinov, Science 2006)

MNIST dataset

Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).



Problems of training deep autoencoders

- **Data hungry approach**
- **Weight initialization** “With large initial weights, autoencoders typically find poor local minima; with small initial weights, the gradients in the early layers are tiny, making it infeasible to train autoencoders with many hidden layers”
- **Suggested approach for training:** pretraining every couple of neighbour layers using another type of neural networks (restricted Boltzman machine) + fine-tuning using back propagation afterwards

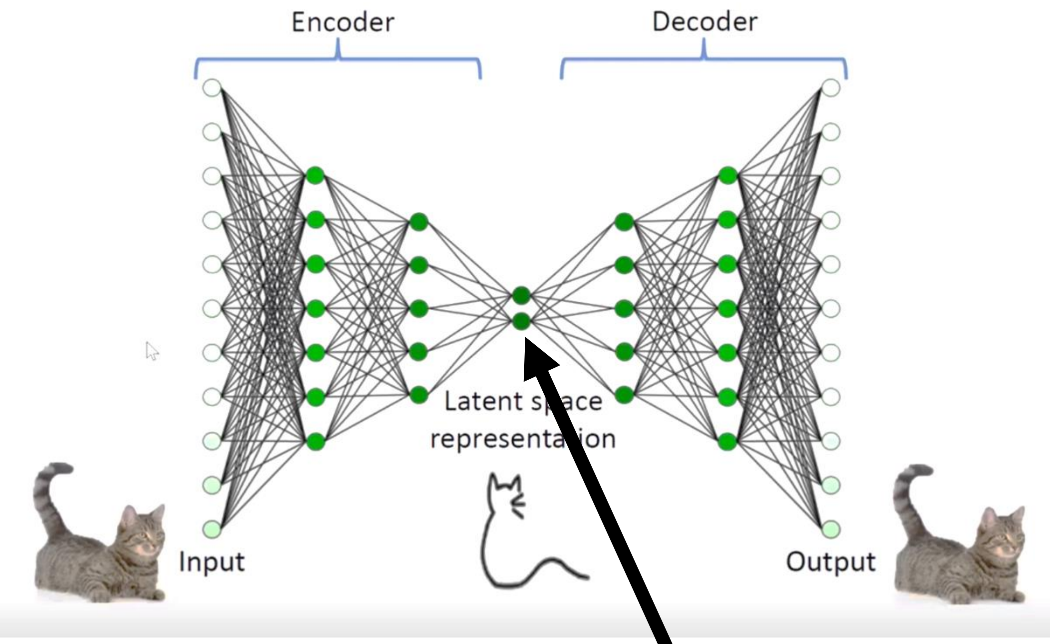
Advantages of deep vs shallow autoencoders*

- Depth can exponentially reduce the computational cost of representing some functions
- Depth can exponentially decrease the amount of training data needed to learn some functions
- Experimentally, deep autoencoders yield better compression compared to shallow or linear autoencoders

