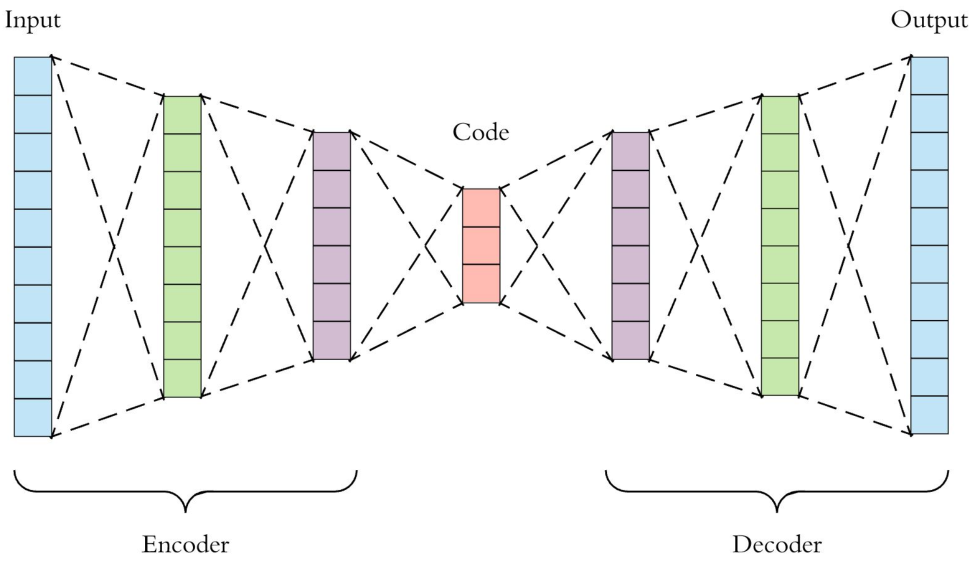
Classical Autoencoders

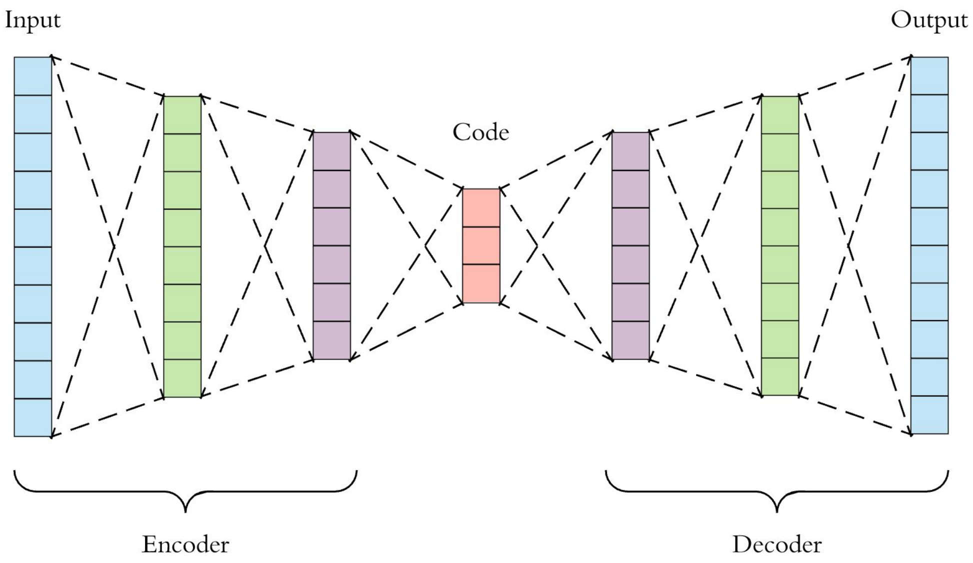
- Autoencoders (AE) are usually described as an unsupervised algorithm (no labels are needed for the training data) although they are more accurately a self-supervised algorithm (labels are automatically generated from inputs).
- The task during training is to reconstruct the input after having compressed it.



source

Classical Autoencoders

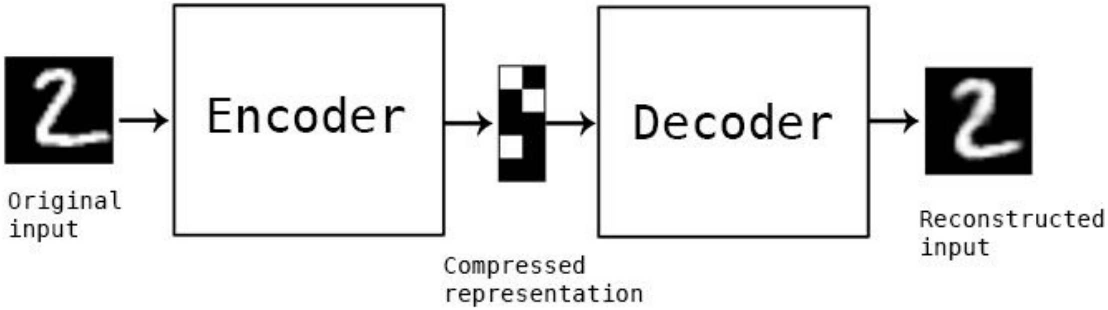
- An additional objective is to learn a compressed representation of your data.
- To build an autoencoder we need to define: an encoder function, a decoder function and a loss function.



source

Application to images

- We learn to compress an 3D RGB image to an 1D vector using a convolutional AE.



