Autoencoders

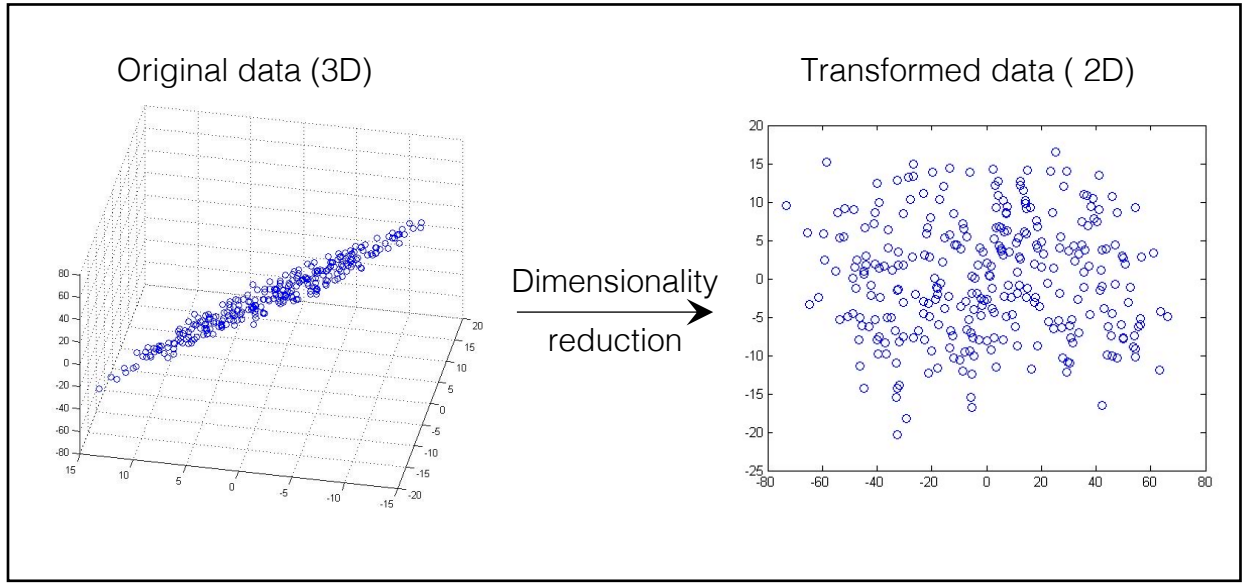
Nezar Abdennur

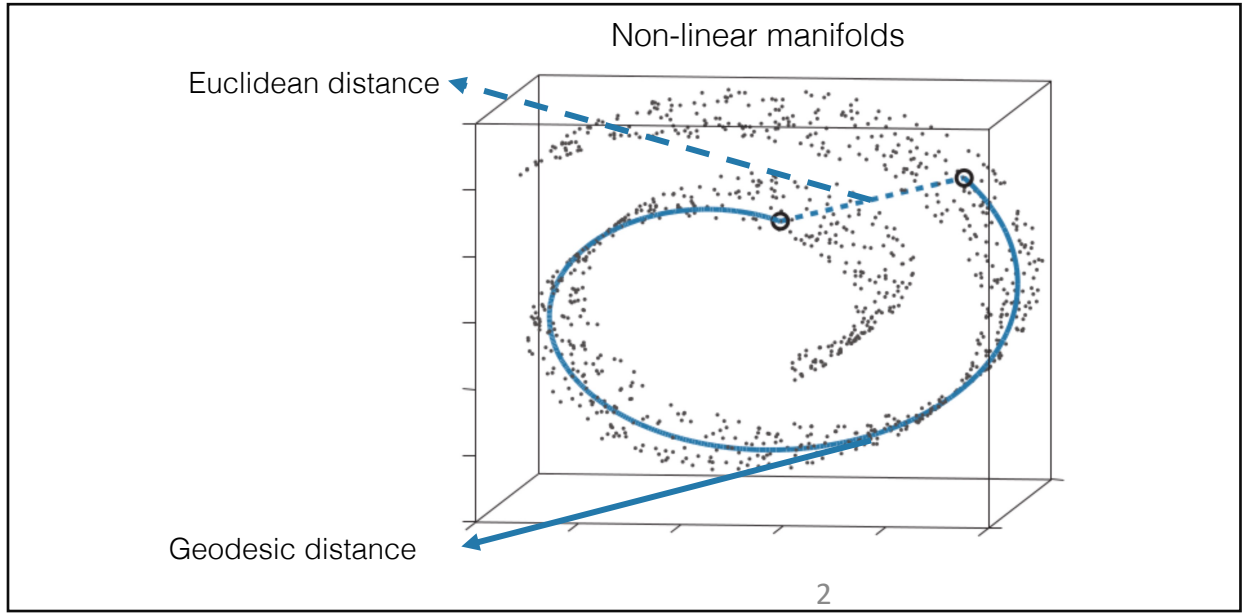
BBS 741

Review: dimensionality reduction

Response or Output (Y)	Available	Not available	Partially available
	Supervised	**Unsupervised**	**Semi-supervised**
Discrete / Categorical	Classification	Clustering	**Unsupervised techniques** to learn structure of data, **supervised techniques** to make best guess predictions
Continuous	Regression	Dimensionality reduction	
	Find function f: X → Y	Infer the underlying structure of X	

Adapted from An Introduction to Statistical Learning



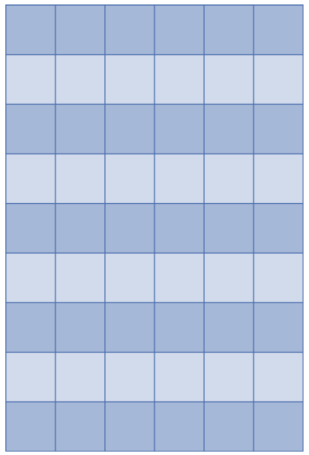


PCA: linear, parametric

`fit`

Features (high-dim: p)

n points

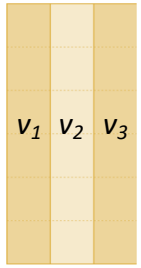


`(n, p)`

X

mean-centered data

`eig(XᵀX)`
or
`svd(X)`

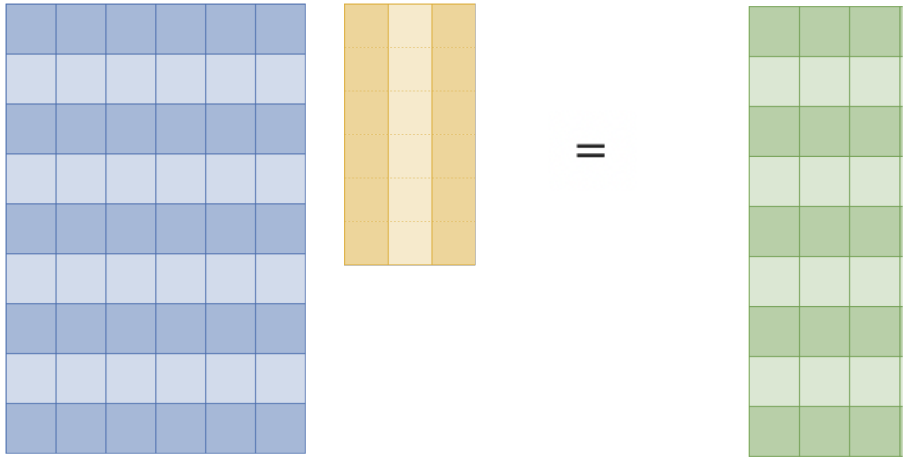


`(p, q)`

V

First q PC loading vectors

`transform`



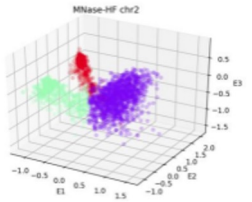
`(n, p)` `(p, q)` `(n, q)`

X V Z

Projected data (low-dim: q)

PC Scores

Loss: L2 projection error
Opt. strategy: Matrix factorization

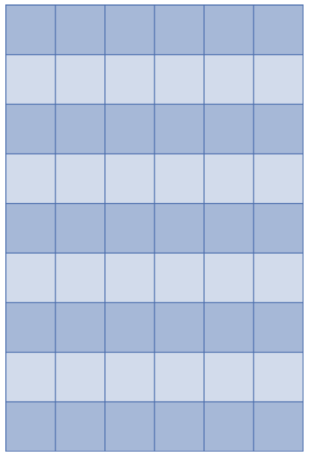


PCA: linear, parametric

`fit`

Features (high-dim: p)

n points

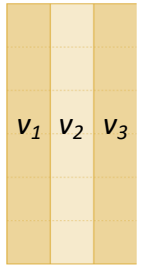


`(n, p)`

X

mean-centered data

`eig(XᵀX)`
or
`svd(X)`



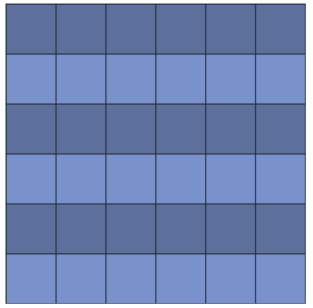
`(p, q)`

V

First q PC loading vectors

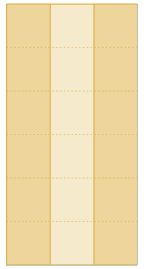
`predict`

New data



`(n, p)`

X'

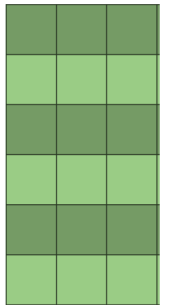


`(p, q)`

V

=

Projected data (low-dim: q)

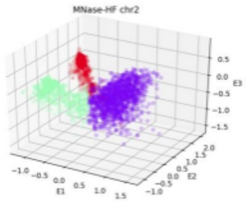


`(n, q)`

Z'

PC Scores

Loss: L2 projection error
Opt. strategy: Matrix factorization

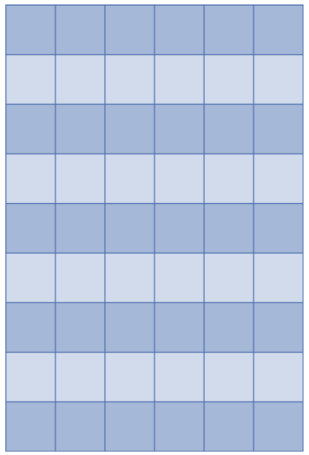


PCA: linear, parametric

`fit`

Features (high-dim: p)

n points

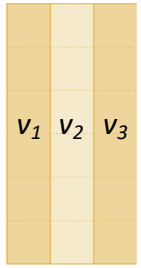


`(n, p)`

X

mean-centered data

`eig(XᵀX)`
or
`svd(X)`



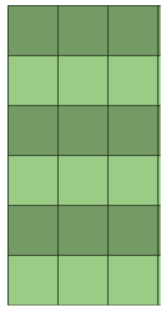
`(p, q)`

V

First q PC loading vectors

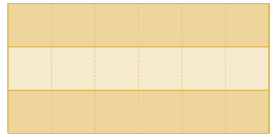
`reconstruct`

Projected data



`(n, q)`

Z'



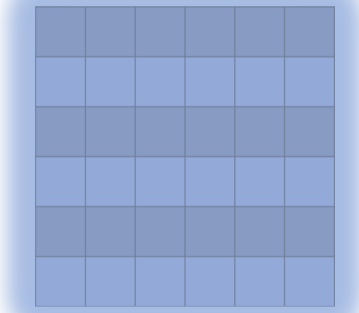
`(q, p)`

V^T

PC Scores

=

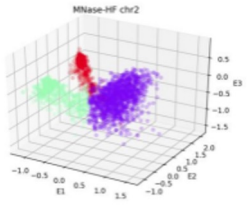
Reconstructed data



`(n, p)`

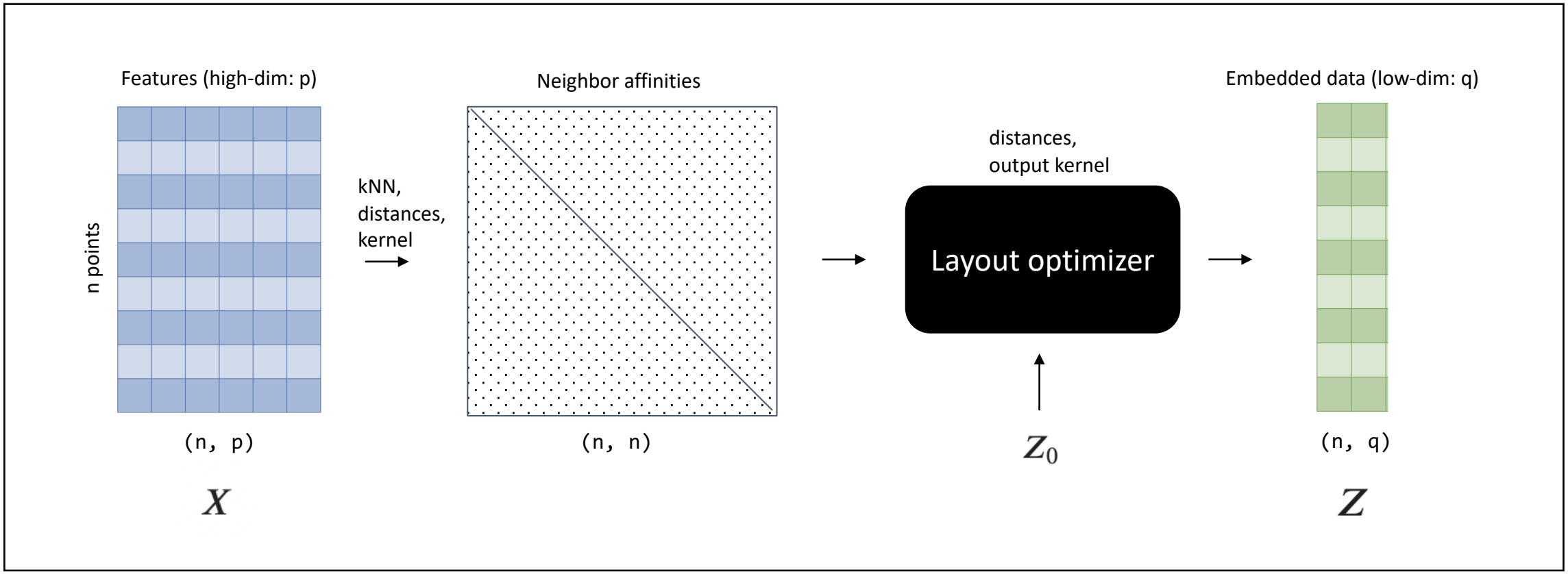
X'

Loss: L2 projection error
Opt. strategy: Matrix factorization

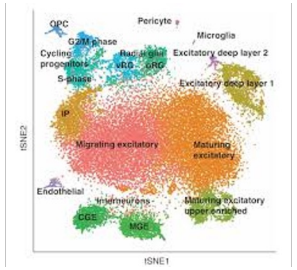


Neighbor embedding: non-linear, non-parametric

`fit & transform`

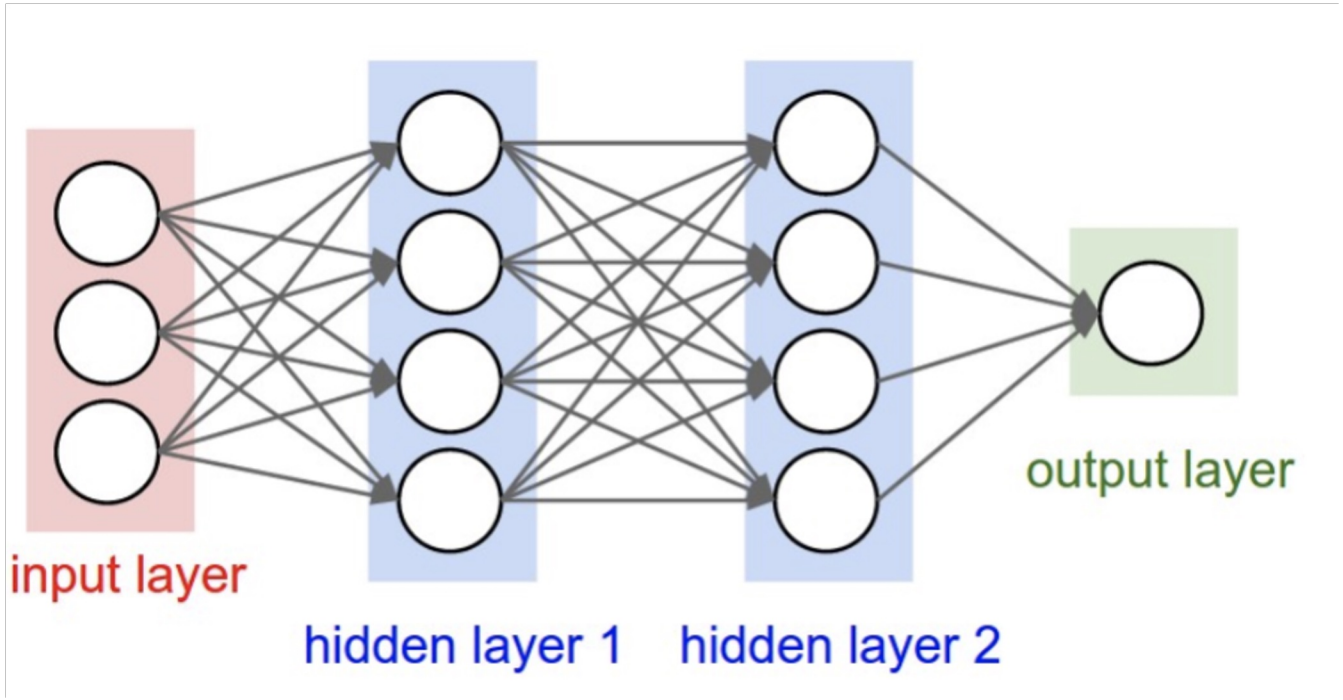


Loss: KL div or cross-entropy between input & output affinities
Opt. strategy: Gradient Descent