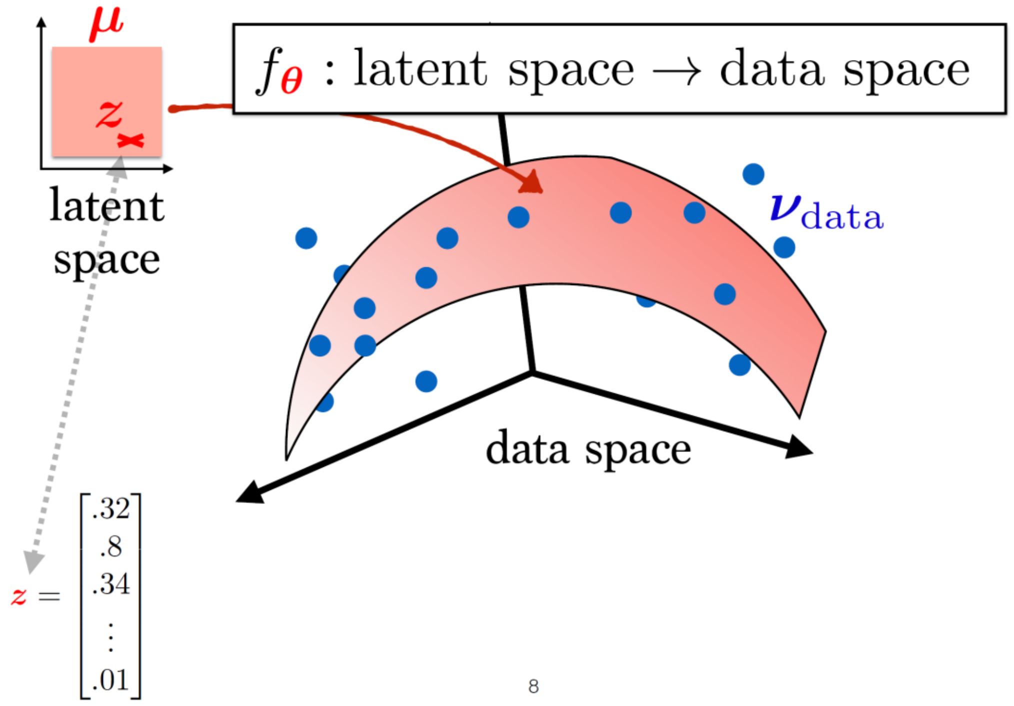
* Take this message critically

Deep generative models

Autoencoders are **injective** dimensionality reduction methods (we know the DECODE!)

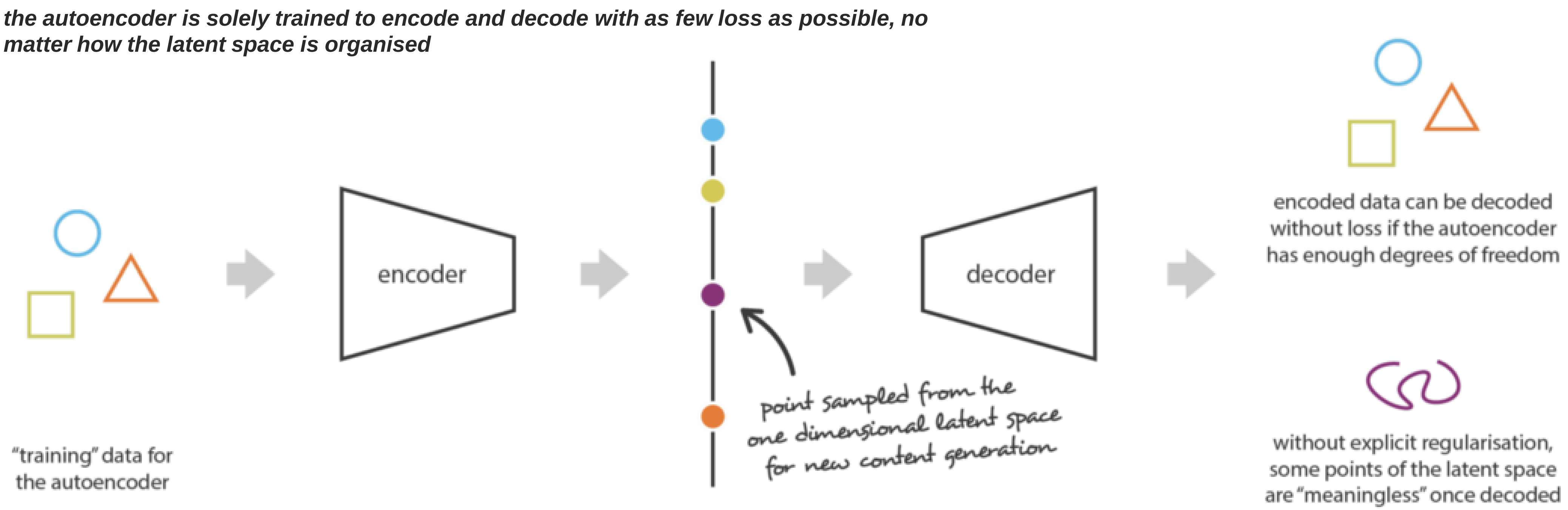


We need to know the distribution to sample! Let us denote it *z*. We want it to have some regular properties!