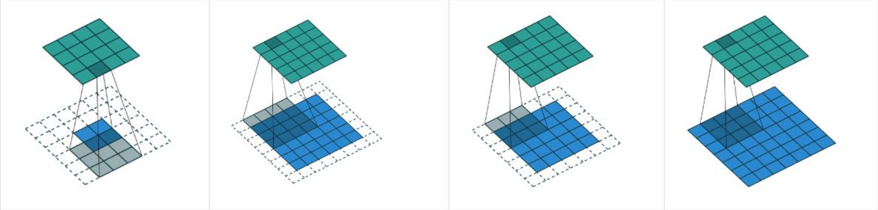
- **Encoder:** convolutional layers, pooling layers
- **Decoder:** transposed convolutional (or deconvolutional) layer, unpooling layers.

source

Application to images - Decoder layers

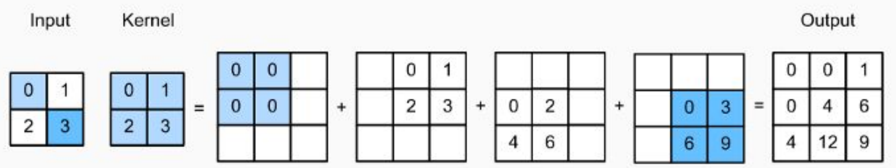
Possible transposed convolutions

N.B.: Blue maps are inputs, and cyan maps are outputs. [Animated GIF]



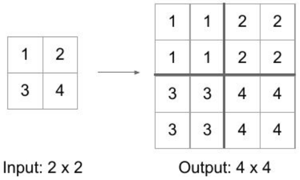
No padding, no strides, transposed	Arbitrary padding, no strides, transposed	Half padding, no strides, transposed	Full padding, no strides, transposed

A small example:

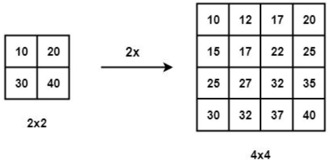


Possible unpoolings

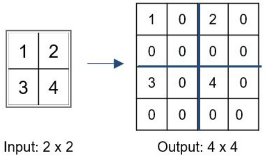
Nearest neighbors



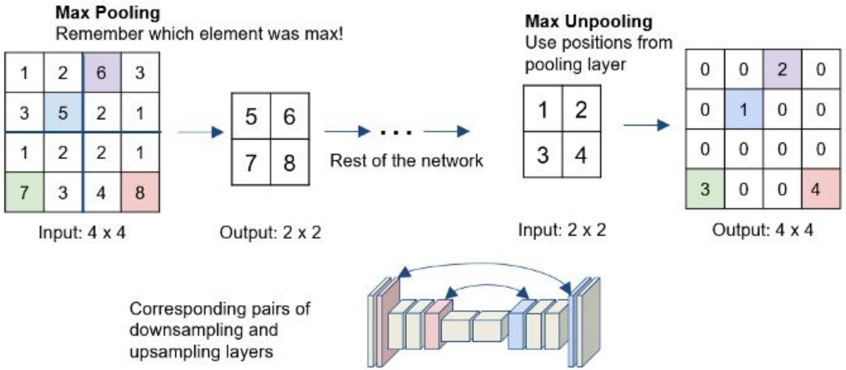
Bilinear interpolation



Bed of nails



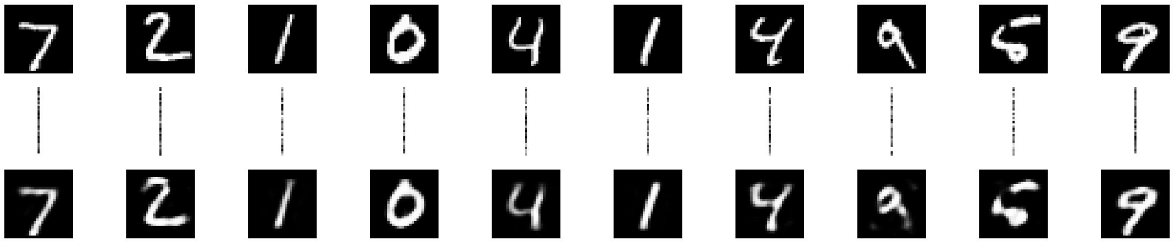
Max unpooling



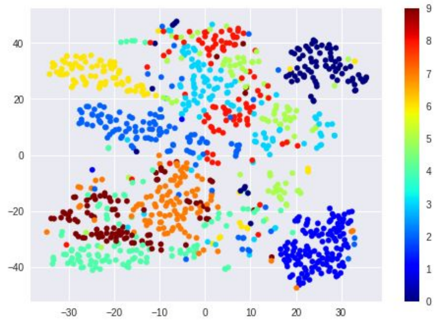
[source]

Exercice 1 - Encoding MNIST

- Create a shallow autoencoder to encode MNIST data, using Dense layers (reshape input image to vector).
- Visualize the results of encoding and decoding of test data.

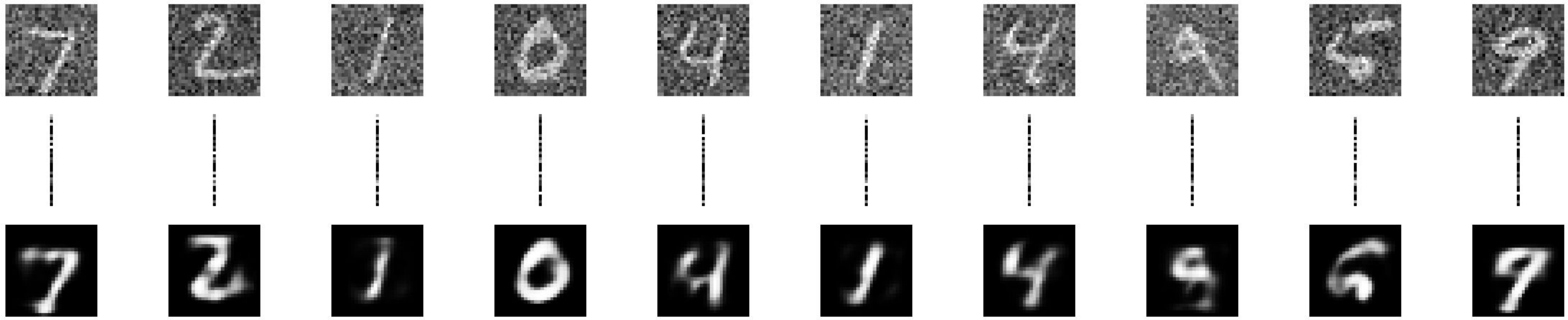


- Visualize the results of encoding of test data using the t-SNE algorithm.