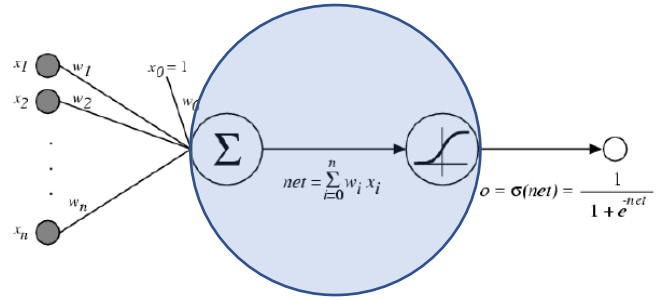
ANNs: non-linear, parametric

- Feedforward NNs (MLPs) are universal function approximators
- Hidden layers are feature extractors
- Typically used for <u>supervised</u> learning: target = labeled output



Parameters: weights (and biases)
Depth: # of hidden layers
Width: # of units per hidden layer

Type of **activation function** (non-linearity)

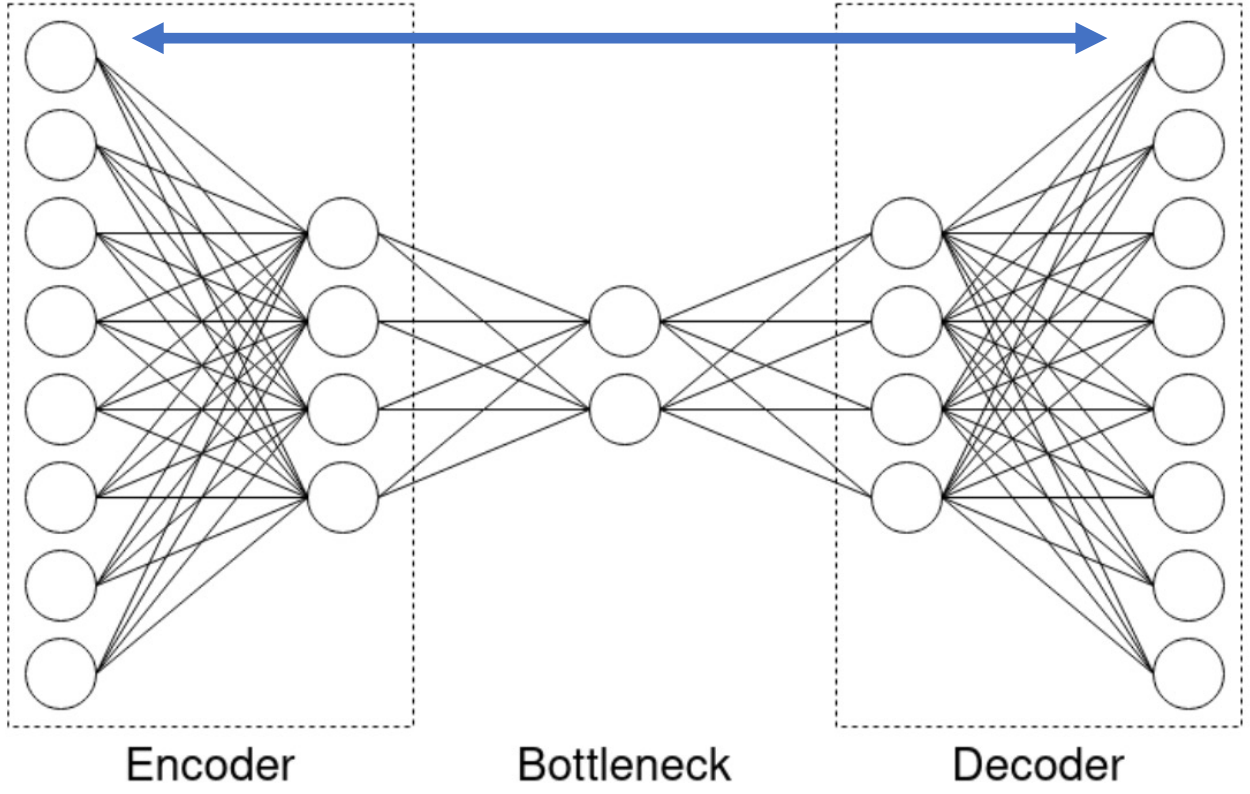


Loss: Various forms, depending on problem
Opt. strategy: Gradient Descent
Opt. method: (Stochastic) Gradient Descent by Backprop

Regularization of the loss function to prevent overfitting

NNs for “unsupervised” learning

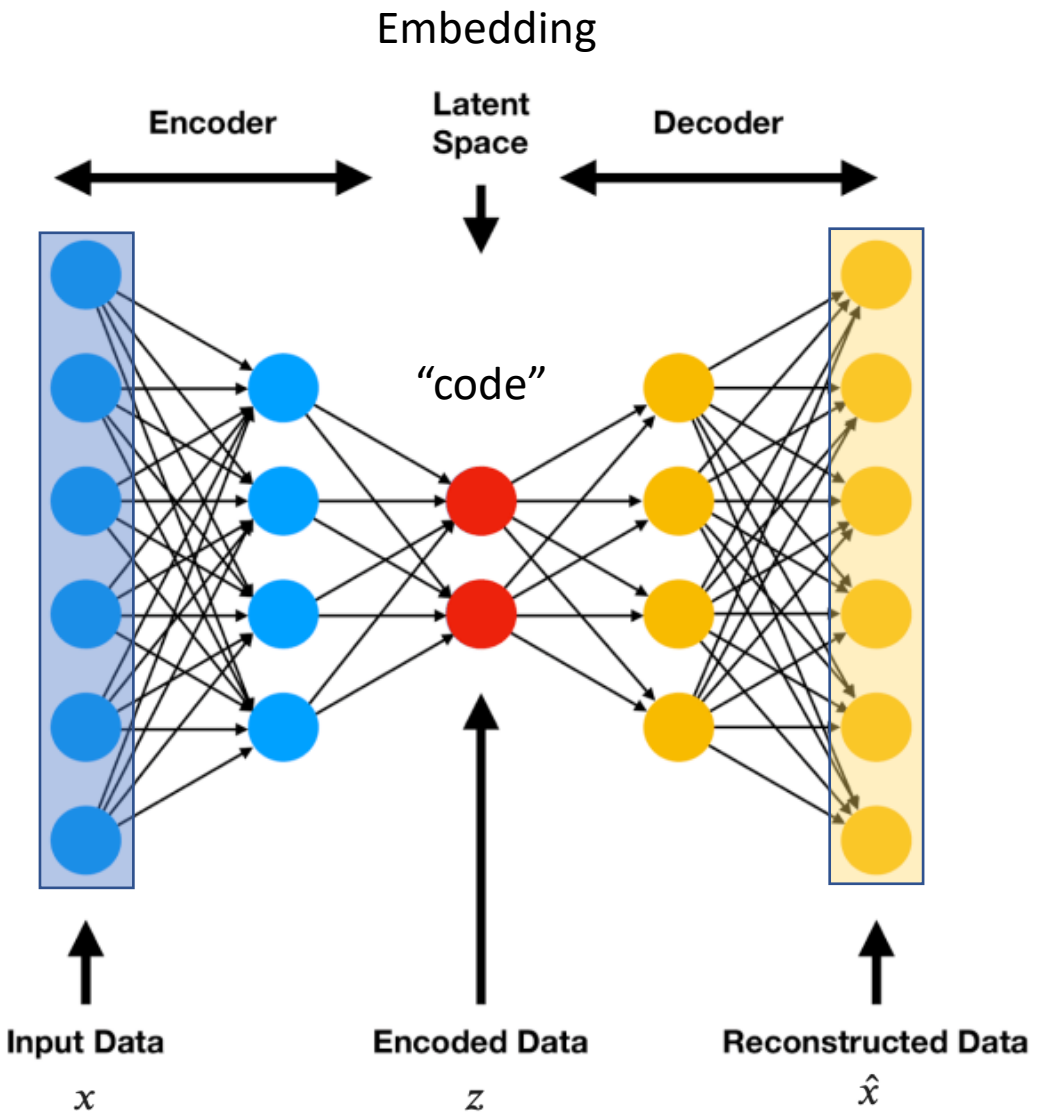
- Make the target identical to the input!
- The objective is now for the network to reconstruct the input.
- Technically, this is not unsupervised but ***self-supervised*** learning.



Alberts, K.J. (2021) http://essay.utwente.nl/85686/

Autoencoders

Encoder:
Reduce dimensionality.
Map input data to a “code” in a **compressed, latent space**.



Decoder:
Reconstruct data from latent representation.
Interpolate or generate synthetic examples from unseen codes.

https://www.compthree.com/blog/autoencoder/

Autoencoders

Example with 1 hidden layer (z)

- Given data x (no labels) we would like to learn the functions f (encoder) and g (decoder) where:

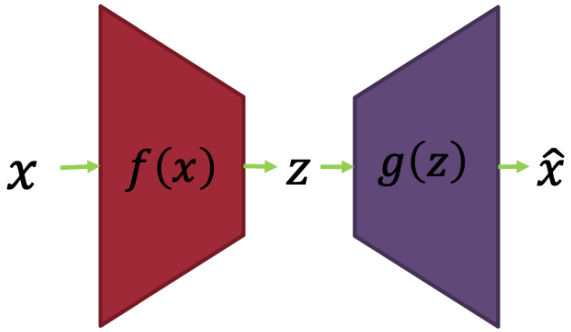
$$f(x) = s(wx + b) = z$$

and

$$g(z) = s(w'z + b') = \hat{x}$$

s.t $h(x) = g(f(x)) = \hat{x}$

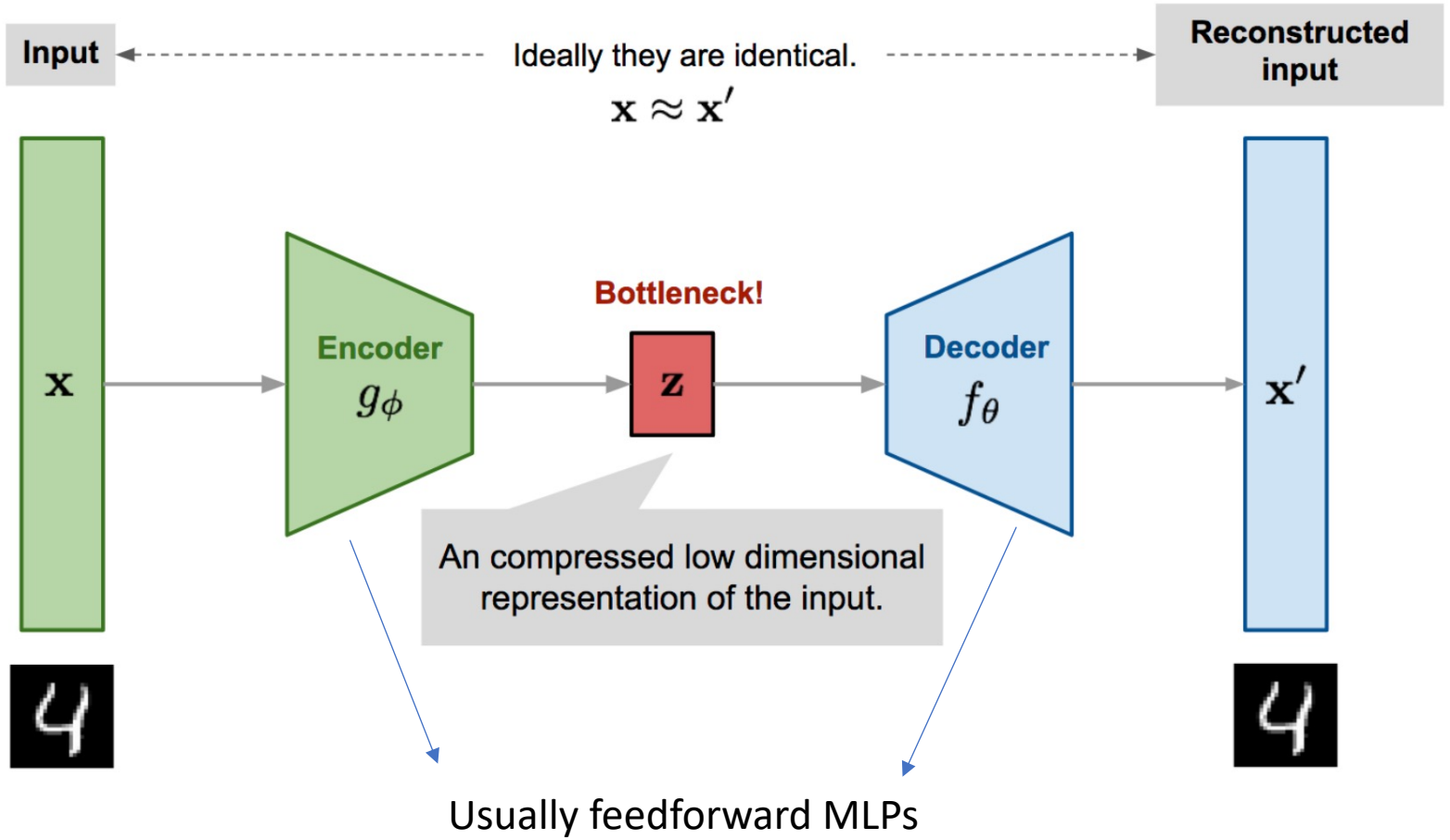
where h is an **approximation** of the identity function.



(z is some **latent** representation or **code** and s is a non-linearity such as the sigmoid)

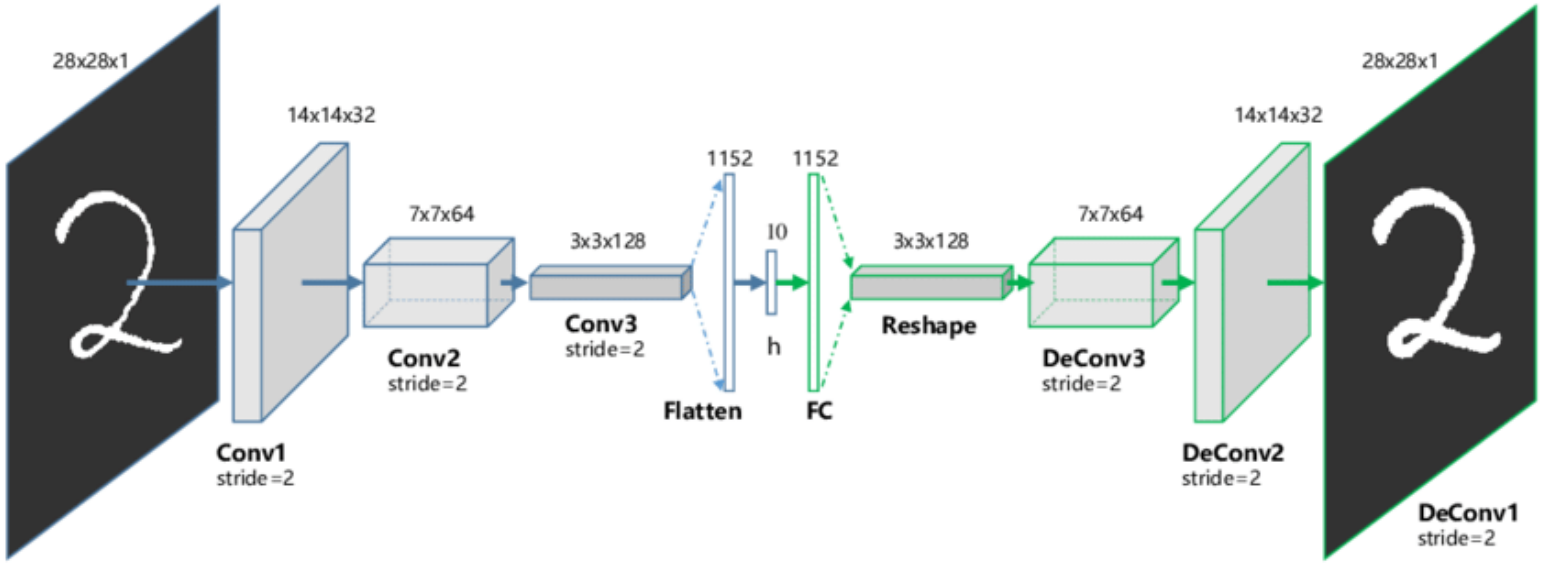
($\hat{x}$ is x's reconstruction)