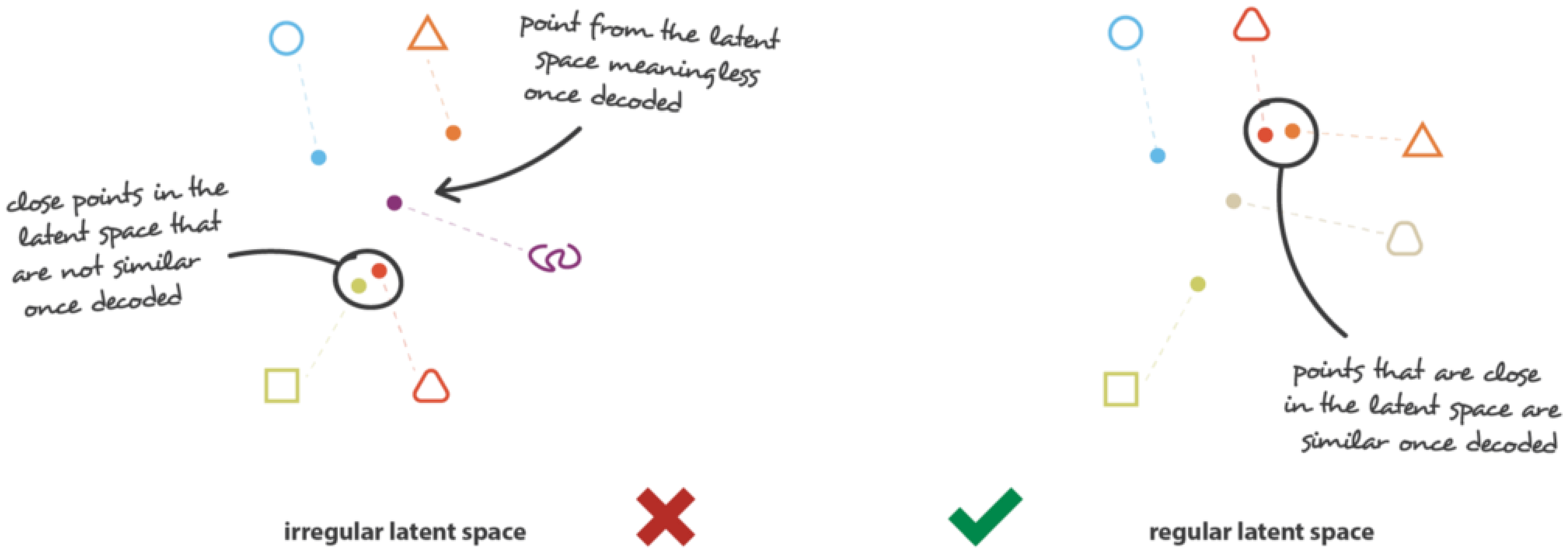
From https://marcocuturi.net/

Latent autoencoder space without regularization is frequently not usable for generating new data



https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Latent autoencoder space without regularization is frequently not usable for generating new data