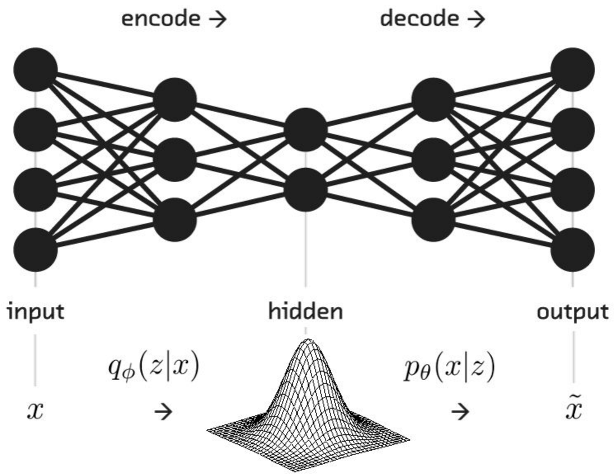
Exercice 2 - Denoising

- Use the previous model to create a denoising application.



Variational Autoencoders

- Variational Autoencoders (VAE) are like AE but with added constraints.
- So instead of learning an arbitrary function to encode the input, you are learning the parameters of a probability distribution (eg. a Gaussian) modeling your data.
- This gives more *structure* to the latent space: not only that point encodes that face, but neighbouring points encode *similar* faces.

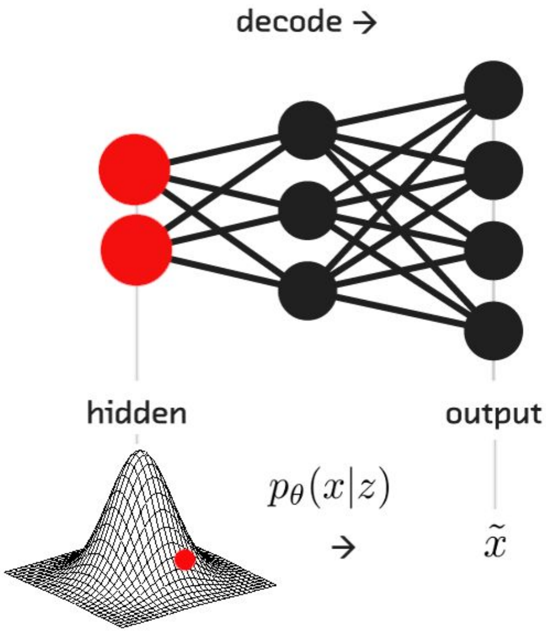


source

Variational Autoencoders - Inference

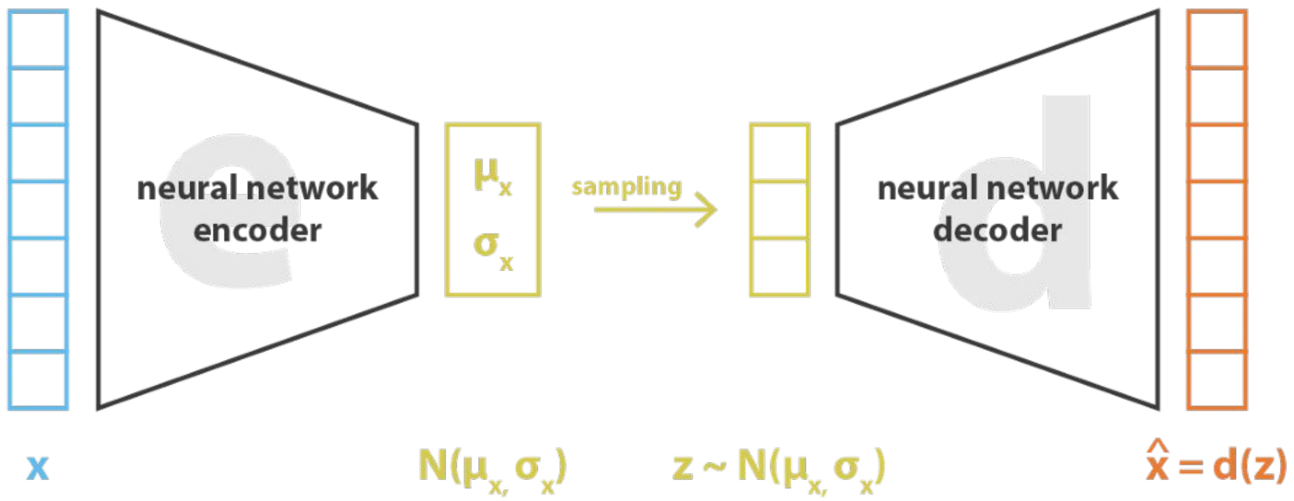
- At inference time, you sample points from this distribution, you pass them through the decoder and you get *new* data samples (that didn't existed in the training set)

  →VAEs are "generative models".