Autoencoders



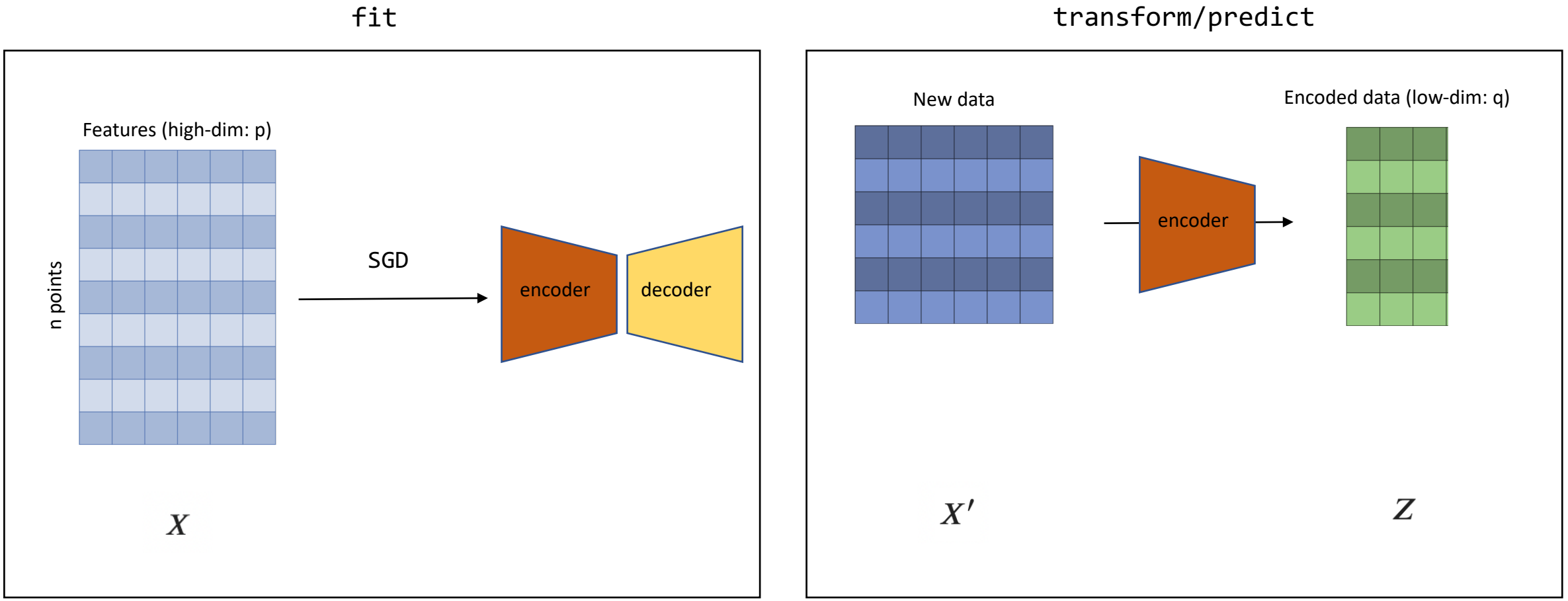
https://lilianweng.github.io/posts/2018-08-12-vae/

Autoencoders

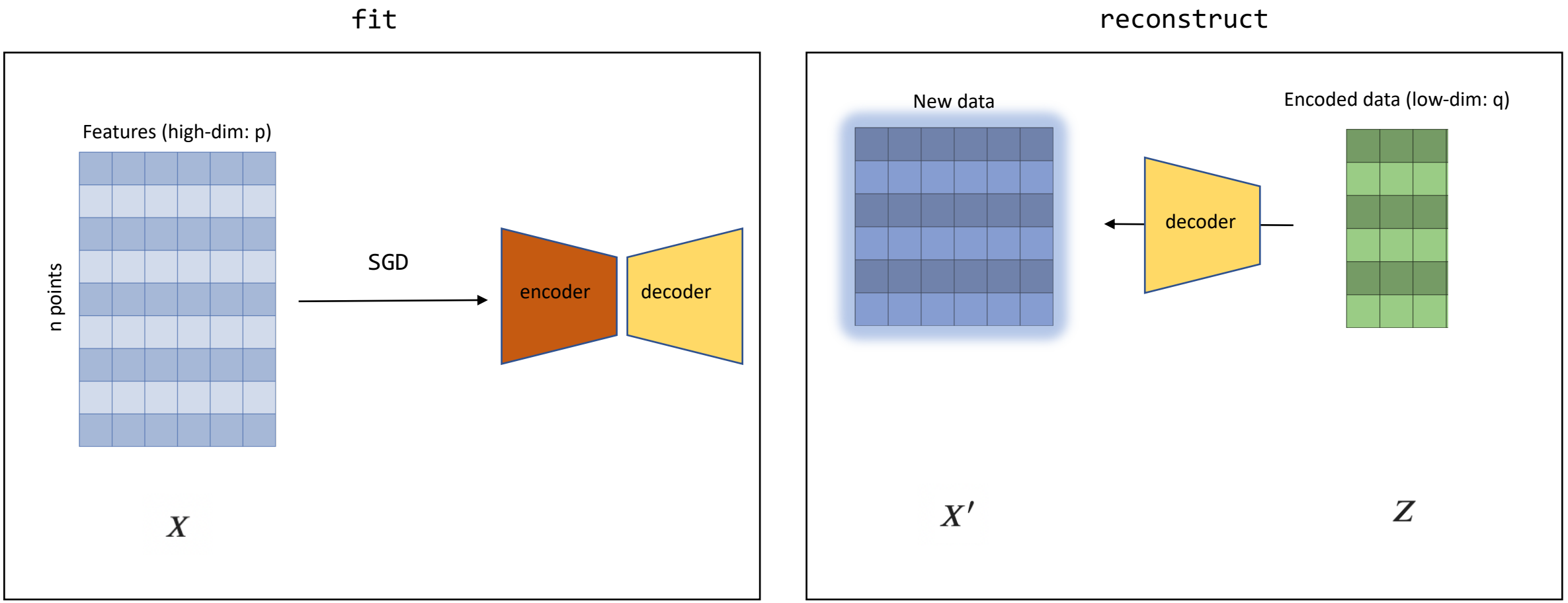


Guo et al, 2017. doi: 10.1007/978-3-319-70096-0_39

Non-linear, parametric



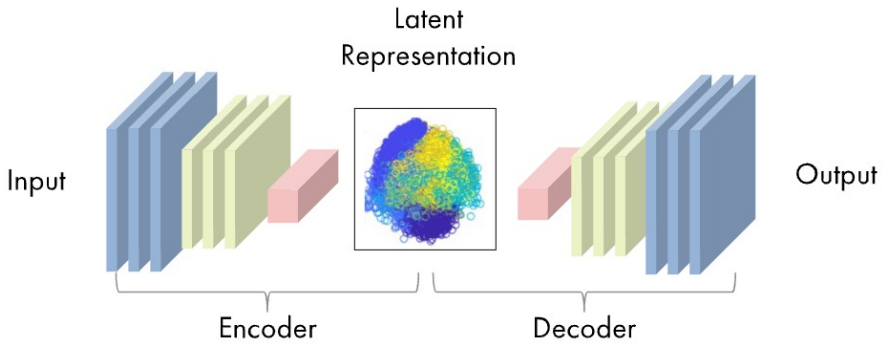
Non-linear, parametric



Uses for autoencoders

*Not good general-purpose data compressors

- Part of NN history for decades (LeCun et al., 1987)
- Representation learning
  - dimensionality reduction
  - manifold learning and latent space modeling
  - self-supervised feature extraction
  - representations consumed by humans or to improve ML!
- Anomaly detection
- Information retrieval, semantic hashing
- Signal denoising, image in-painting
- The connections to representation learning have made autoencoders an important and powerful framework for **generative modeling**

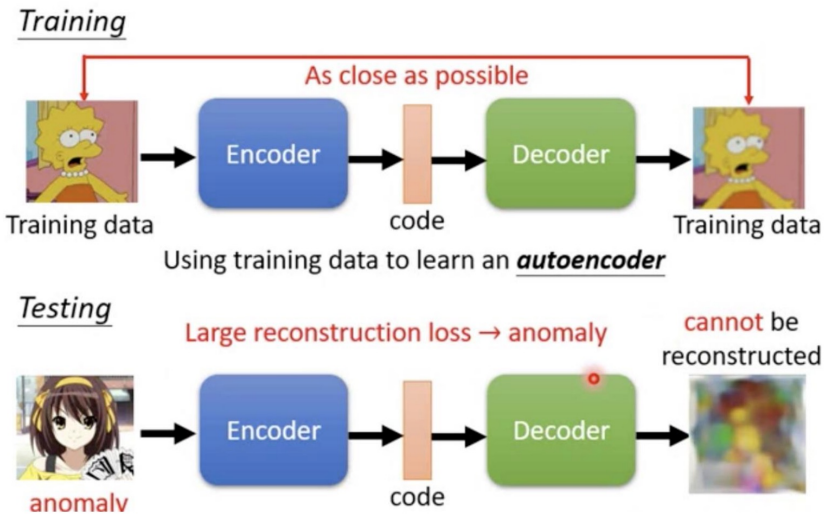


https://www.mathworks.com/discovery/autoencoder.html

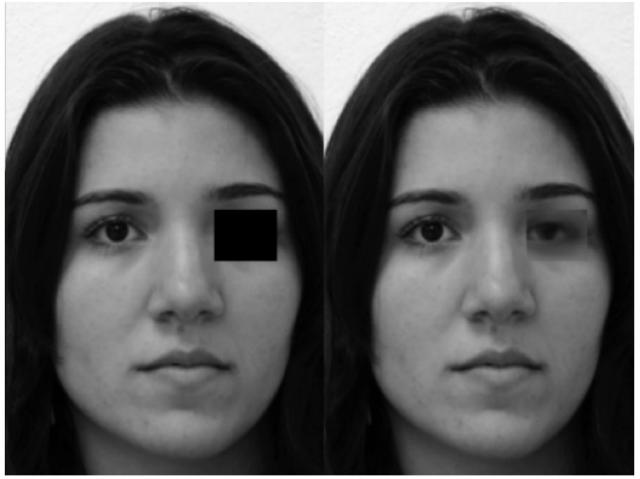
Uses for autoencoders

*Not good general-purpose data compressors

- Part of NN history for decades (LeCun et al., 1987)
- Representation learning
  - dimensionality reduction
  - manifold learning and latent space modeling
  - self-supervised feature extraction
  - representations consumed by humans or to improve ML!
- Anomaly detection
- Information retrieval, semantic hashing
- Signal denoising, image in-painting
- The connections to representation learning have made autoencoders an important and powerful framework for **generative modeling**



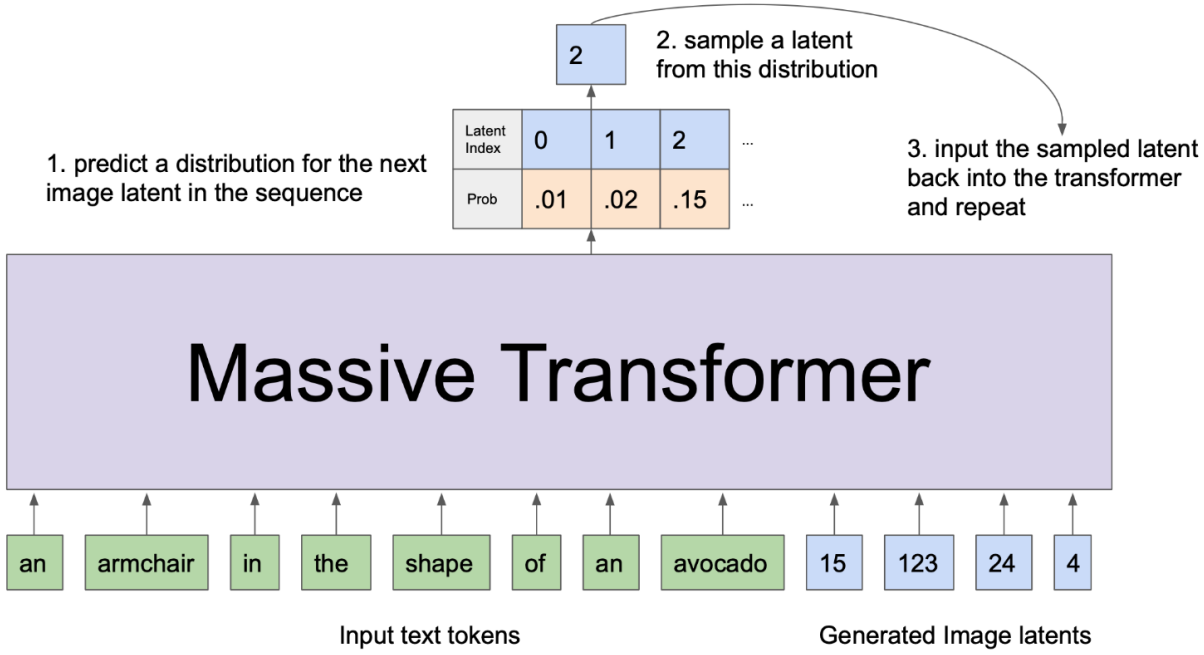
http://www.wcoomd.org/-/media/wco/public/global/pdf/events/2019/picard/data-analytics1_muzhi-guan.pdf

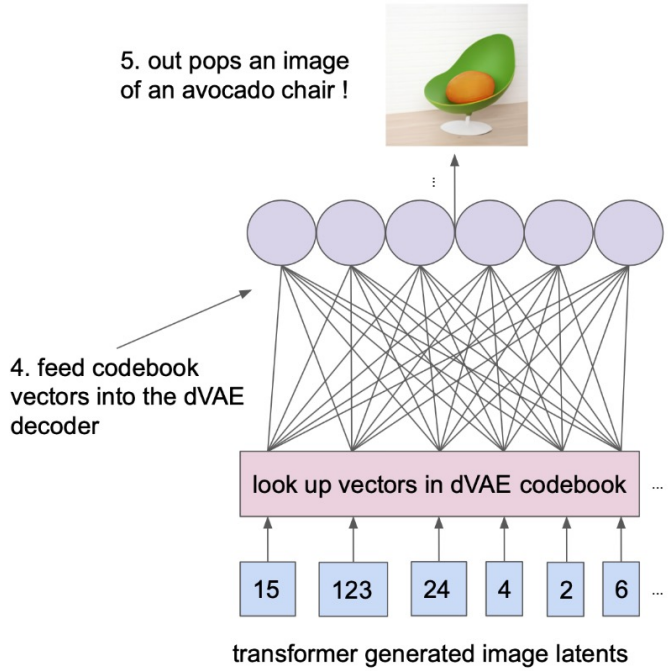


Jacques et al. (2017) doi: 10.1109/ACII.2017.8273601

Latent representations are critical components of powerful AI

DALL-E text-to-image generator depends on a specialized autoencoder image code





Two components:

1) Discrete autoencoder that learns to represent images in a compressed latent space

2) “Transformer” learns correlations between language and discrete image code

https://ml.berkeley.edu/blog/posts/dalle2/

Model capacity and generalizability

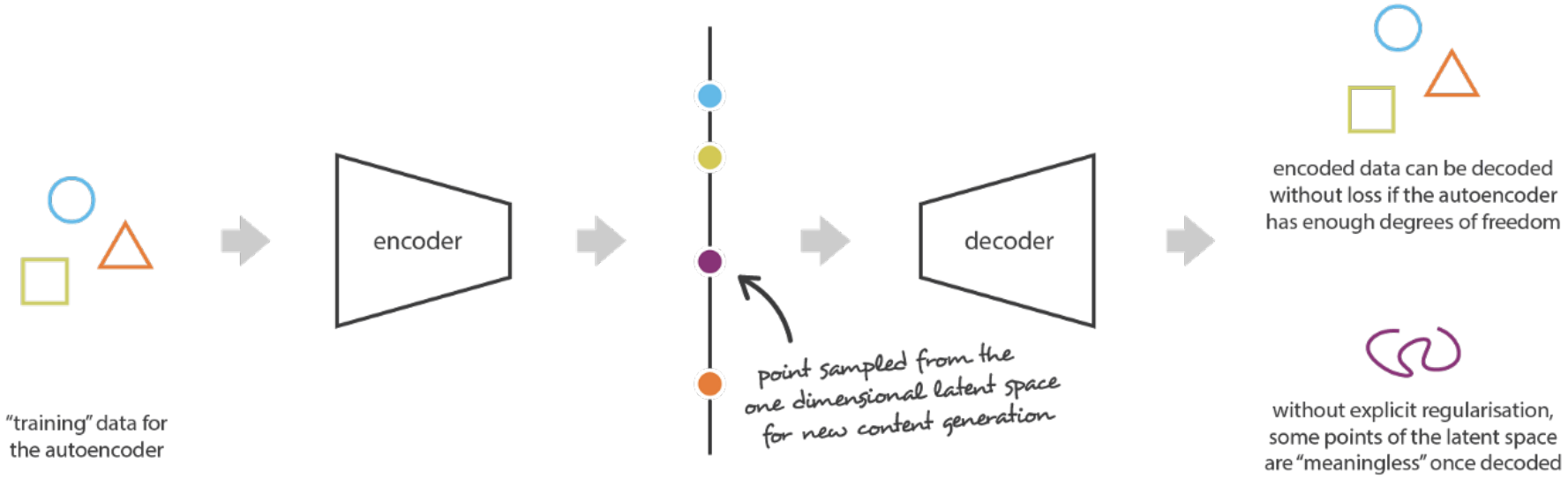
An ideal autoencoder is

- Sensitive enough to the inputs to accurately build a reconstruction.
- Insensitive enough that the model doesn't simply memorize or overfit the training data.

i.e., extracts useful information and sheds redundancies to learn a generalizable encoding and decoding.