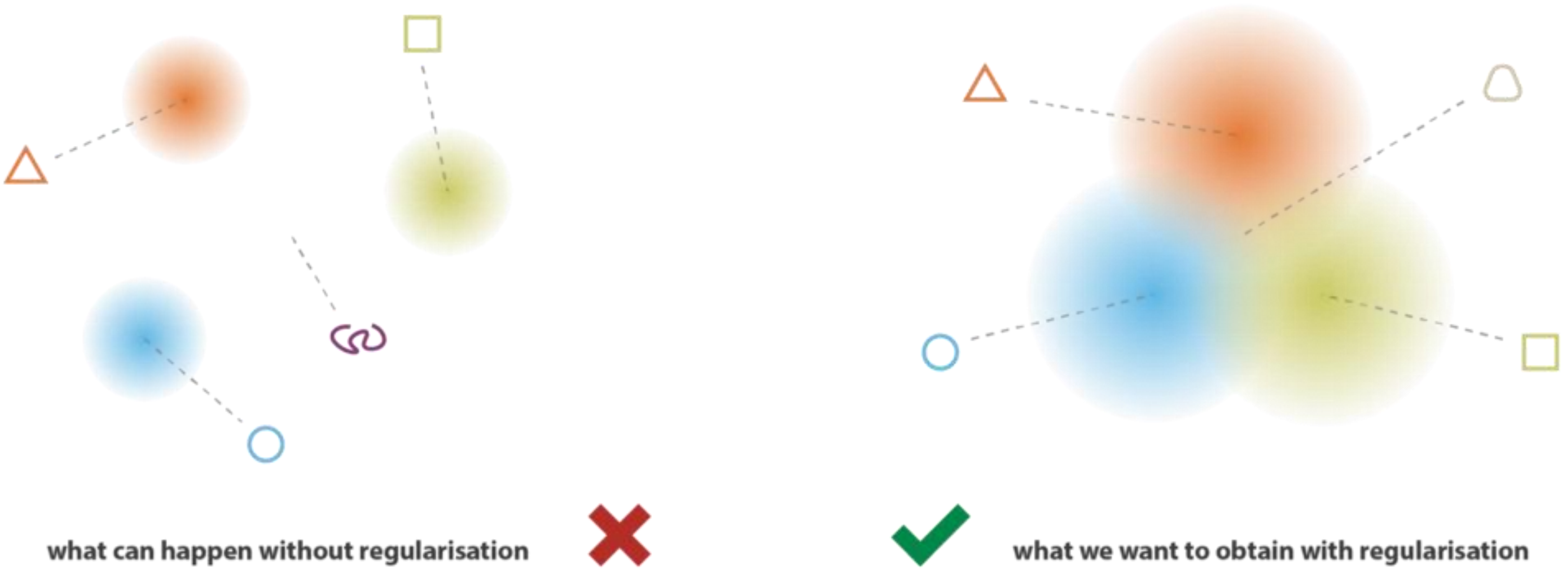


https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational AutoEncoder trick

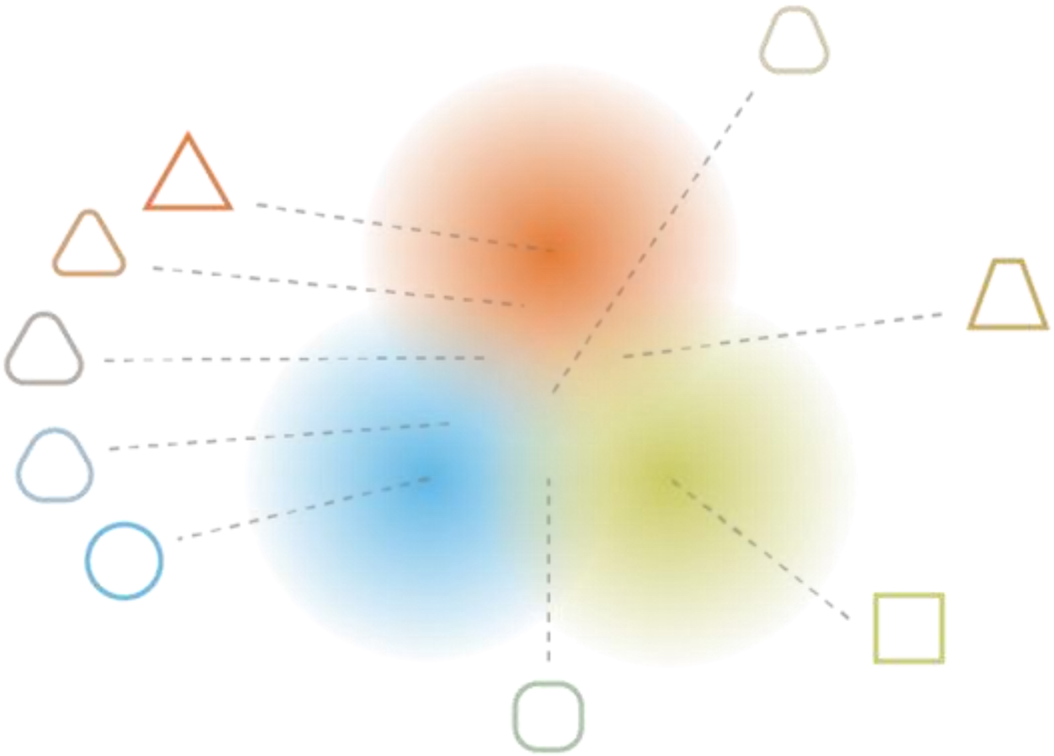
- Variational autoencoder (VAE) can be defined as being an autoencoder whose training is regularised to avoid overfitting and ensure that the latent space has good properties that enable generative process
- first, the input is encoded as distribution (*usually, Gaussian*) over the latent space
- second, a point from the latent space is sampled from that distribution
- third, the sampled point is decoded and the reconstruction error can be computed
- finally, the reconstruction error is backpropagated through the network