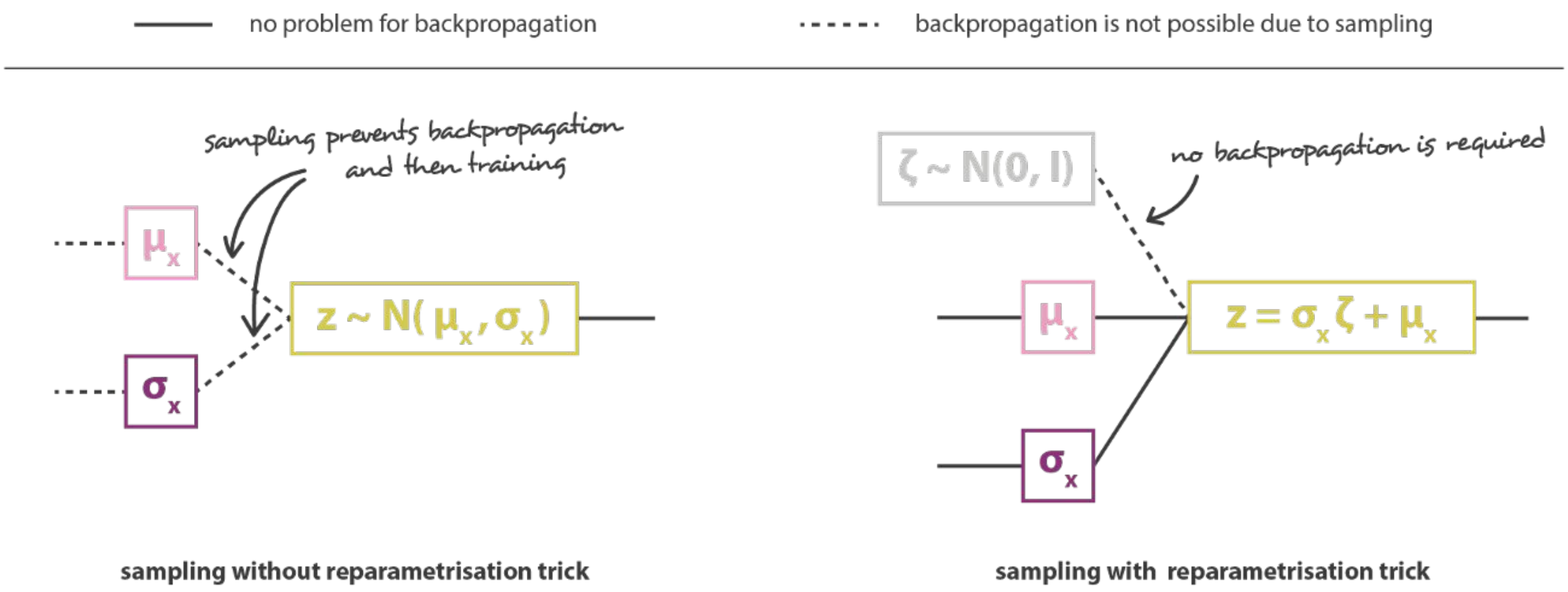
Variational Autoencoders - Training

- During training we predict the $\boldsymbol{\mu}_x$, $\boldsymbol{\sigma}_x$, we sample from N($\boldsymbol{\mu}_x$, $\boldsymbol{\sigma}_x$) and generate an output.



Variational Autoencoders - Training

- We can backpropagate through the random sampling by factoring out the randomness. This is the *reparametrization trick*: $\boldsymbol{z} \sim N(\boldsymbol{\mu}, \boldsymbol{\sigma}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot N(0, 1)$



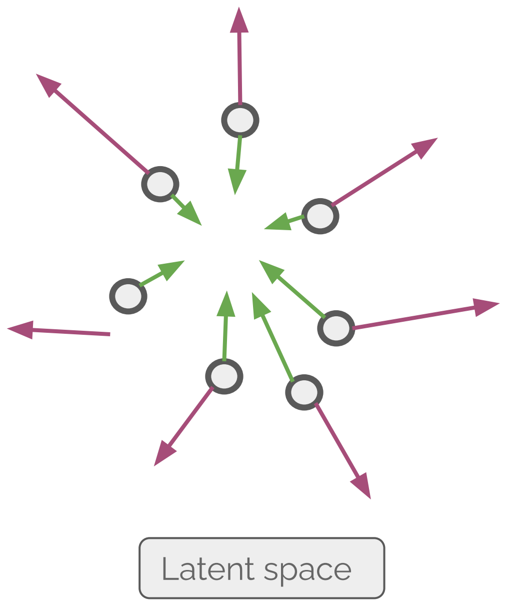
Variational Autoencoders - Loss terms

There are two terms:

- **reconstruction loss**: this is the same as in the AE.
- **KL loss**: this forces points in latent space to look as closely as possible as being sampled from a random normal gaussian N(0, 1).

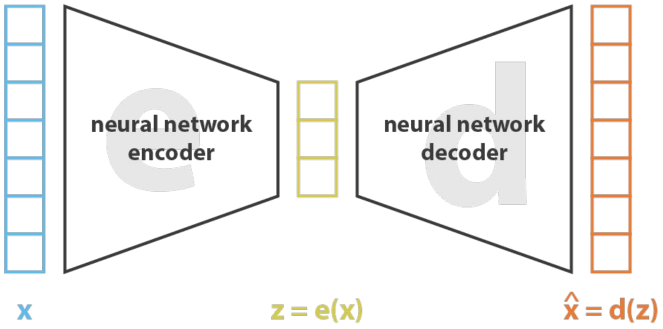
The two terms compete with each other:

- the first wants wants to make the reconstruction as good as possible (that is separate points as much as possible),
- the second wants to group points as much as possible, eventually overlapping, to have a continuous space (not sparse as in the AE case).