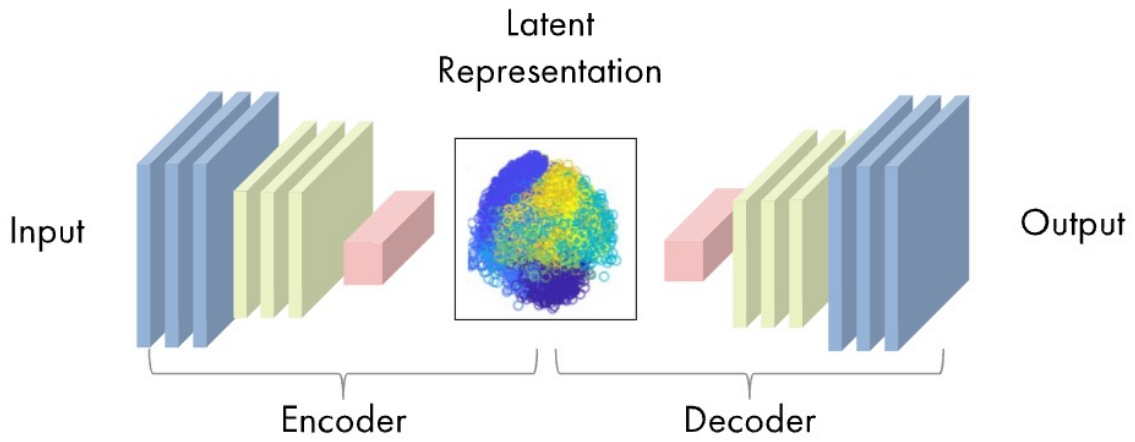
Model capacity and generalizability

With enough capacity, it's possible for an AE with a code layer of even a single node (1D encoding) to memorize or "index" the training data set without learning any generalizable structure in the data.



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

Controlling the balance



$$\mathcal{L}(x, \hat{x}) + regularizer$$

Reconstruction loss	**Regularization penalty**
Encourage model to be sensitive to inputs	Prevent memorization/overfitting of the inputs

You can go deep

- Universal approximator theorem guarantees that a feedforward NN with at least 1 hidden layer can represent an approximation of any function (within a broad class) to an arbitrary degree of accuracy, provided that it has enough nodes.

- With enough nodes, even 1 hidden layer is able to represent the identity function for a given data set arbitrarily well.

- However, the mapping from input to code is shallow, which which limits the ability to enforce constraints to help extract useful features, even with regularization.

You can go deep

- In many circumstances, deeper models can reduce the number of units required to represent the desired function and can reduce the amount of generalization error (vs wider models).
- Encoder and Decoder are both feedforward NNs so both can benefit from depth.
- Beware that, in general, adding hidden units and layers increases the information capacity of the NN to memorize the data!

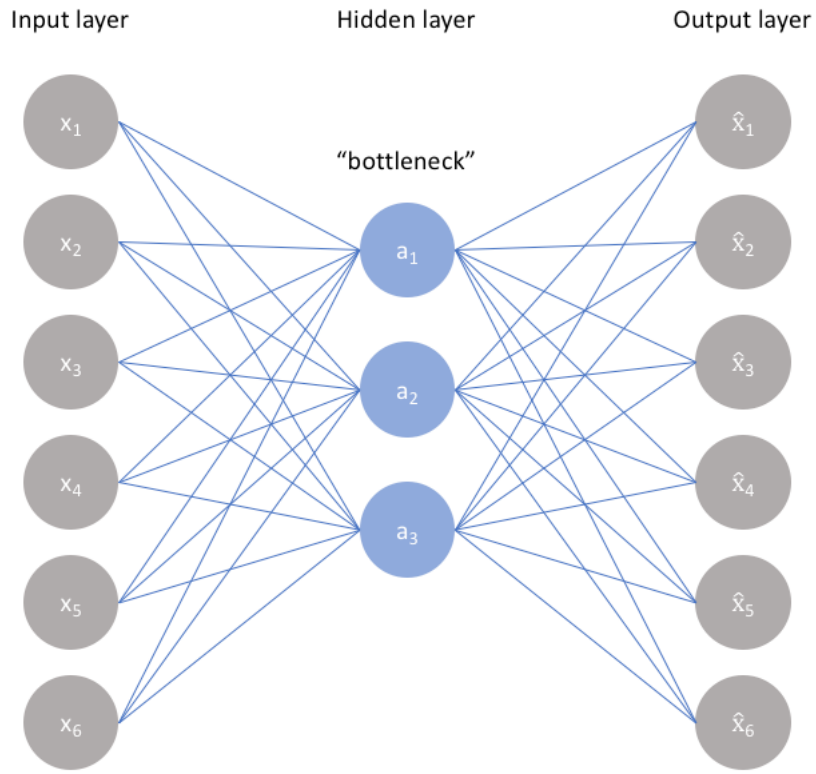
  → Need regularization

Autoencoder architectures

- Undercomplete
- Sparse
- Denoising
- Contractive
- Variational

Undercomplete Autoencoders

- Impose an explicit bottleneck by limiting width in the hidden layers.
- This forces a **compressed** knowledge representation of the input.
- When forcing input through this bottleneck, structures in the data (e.g. correlations between features) can be learned.
- No explicit regularization term in loss function. Loss = reconstruction error.



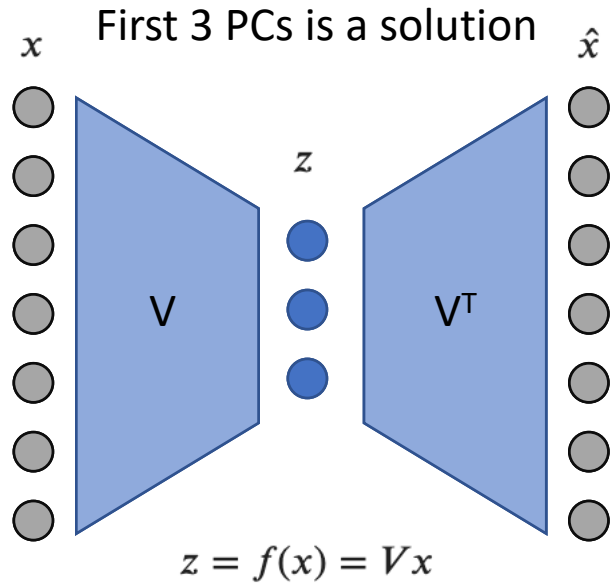
$$\mathcal{L}(x, \hat{x})$$

Reconstruction Error
(e.g. MSE or binary cross-entropy)

Undercomplete Autoencoders

Linear AEs find the same representation as PCA!

- An undercomplete autoencoder:
  - Without activation functions (i.e., linear)
  - With mean squared error reconstruction loss

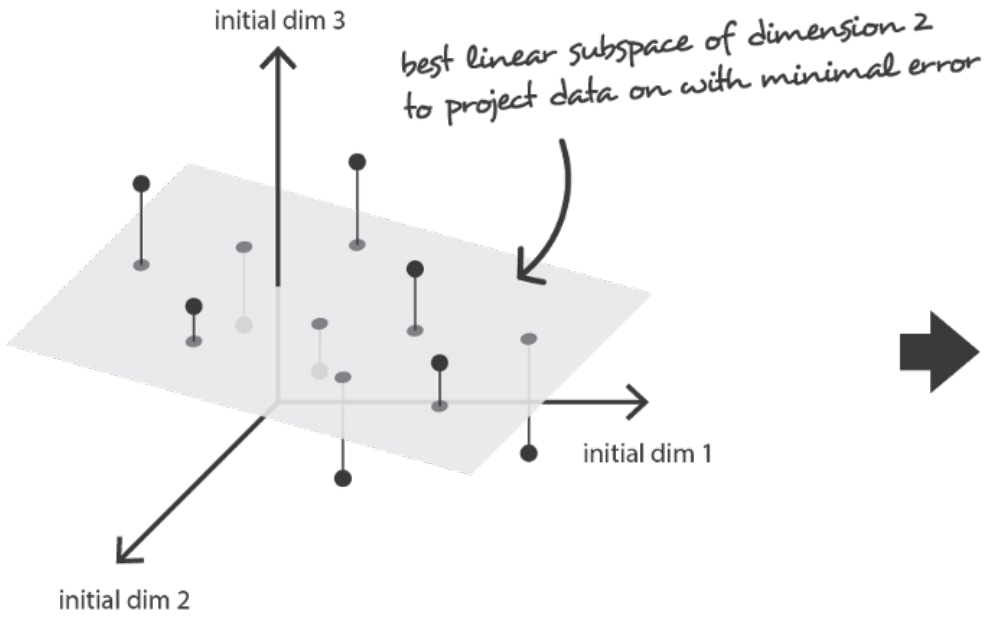


- Learns the exact same linear subspace as PCA!
- The actual basis given by the weights of the encoder will not, in general, correspond to the principal directions or even be orthogonal, but they will span the same subspace.

Kunin, Daniel, et al. (2019). “Loss Landscapes of Regularized Linear Autoencoders” PMLR

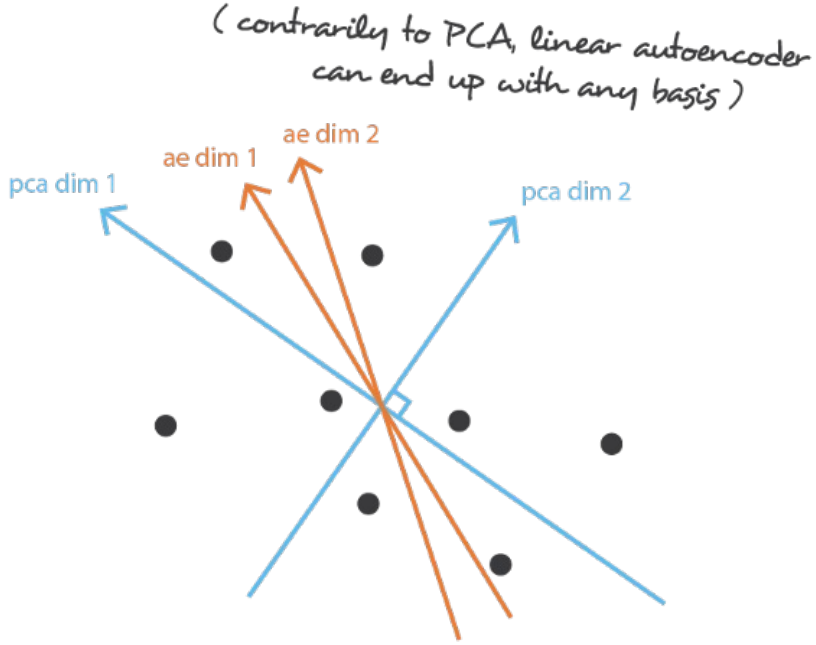
Undercomplete Autoencoders

Linear AEs find the same representation as PCA!



Data in the full initial space

In order to reduce dimensionality, PCA and linear autoencoder target, in theory, the same optimal subspace to project data on...



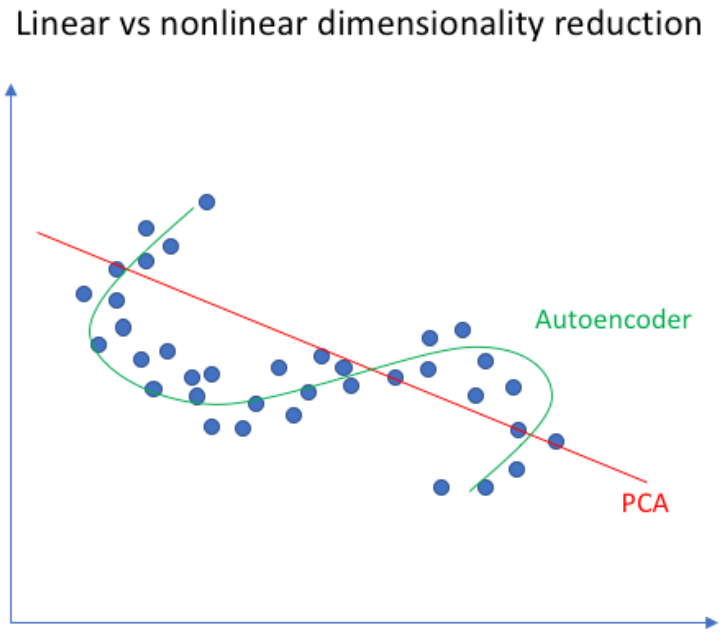
Data projected on the best linear subspace

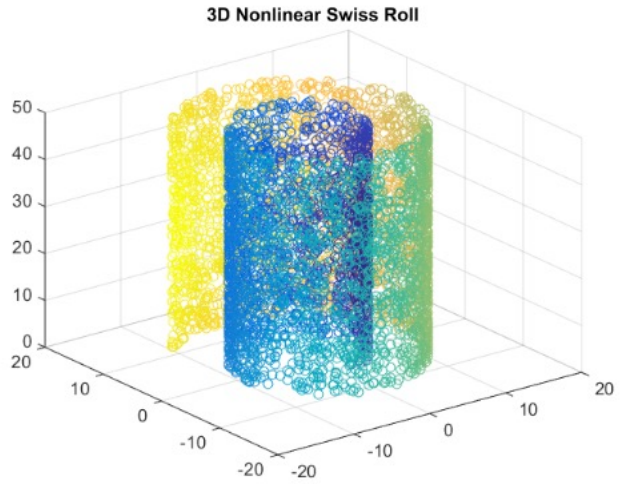
... but not necessarily with the same basis due to different constraints (in PCA the first component is the one that explains the maximum of variance and components are orthogonal)

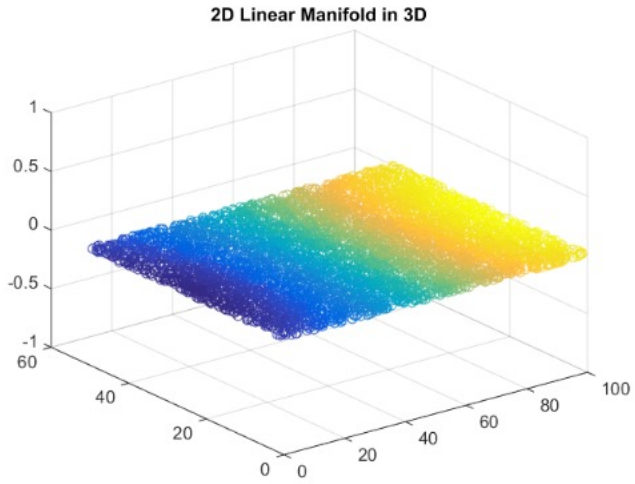
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Undercomplete Autoencoders

- Activation functions allow autoencoders to learn non-linear manifolds!





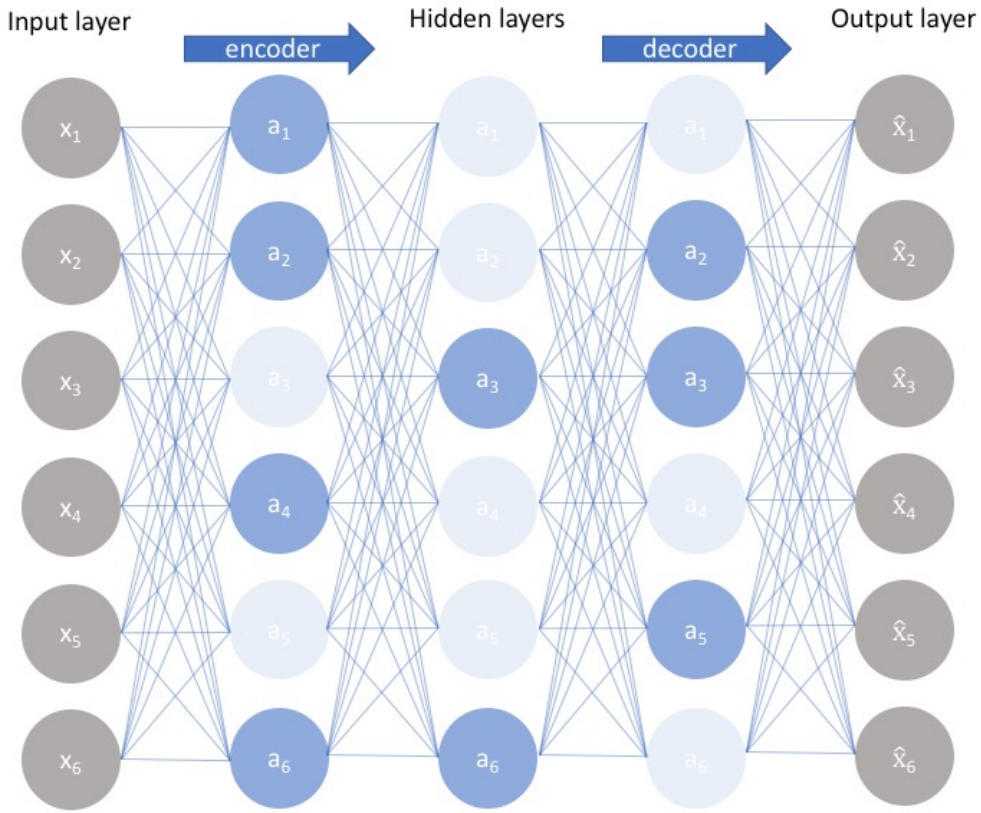


https://www.jeremyjordan.me/autoencoders/

Sparse Autoencoders

Instead of removing nodes from our hidden layers, we can construct a loss function that penalizes **activations** within a layer.