Latent distribution must be as compact as possible (regularization)

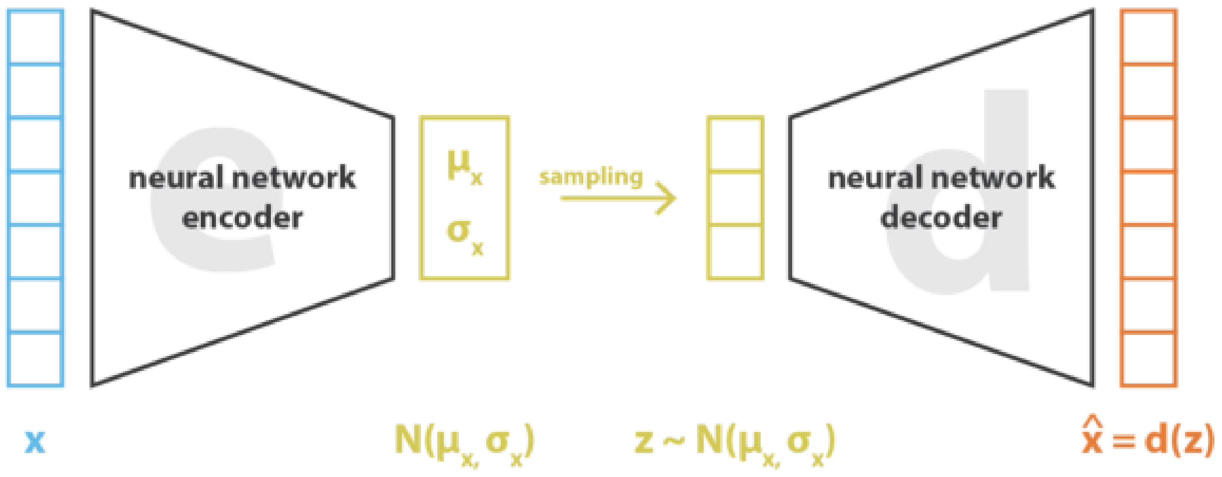


We force individual distributions p(z|x) to be as close to the standard Gaussian (zero mean, unit covariance) as possible

Then the new generated data is smooth



Mathematical formulation



$$\text{loss} = \| x - \hat{x} \|^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = \| x - d(z) \|^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

A special technique how to train such a network:

1) variational inference

2) reparametrization trick

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

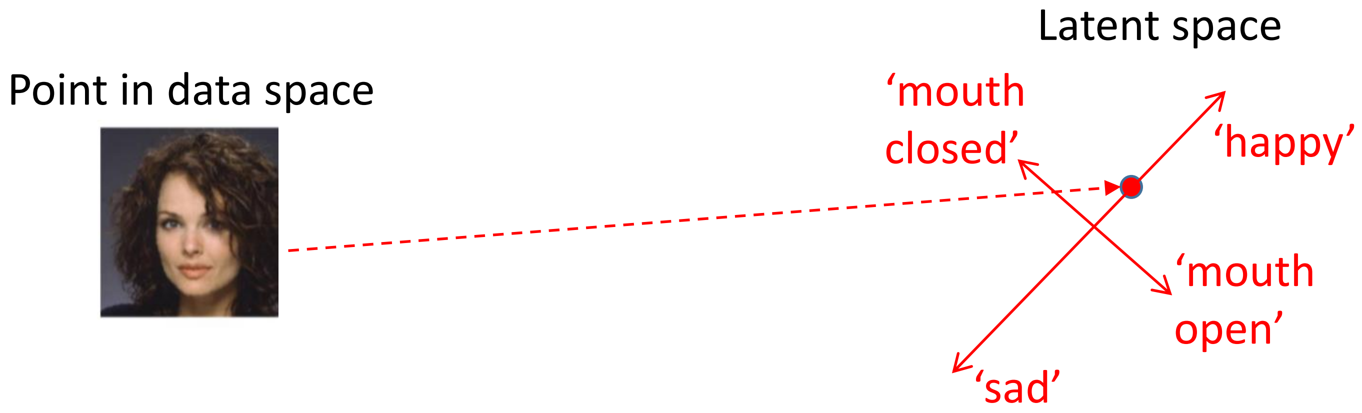


Figure 7: Decoupling attribute vectors for smiling (x-axis) and mouth open (y-axis) allows for more flexible latent space transformations. Input shown at left with reconstruction adjacent. (model: VAE from Lamb 16 on CelebA)