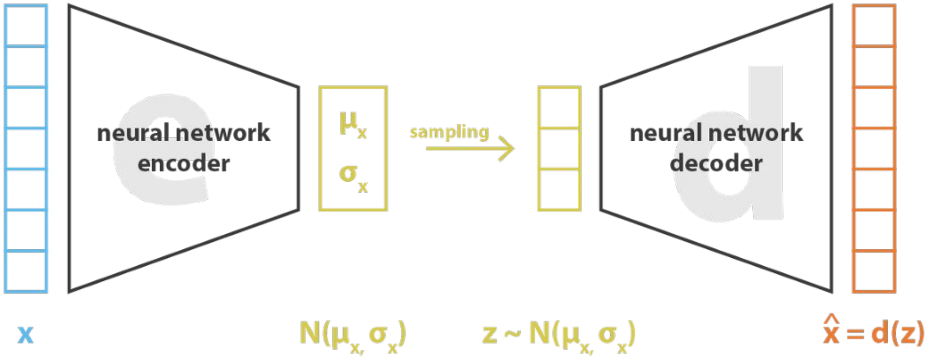
Variational Autoencoders - Loss terms

Autoencoders



$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

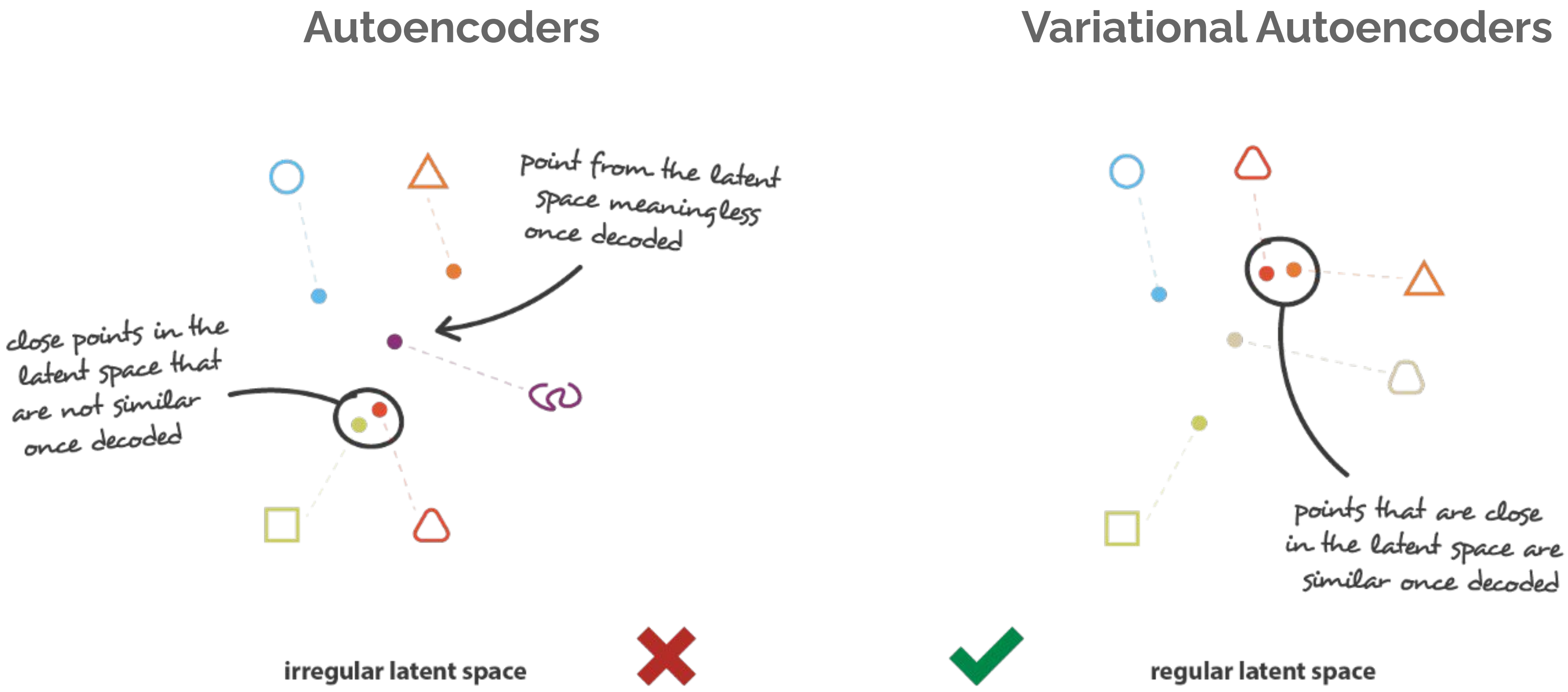
Variational Autoencoders



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