Note: This is NOT the same as penalizing the weights!



$$\mathcal{L}(x, \hat{x}) + \lambda \sum_i \left| a_i^{(h)} \right|$$

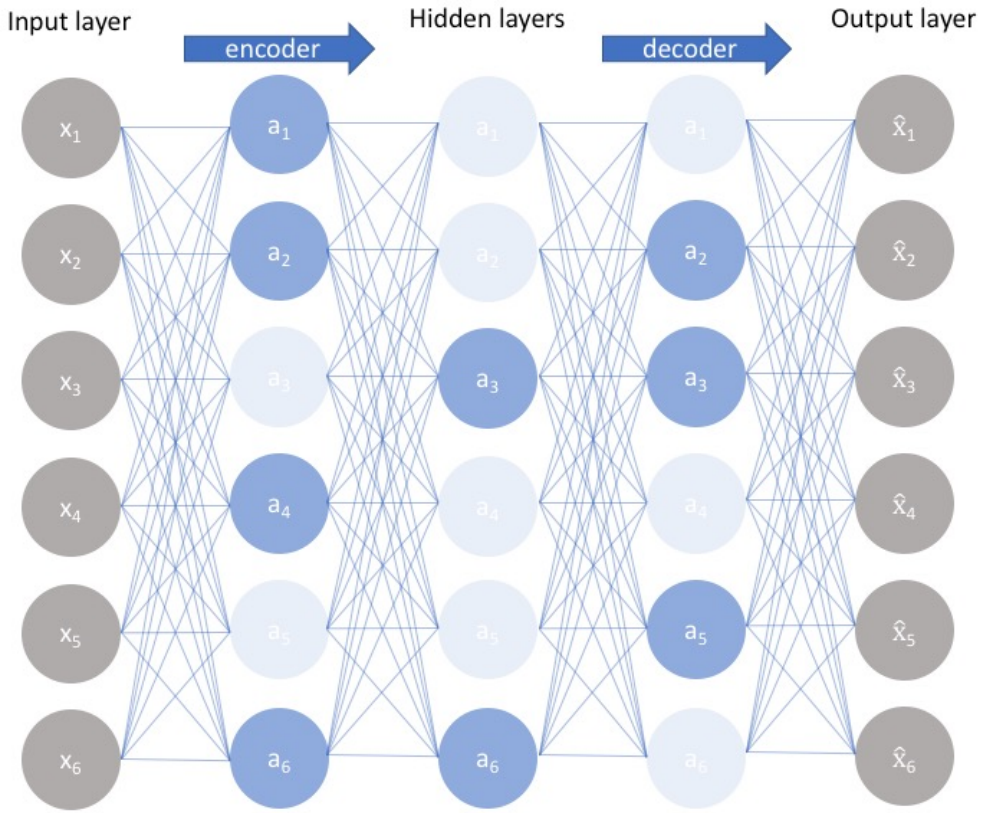
L1 regularization penalty on the magnitude of activations in hidden layers

Sparse Autoencoders

For any training input example, we encourage the network to activate only a small number of neurons.

Different inputs will activate different nodes throughout the network: *activations are data-dependent*.

This allows the network to sensitize itself to properties of the input features while limiting its ability to memorize the input data.



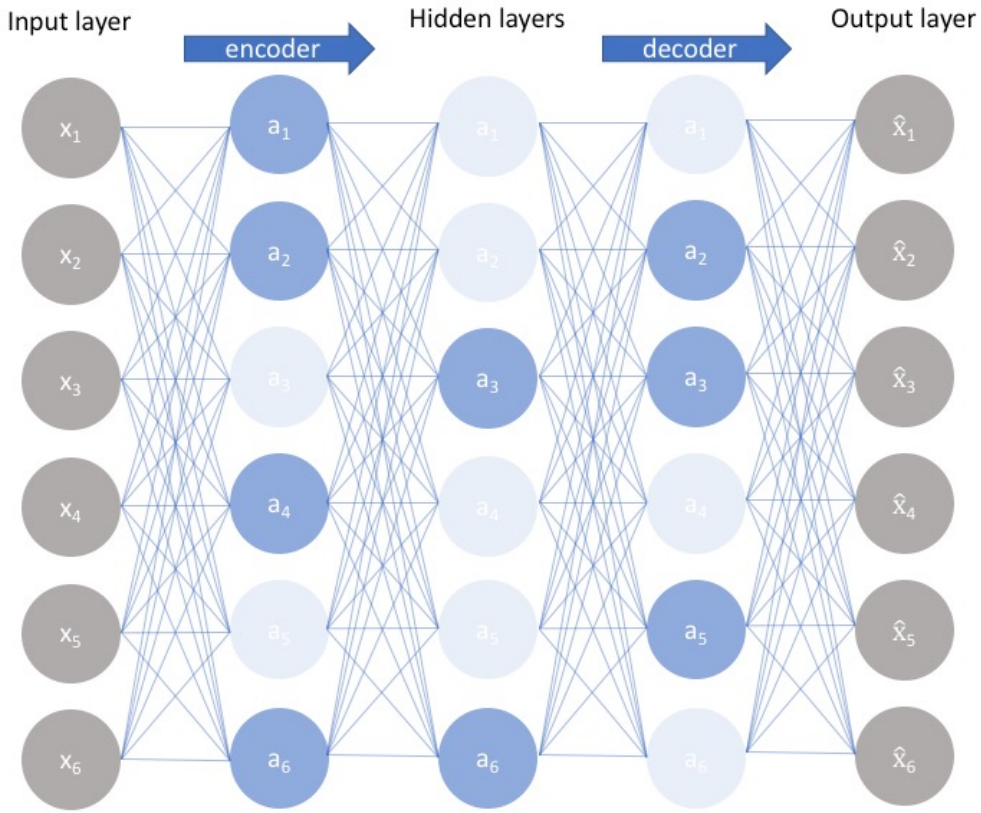
$$\mathcal{L}(x, \hat{x}) + \lambda \sum_i \left| a_i^{(h)} \right|$$

L1 regularization penalty on the magnitude of activations in hidden layers

Sparse Autoencoders

By penalizing activations instead of weights or number of hidden nodes, we are also able to separate latent state representation from regularization.

i.e., we can choose an encoding dimensionality suitable for the task at hand, and independently tune the regularization hyperparameter to control overfitting.



$$\mathcal{L}(x, \hat{x}) + \lambda \sum_i \left| a_i^{(h)} \right|$$

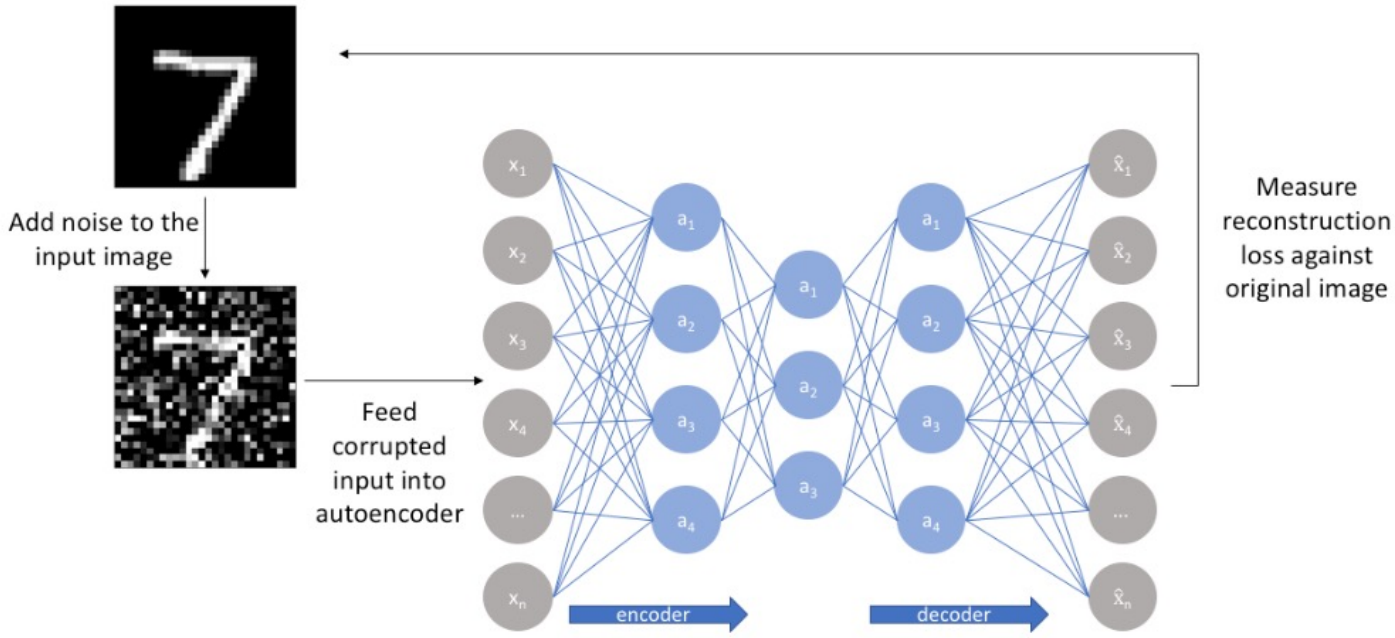
L1 regularization penalty on the magnitude of activations in hidden layers

Denoising Autoencoders

Slightly corrupt the input data with noise and use the clean data as target output.

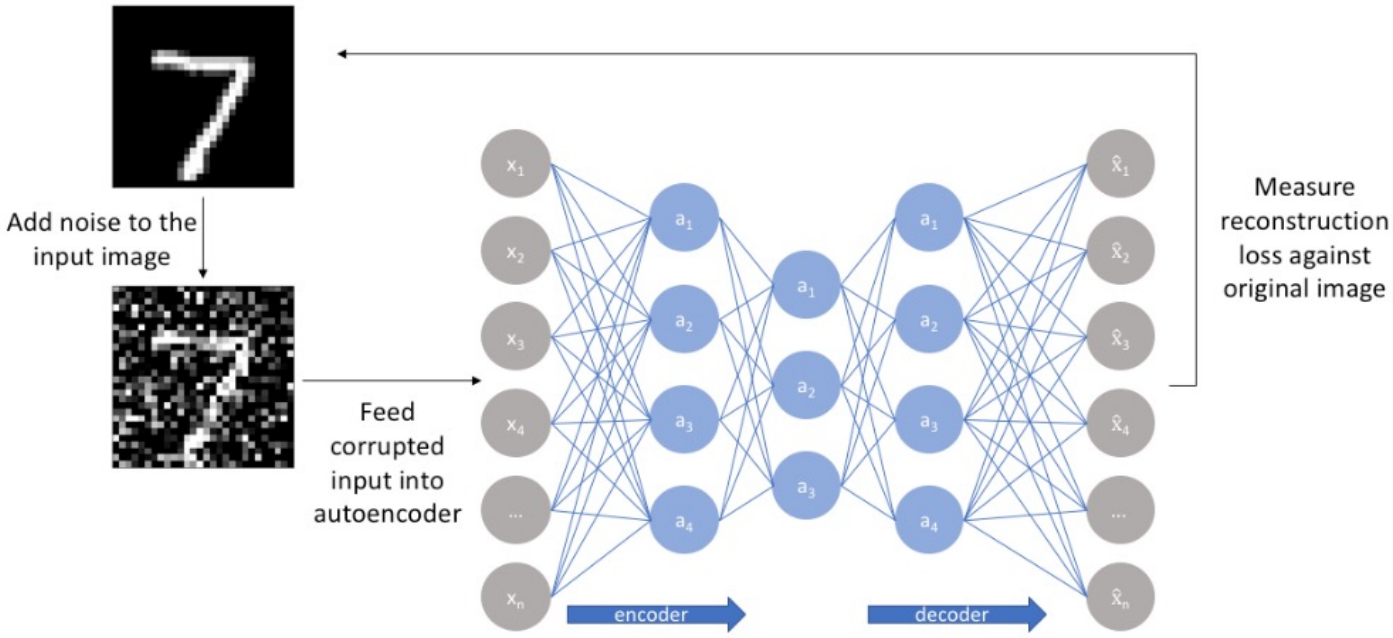
Input and target output are no longer the same, so the network cannot learn to memorize the data!

Trained model can be used to remove noise from new inputs.



Denoising Autoencoders

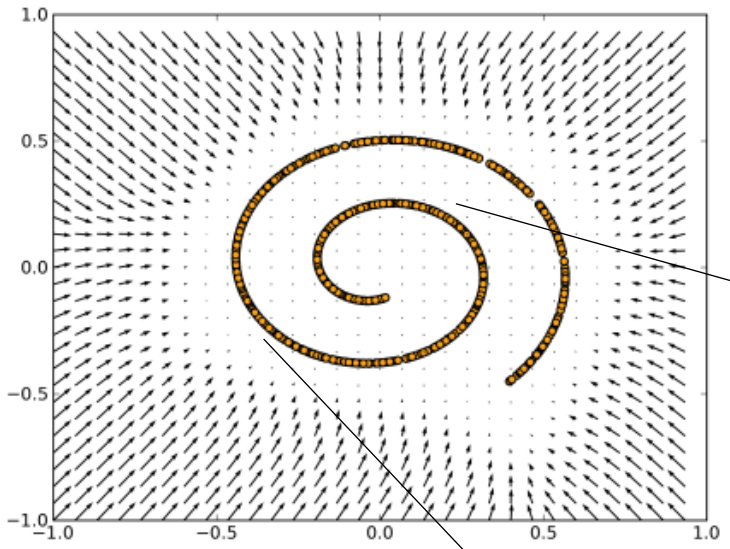
Since the mapping is no longer exactly input-input, what exactly is a DAE learning?



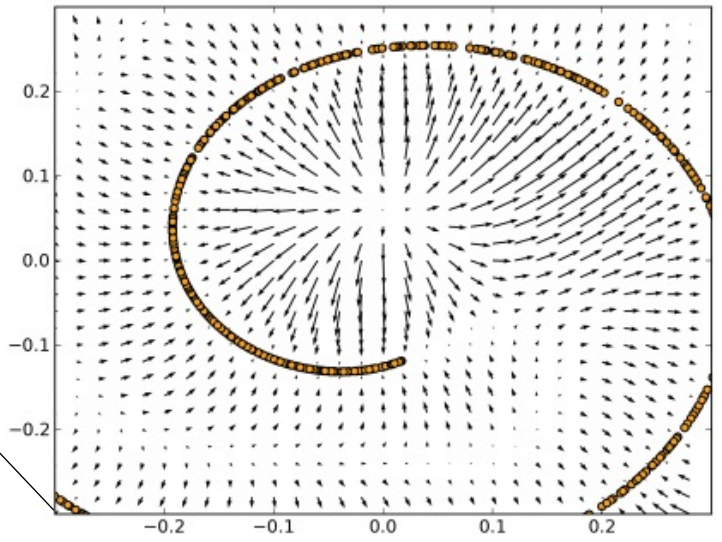
Denoising Autoencoders

The DAE's decoder takes latent representations of corrupted inputs and maps them towards reconstructions of clean outputs.

This can be thought of as a **vector field** flowing towards the clean training data.



(a) $r(x) - x$ vector field, acting as sink, zoomed out



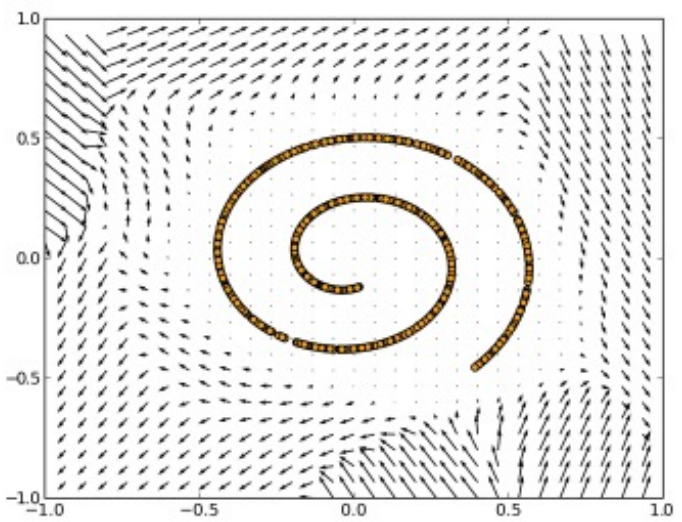
(b) $r(x) - x$ vector field, close-up

Image: Alain and Bengio (2014) "What regularized auto-encoders learn from the data-generating distribution.". JMLR

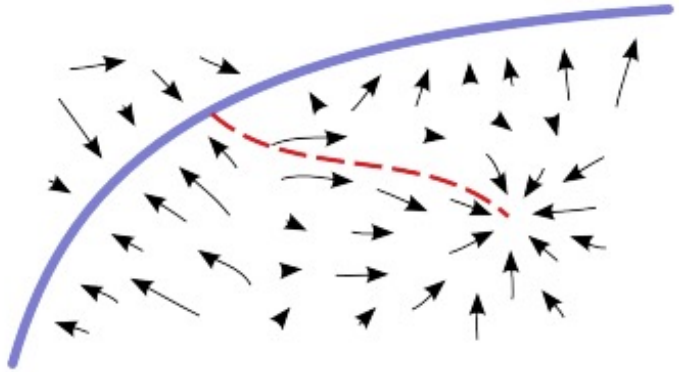
Denoising Autoencoders

Typically, only well-behaved near areas of high density of training data.