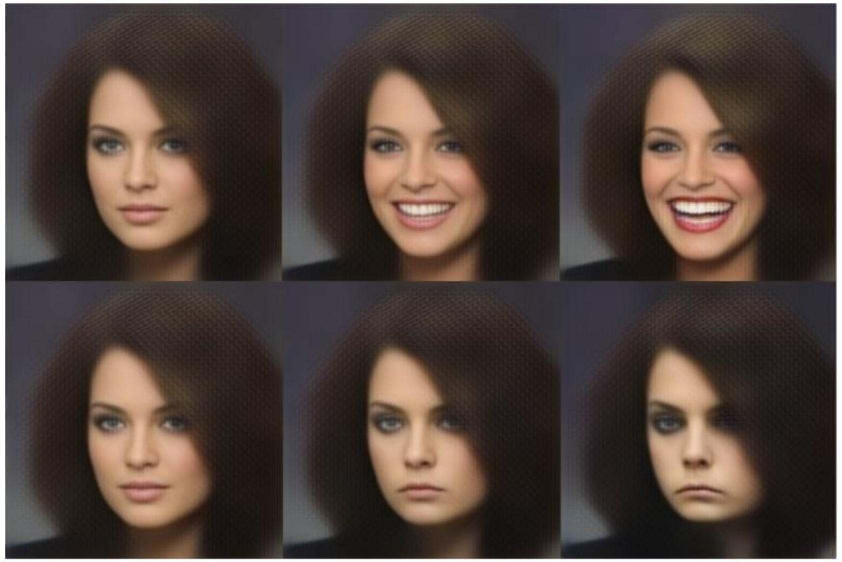


Figure 4.4: VAEs can be used for image resynthesis. In this example by White, 2016, an original image (left) is modified in a latent space in the direction of a *smile vector*, producing a range of versions of the original, from smiling to sadness.

https://arxiv.org/vc/arxiv/papers/1609/1609.04468v2.pdf

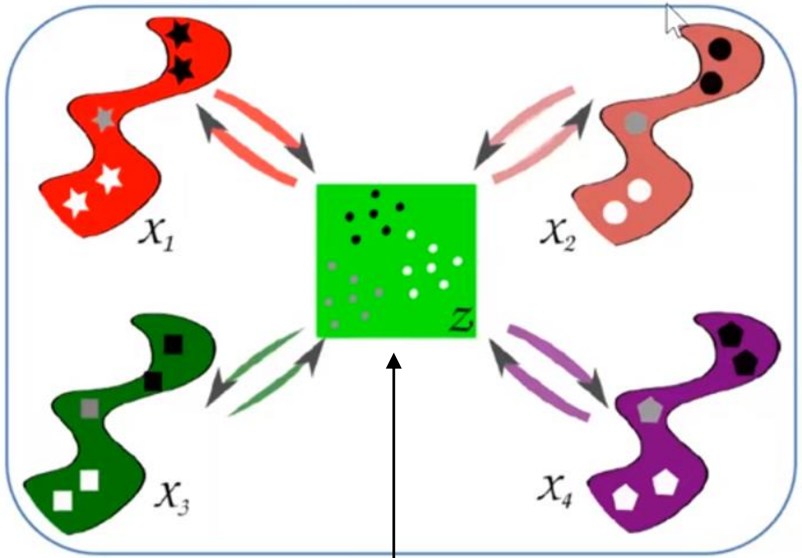
Which face is real?

https://www.whichfaceisreal.com/

You are **correct**. The image on the left is real.