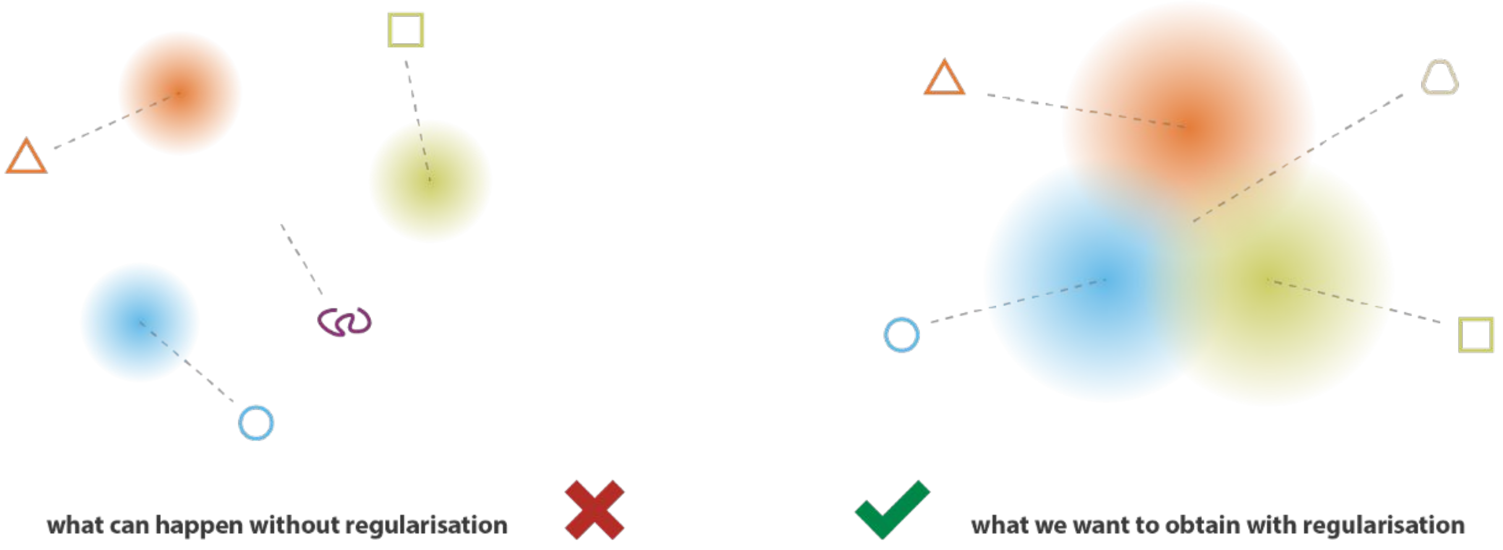
source

Variational Autoencoders - Loss terms



source

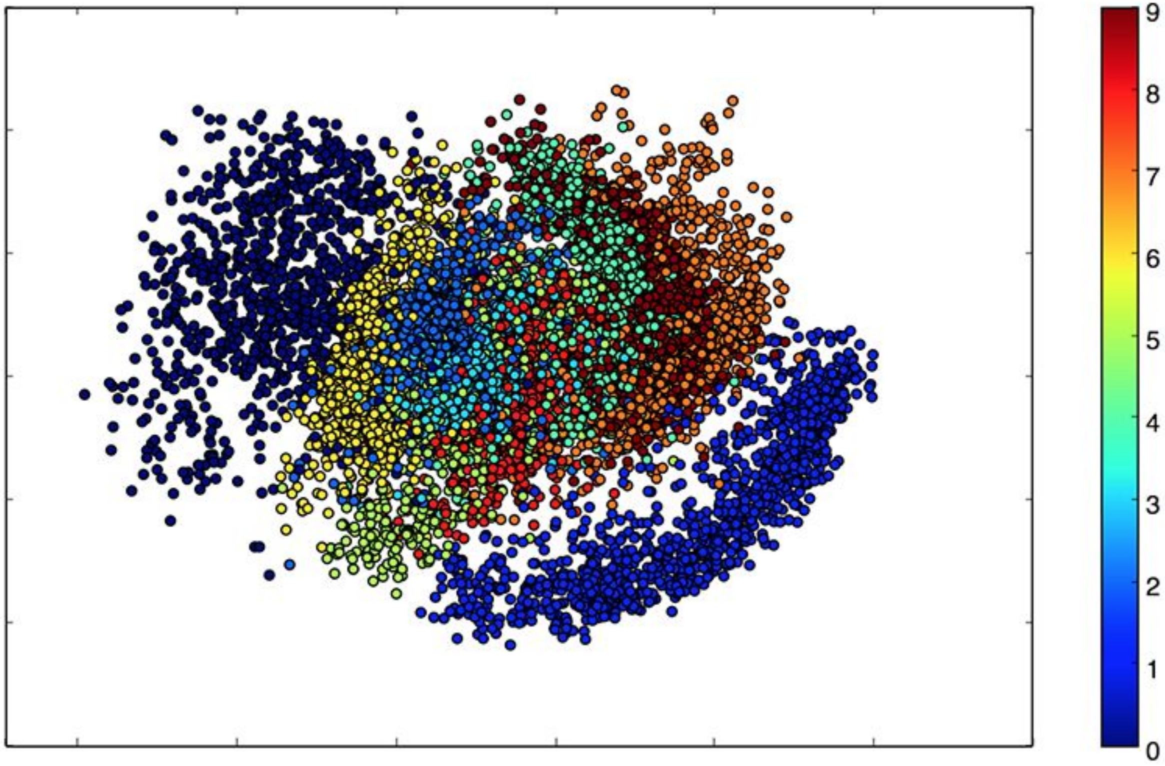
Variational Autoencoders - KL error



[source]

Variational Autoencoders - Latent space

- 2D latent space (**not t-SNE**) [Animated GIF]