May perform poorly on test data far from from what was observed during training.



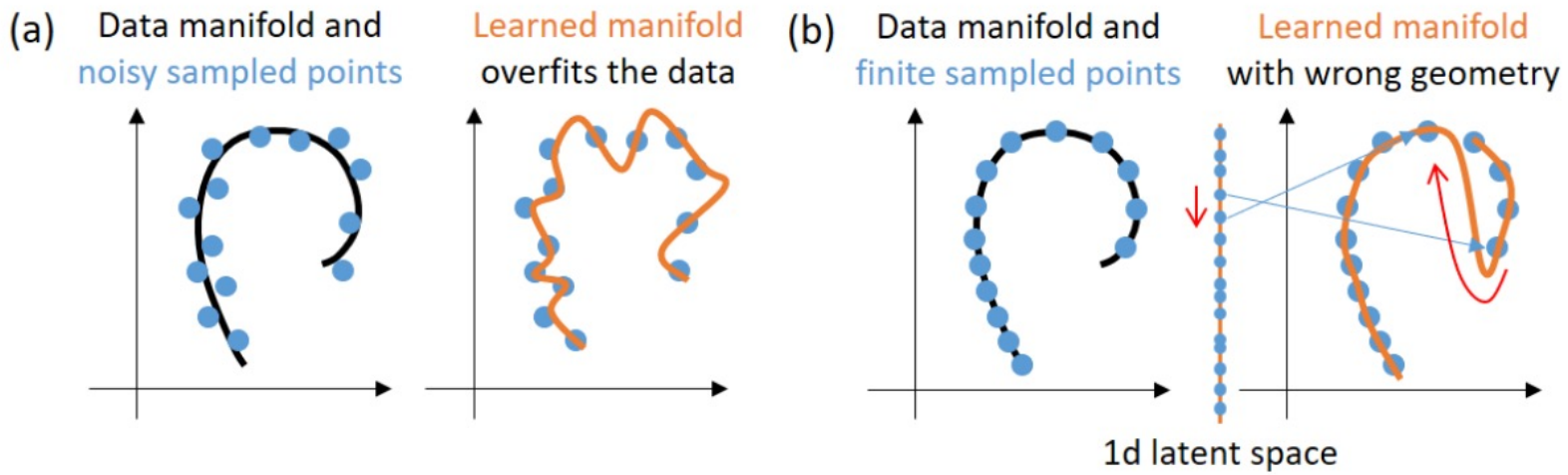
(a) DAE misbehaving when away from manifold



(b) sampling getting trapped into bad attractor

Image: Alain and Bengio (2014) "What regularized auto-encoders learn from the data-generating distribution.". JMLR

Contractive Autoencoders



- Too few and noisy input data makes it difficult to learn a robust latent space representation of the data.
- Goal: enhance robustness of the representation. Force very similar inputs map to very similar latent encodings.

Image: Lee et al. (2021). "Neighborhood Reconstructing Autoencoders." NeurIPS

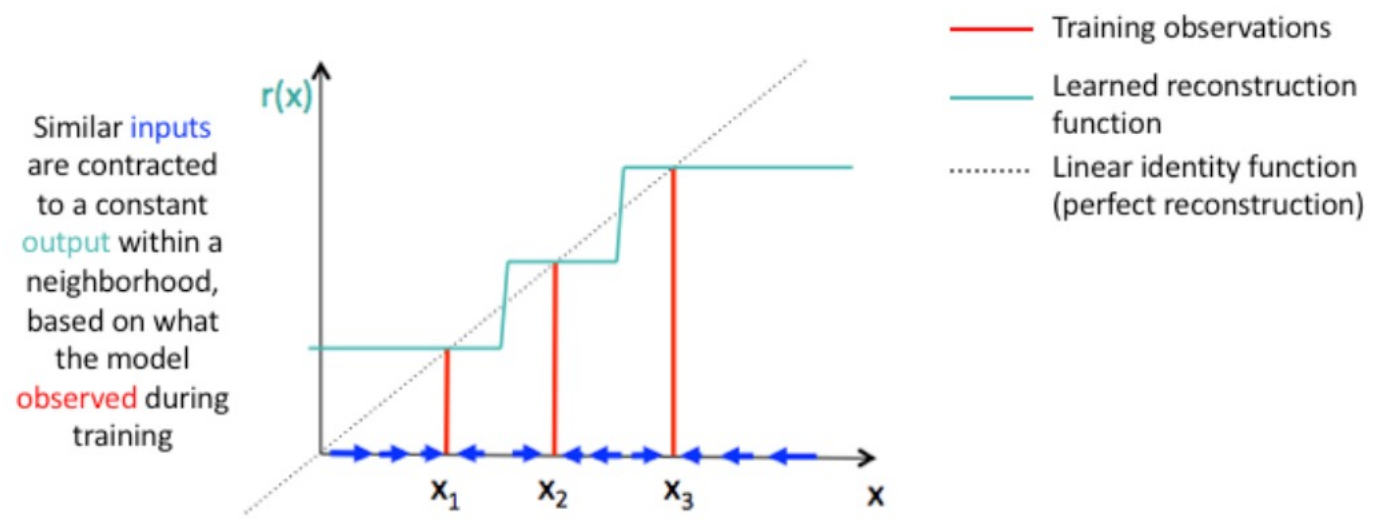
Contractive Autoencoders

- How? Force the **derivatives** of the hidden layer **activations** to be **small** around the input data.
- Regularization penalty on L2 norm of the Jacobian (matrix of partial derivatives) of activations.

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

$$\mathbf{J} = \begin{bmatrix} \frac{\delta a_1^{(h)}(x)}{\delta x_1} & \cdots & \frac{\delta a_1^{(h)}(x)}{\delta x_m} \\ \vdots & \ddots & \vdots \\ \frac{\delta a_n^{(h)}(x)}{\delta x_1} & \cdots & \frac{\delta a_n^{(h)}(x)}{\delta x_m} \end{bmatrix}$$

$$\mathcal{L}(x, \hat{x}) + \lambda \sum_i \left\| \nabla_x a_i^{(h)}(x) \right\|^2$$



Rifai et al (2011). “Contracting auto-encoders: Explicit invariance during feature extraction” ICML
Alain and Bengio (2014) "What regularized auto-encoders learn from the data-generating distribution.”. JMLR

Robustness of representation and reconstruction

Denoising and Contractive autoencoders have similar goals

Contractive autoencoders make the *feature-extraction* function (**encoder**) resist small changes to the input

Denoising autoencoders make the *reconstruction* function (**decoder**) resist small changes to the original input

Notebook

Variational Autoencoder

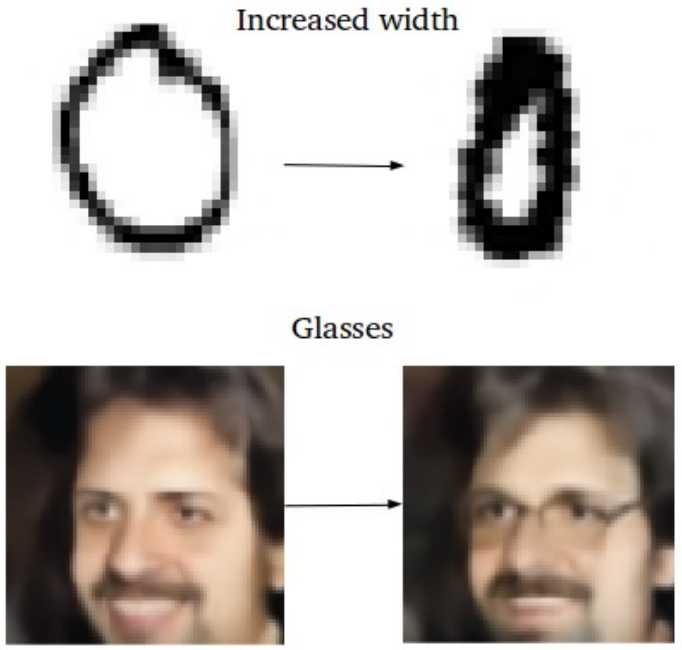
A probabilistic approach to latent space reconstruction.

These are known as ***generative models.***

VAEs map to robust latent spaces that approximate the probability distribution of the input data.

This lets you:

- Generate random, synthetic outputs that look realistic.
- Alter or explore variations on *data you already have*, and not just in a random way, but in a desired, *specific* direction.

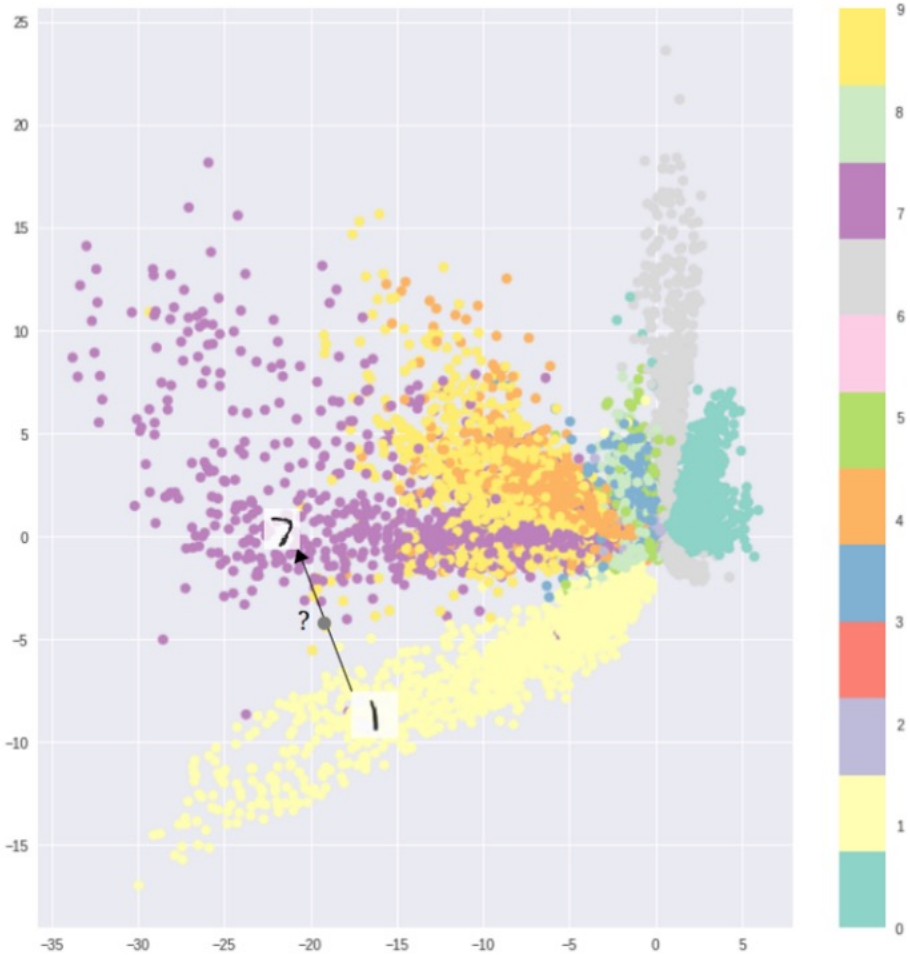


https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

Interpolation

- For standard autoencoders, the latent space they convert their inputs to and where their encoded vectors lie, may not be continuous, or allow easy interpolation.

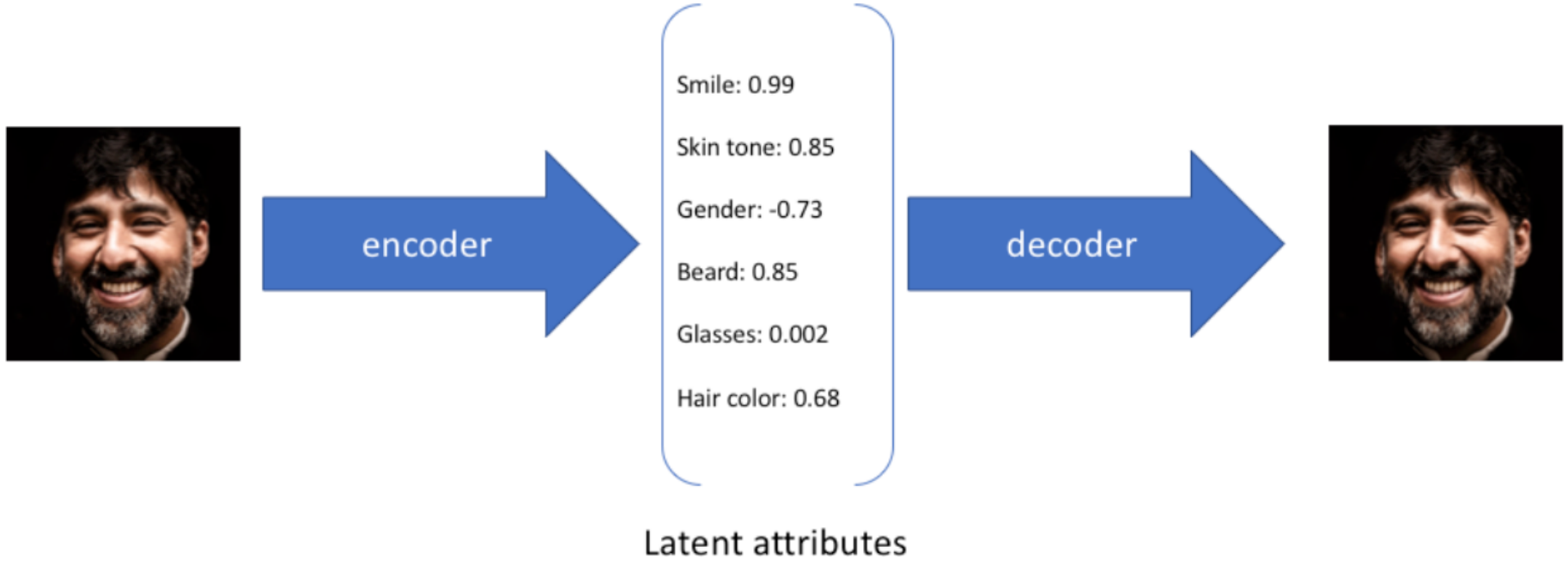
- If the space has discontinuities (e.g., gaps between clusters) and you sample/generate a variation from there, the decoder will simply generate an unrealistic output.



https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

Intuition

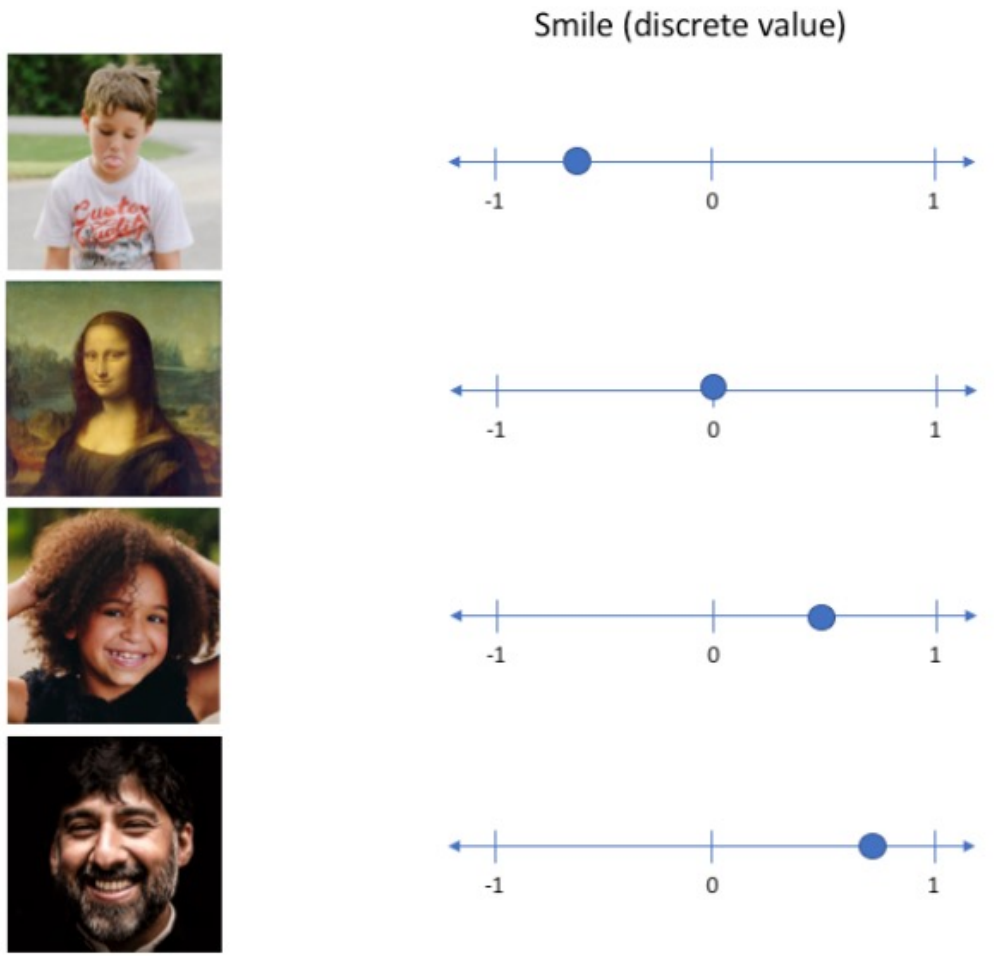
Traditional autoencoder: output a **single value** to describe each latent state attribute (dimension).