[Play again.]()

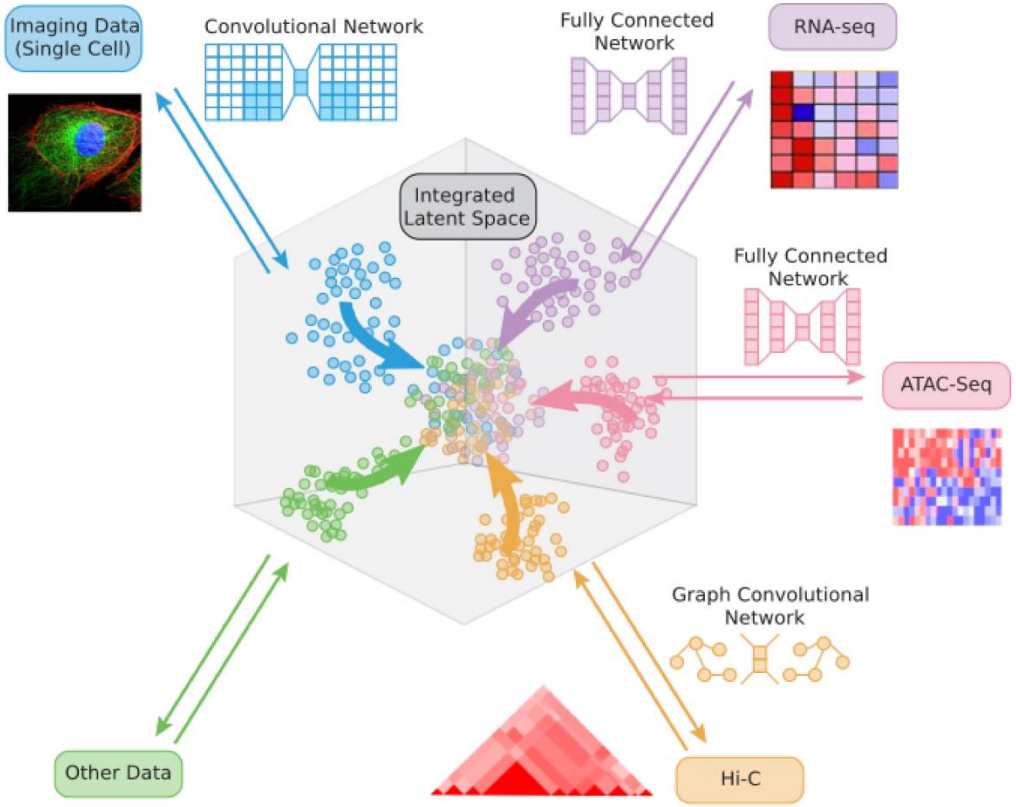
Mapping disjoint data spaces

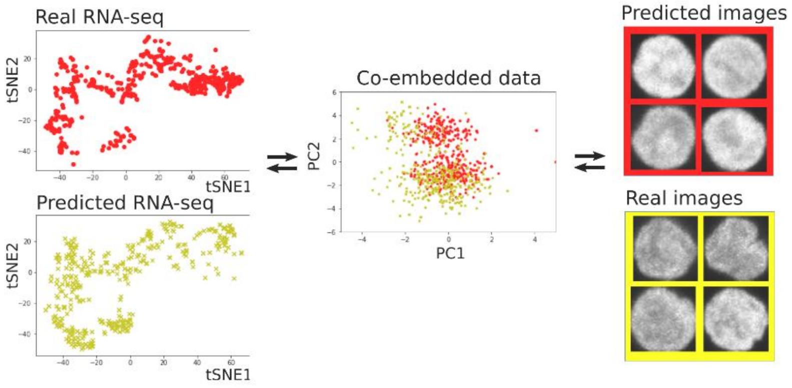


Distribution matching method: Generative Adversarial Network (GAN), optimal transport (Wasserstein distance), maximum mean discrepancy (MMD)

(from https://www.youtube.com/watch?v=Z2T9ZgCsRW8 , Caroline Uhler's presentation)

Learning latent spaces of biological systems: multi-domain data ‘translation’





From Yang et al, Multi-Domain Translation between Single-Cell Imaging and Sequencing Data using Autoencoders. BioRxiv, 2019

What you have to take with you

- Manifold learning methods either learn an explicit manifold (extensions of PCA) or are equivalent to projective non-linear dimensionality reduction (extensions of MDS)
- We can learn something which is more complex that a manifold (e.g., graphs approximating the data)
- Artificial neural network-based autoencoders and variational autoencoders provide both encoding and decoding functions
- Decoding (injection) function of any dimensionality reduction method can be used for generative data modeling