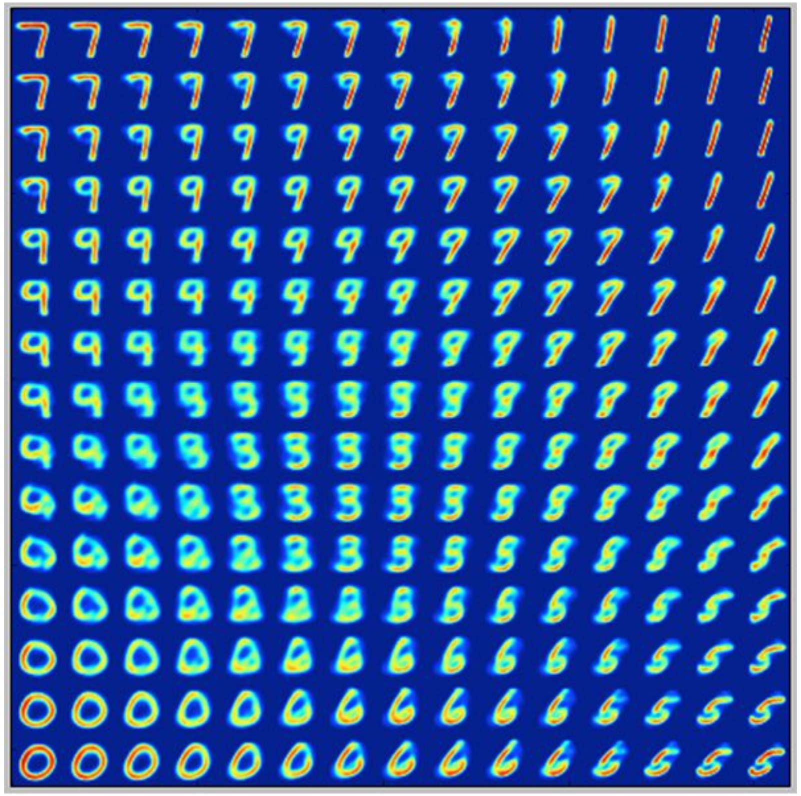


[source]

Variational Autoencoders - Latent space reconstruction

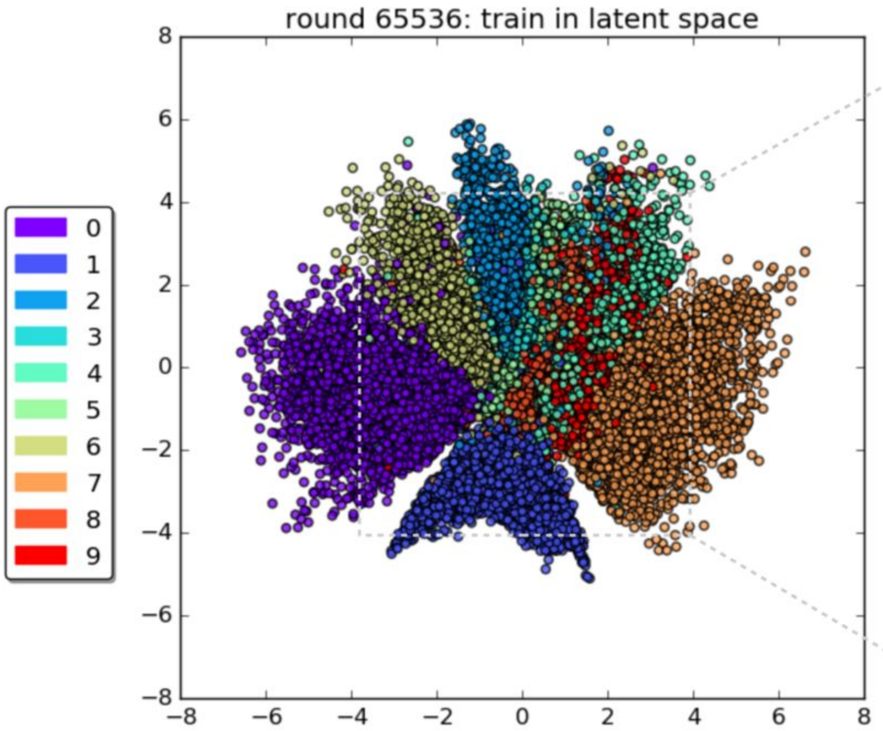
- Reconstruction from latent coordinates into the original data space.

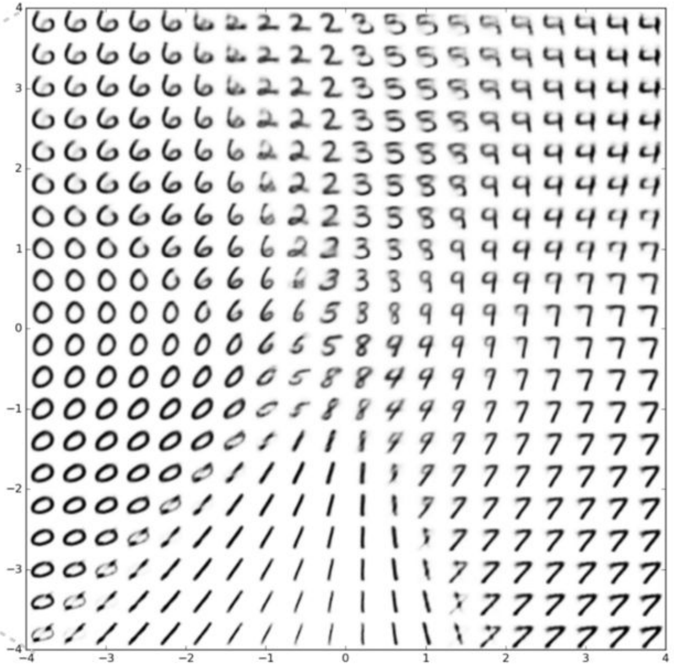
[Animated GIF]



Variational Autoencoders - Latent space reconstruction

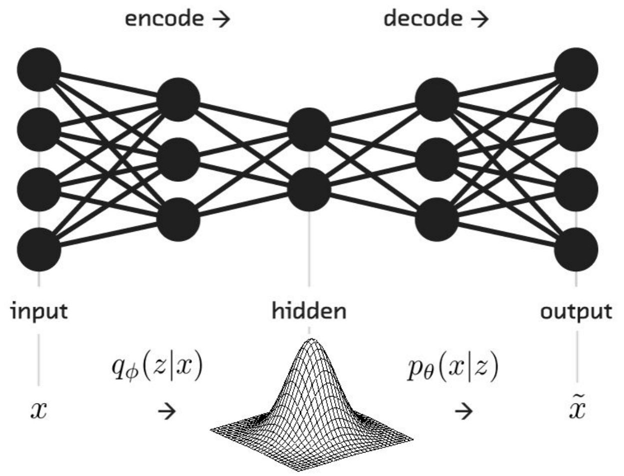
[Animated GIF]