https://www.jeremyjordan.me/variational-autoencoders/

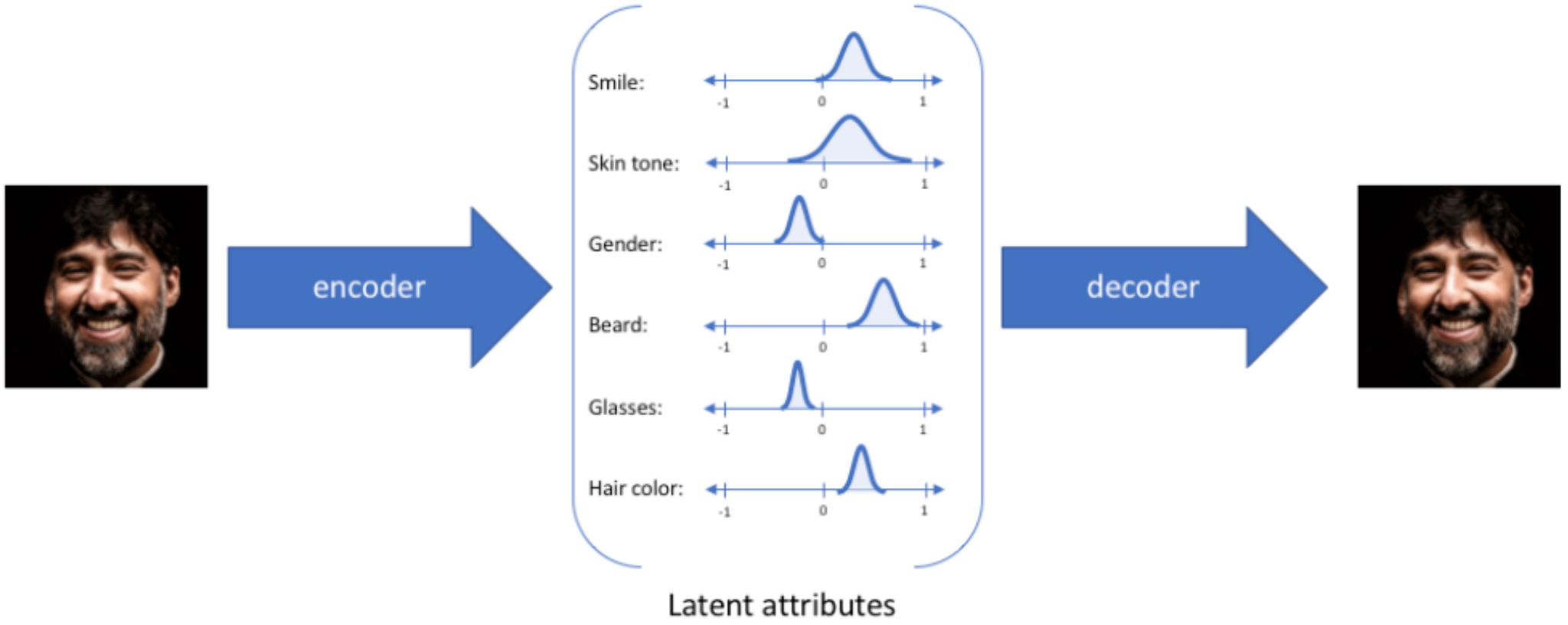
Intuition

VAE: describe a **probability distribution** for each latent attribute!



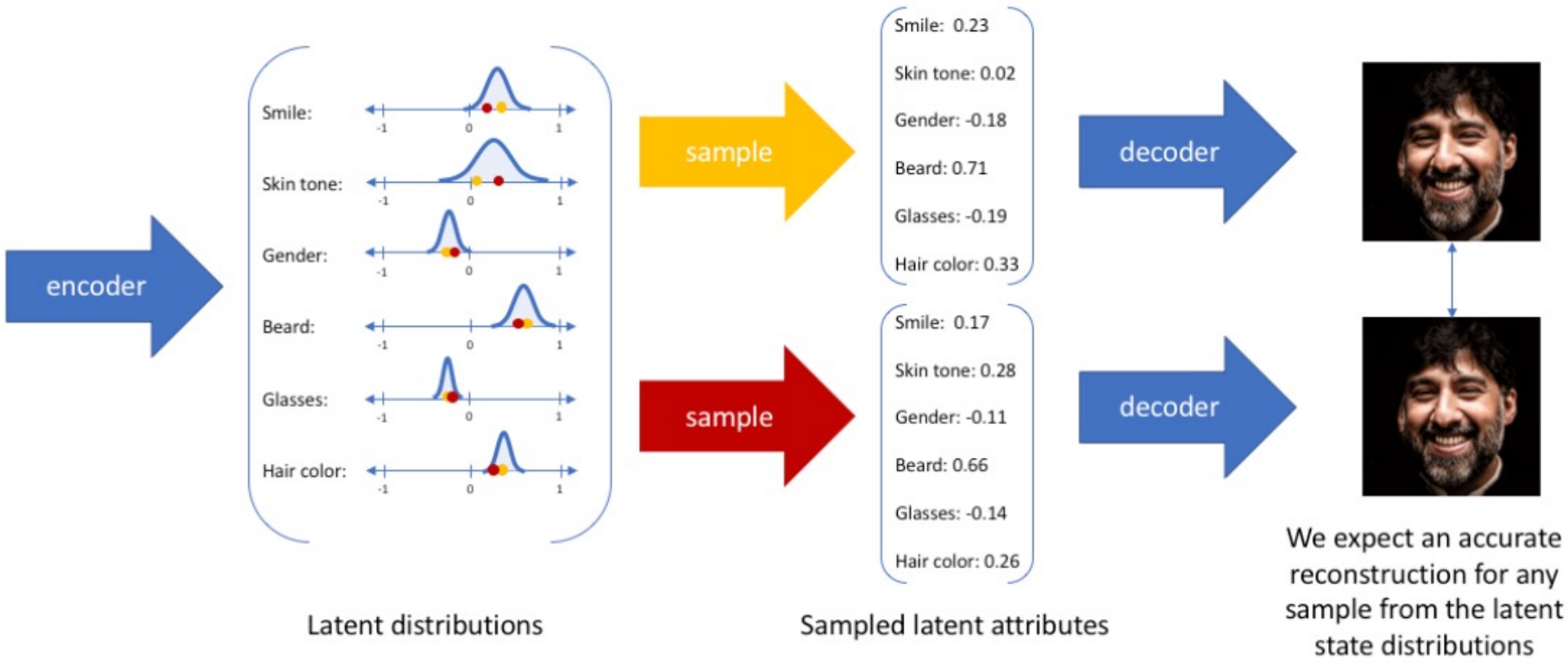
https://www.jeremyjordan.me/variational-autoencoders/

Intuition



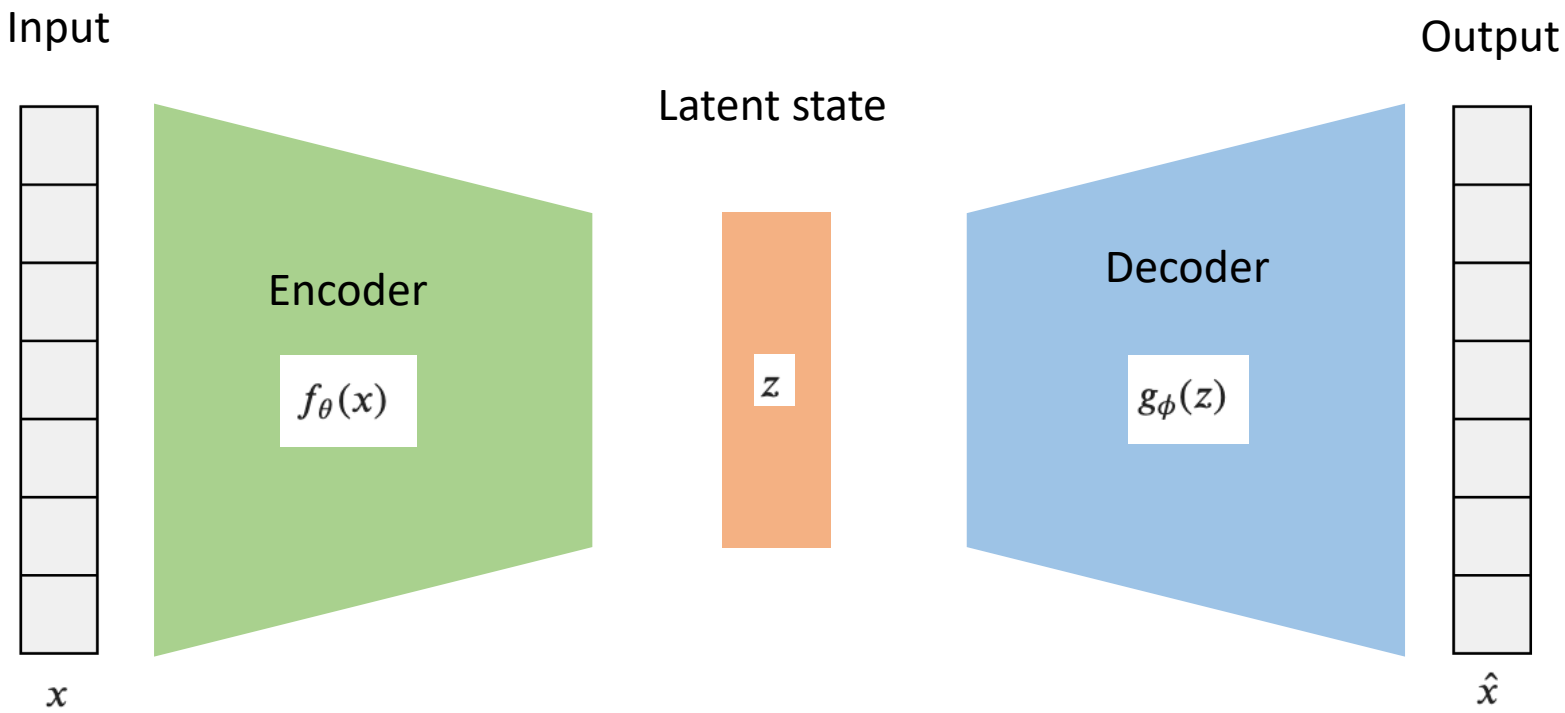
https://www.jeremyjordan.me/variational-autoencoders/

Intuition

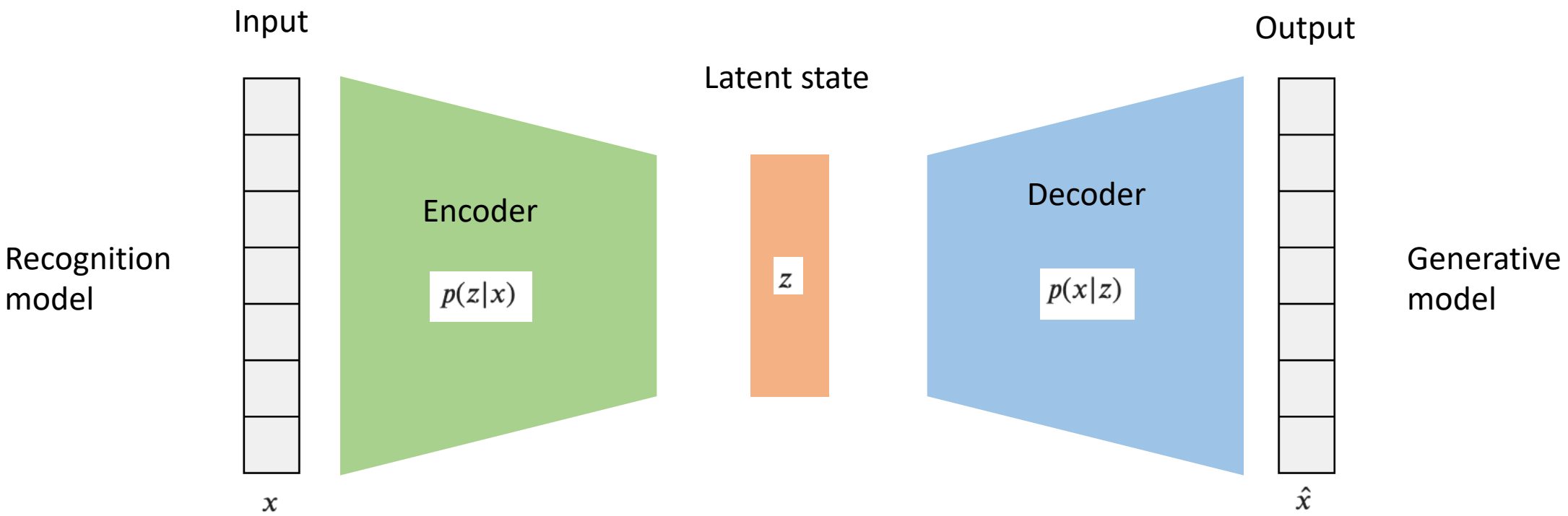


https://www.jeremyjordan.me/variational-autoencoders/

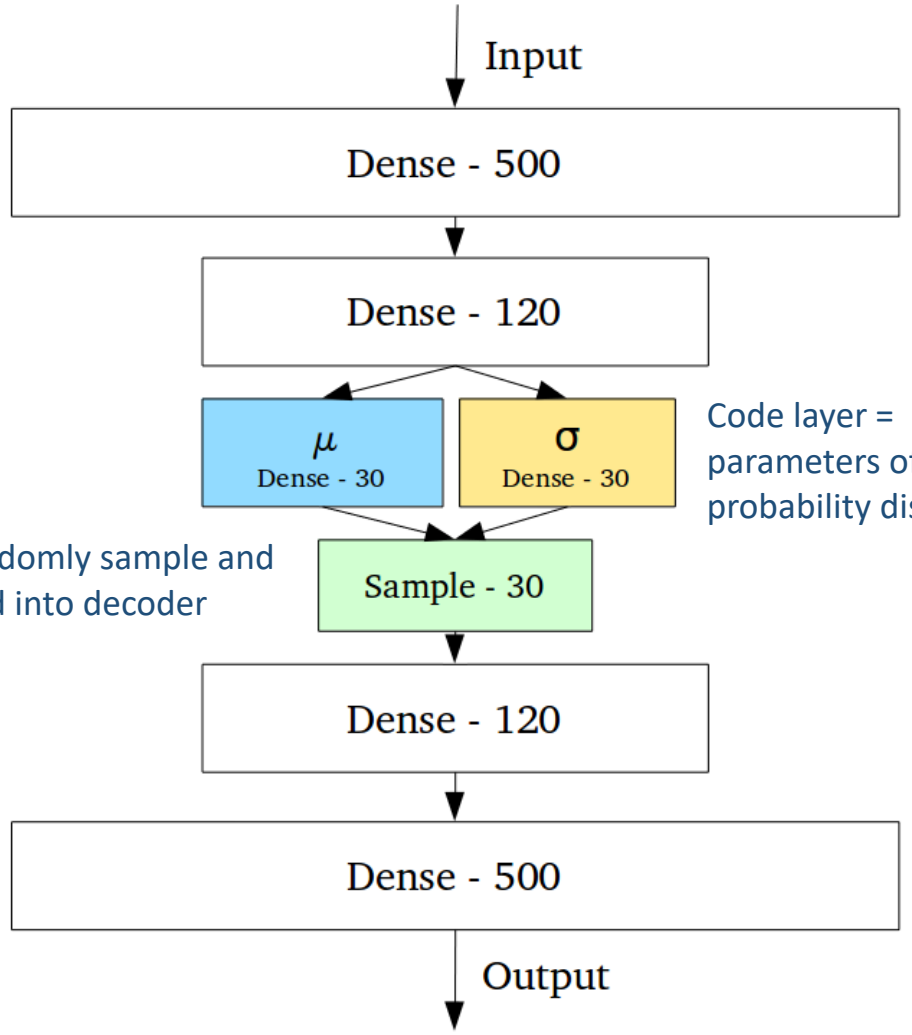
Statistical motivation



Statistical motivation



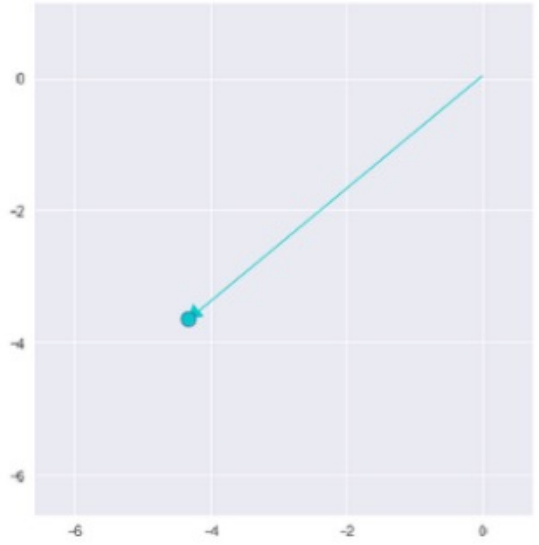
Implementation



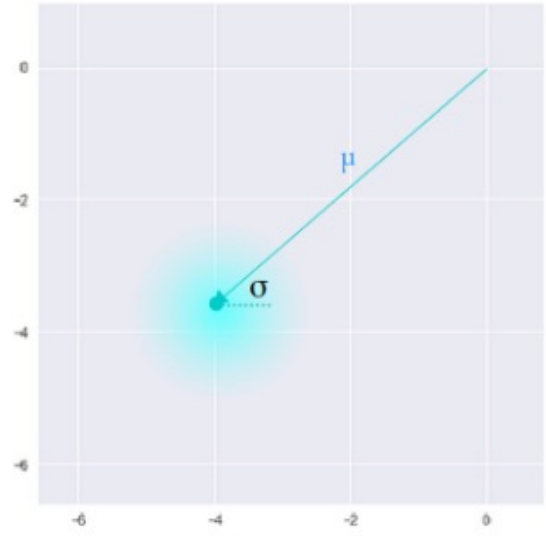
Variational Autoencoder

Code layer = parameters of latent probability dist.