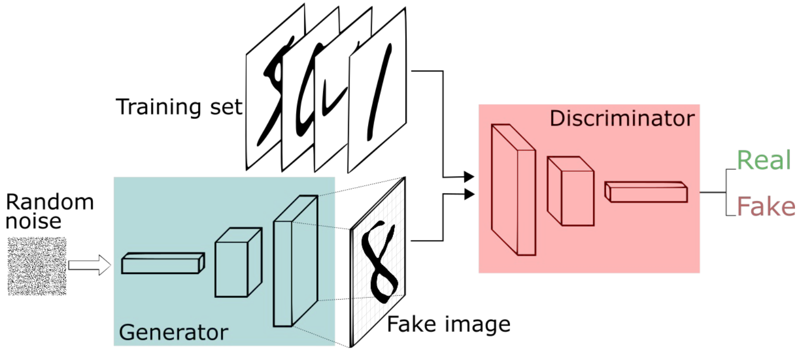




VAEs vs **GANs**

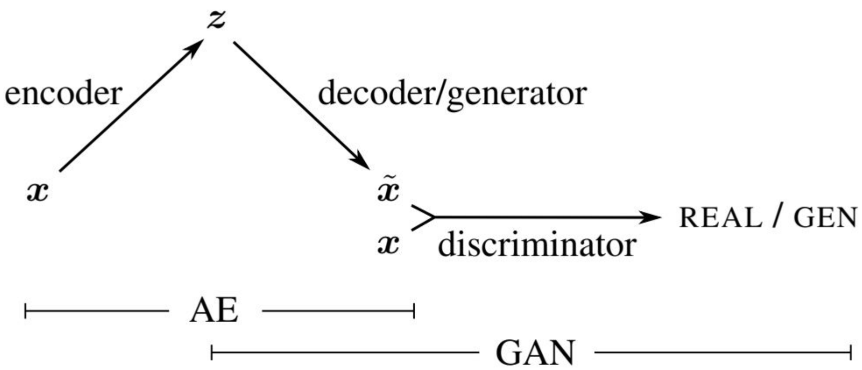
- Variational Autoencoders (VAE) are good at learning representations (interpretable dimensions, possibility to set complex priors).
- Generative Adversarial Networks (GANs) are good at generating new samples (clever loss). Trickier to train.

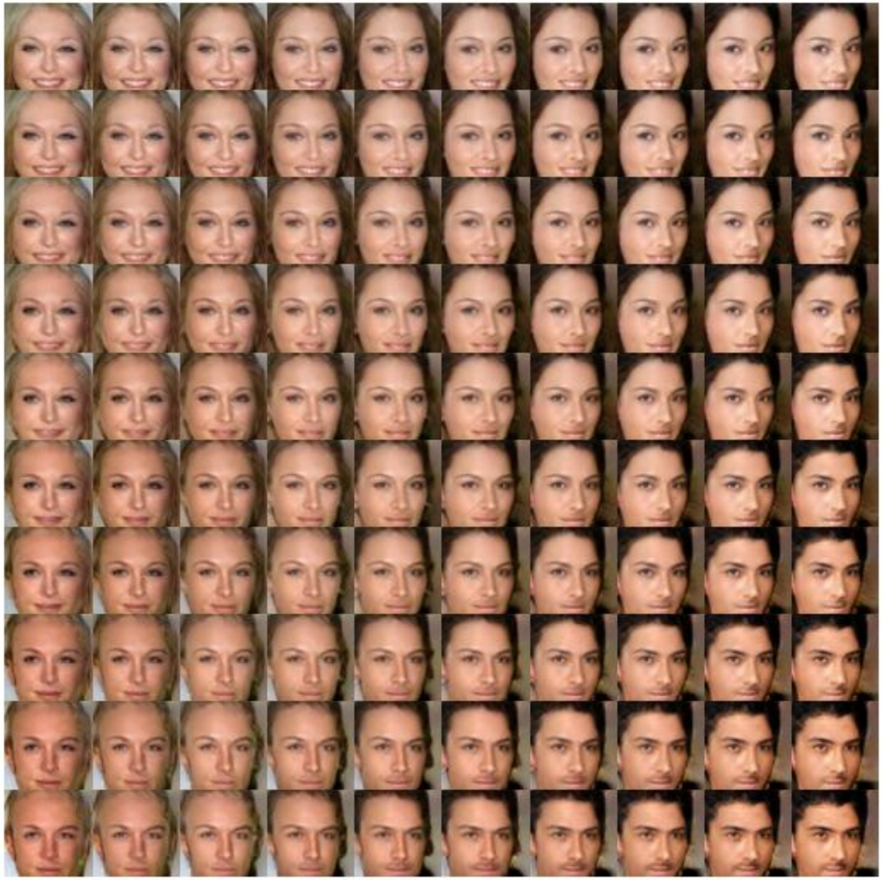




VAEs vs **GANs**

- They can be combined to get the better of both worlds.





source

Summary of applications

- Dimensionality reduction (and clustering if we apply k-means for example after the reduction).
- Data denoising
- Data generation (VAE)

More info

- Understanding Variational Autoencoders (VAEs)
- Building Autoencoders in Keras
- Variational Autoencoders Part 1 + Part 2