Randomly sample and feed into decoder



Standard Autoencoder
(direct encoding coordinates)



Variational Autoencoder
(μ and σ initialize a probability distribution)

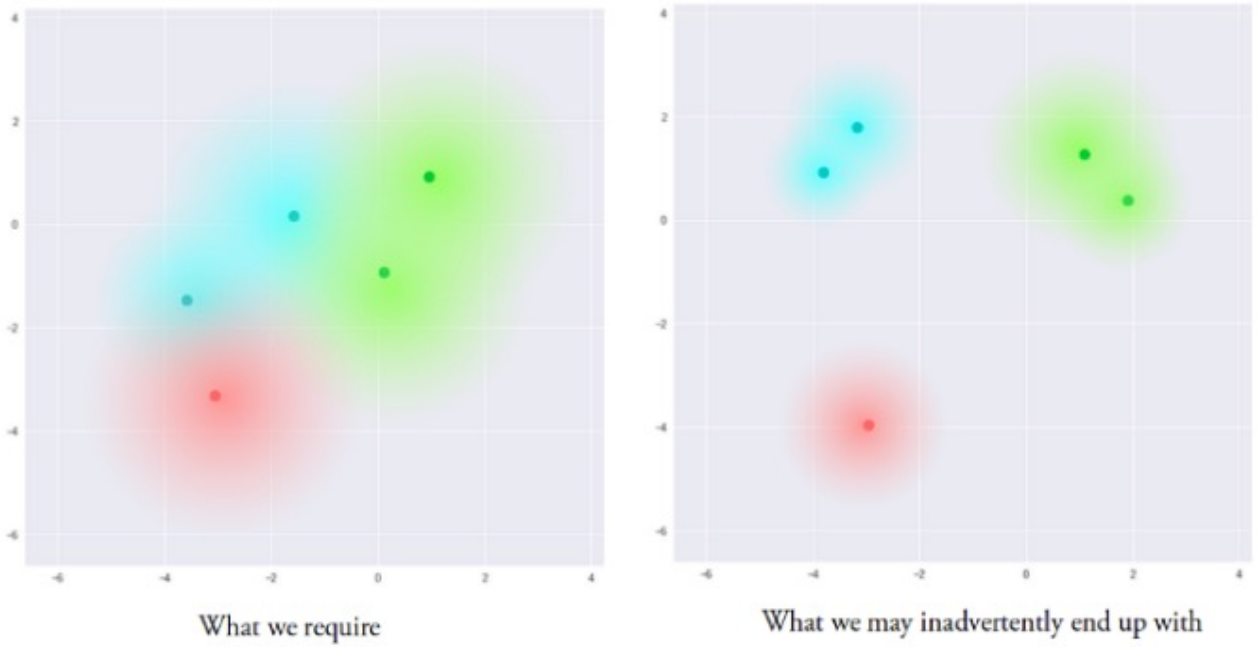
Tractable distribution:
independent multivariate Gaussian

https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

Enabling long-range interpolation

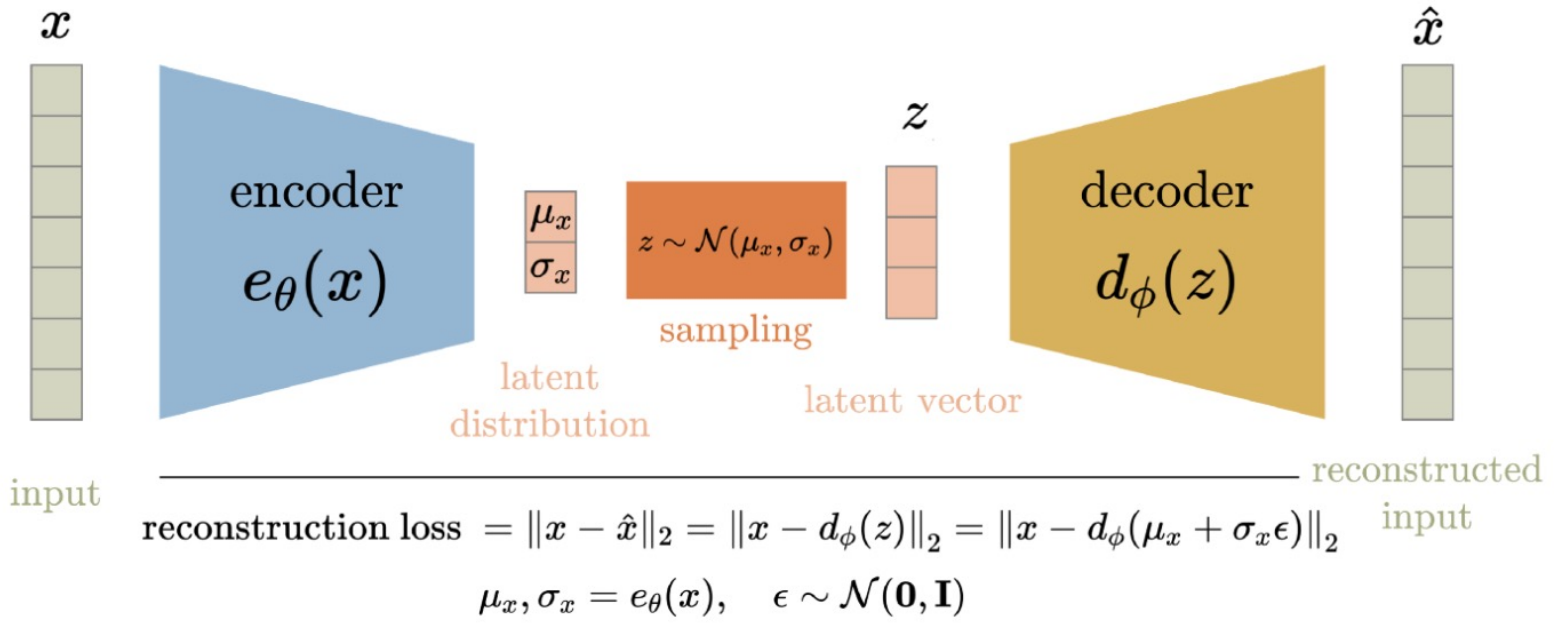
Probabilistic encoding of individual inputs results in locally smooth latent spaces (in clusters of similar samples)

Ideally, we also want overlap between samples that are not very similar in order to interpolate *between* classes.



https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

VAE regularization

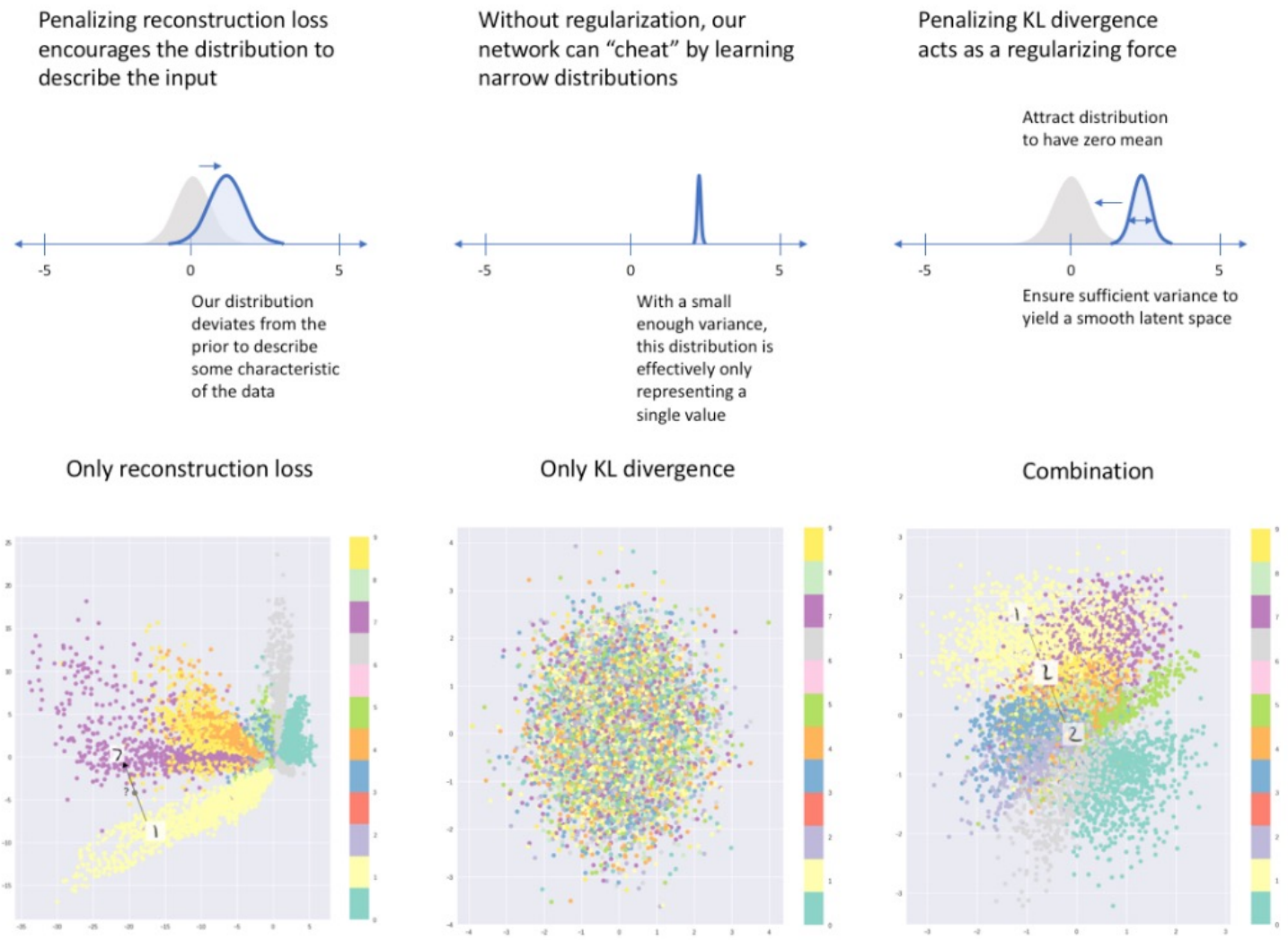


Similarity regularizer

Encourage parameters of the independent multivariate Normal $z_i \sim N(\mu_i, \sigma_i)$ emitted for each input x_i to closely resemble a target distribution (the standard unit Normal N(0, I) centered at the origin of latent space).

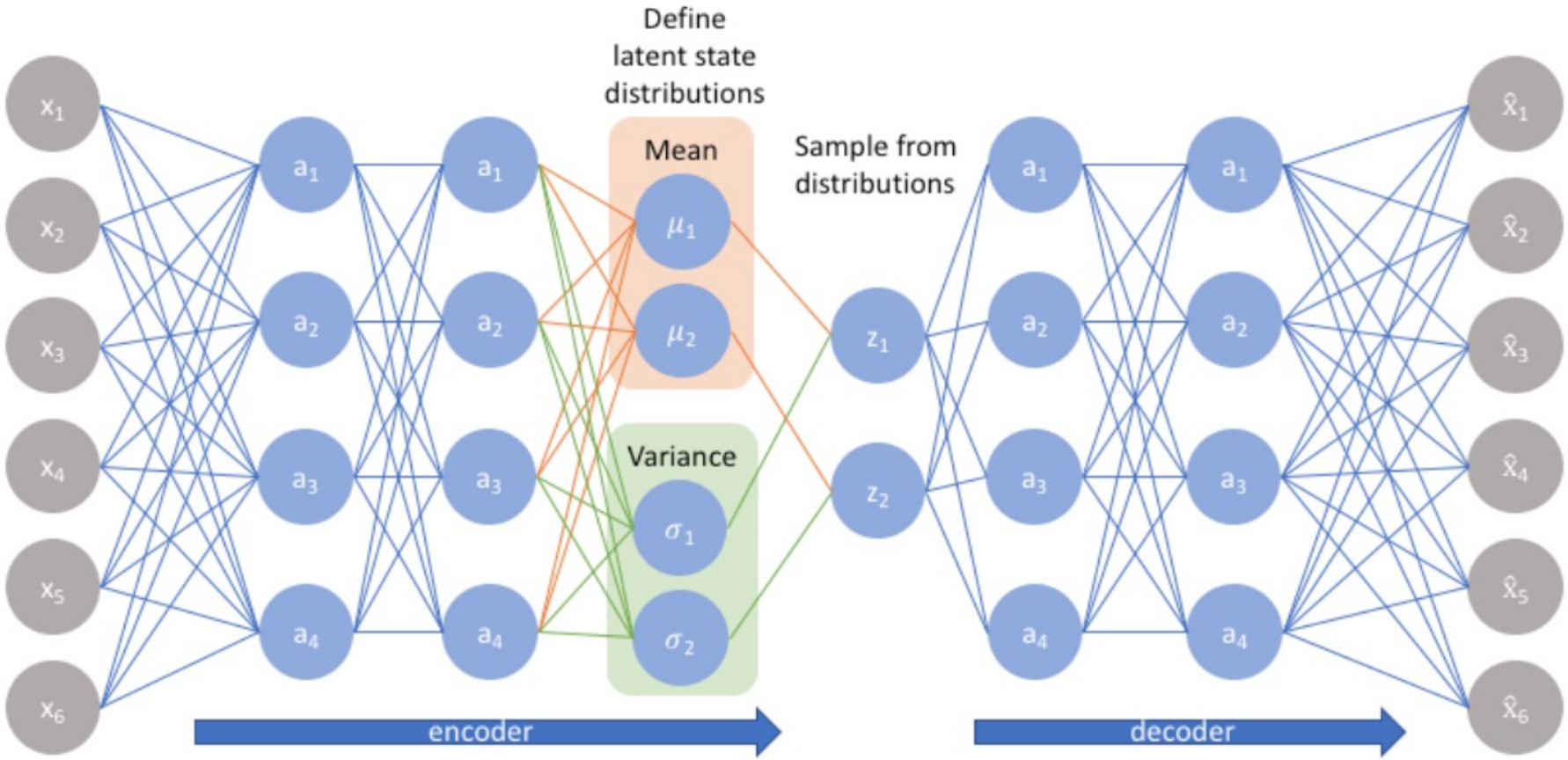
https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2

Effect on latent space



Optimizing both objectives results in a latent space which maintains the similarity of nearby encodings on the *local scale*, yet *globally,* is very densely packed near the latent space origin

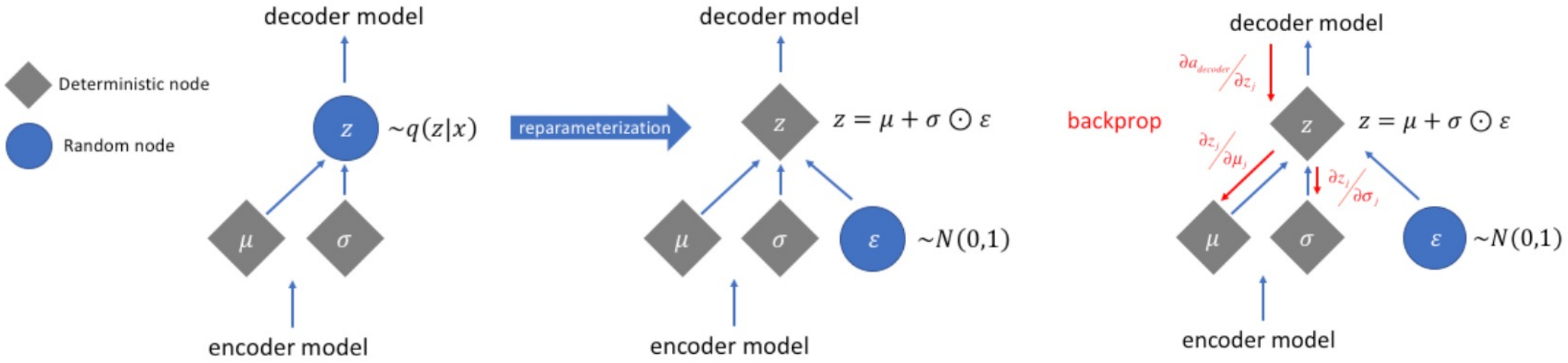
Training



https://www.jeremyjordan.me/variational-autoencoders/

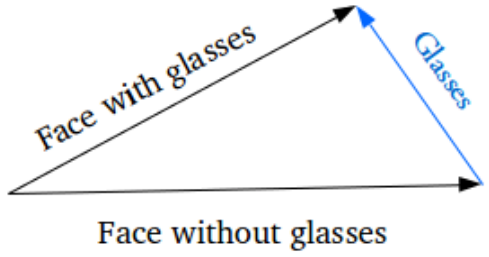
Training: reparameterization trick

- Backpropagation is not compatible with random variable nodes in the code layer.
- Trick: we factor out a unit Gaussian node ε for sampling instances of z during training, so gradients can be calculated.
- Network can now learn moment estimates μ and σ from data!

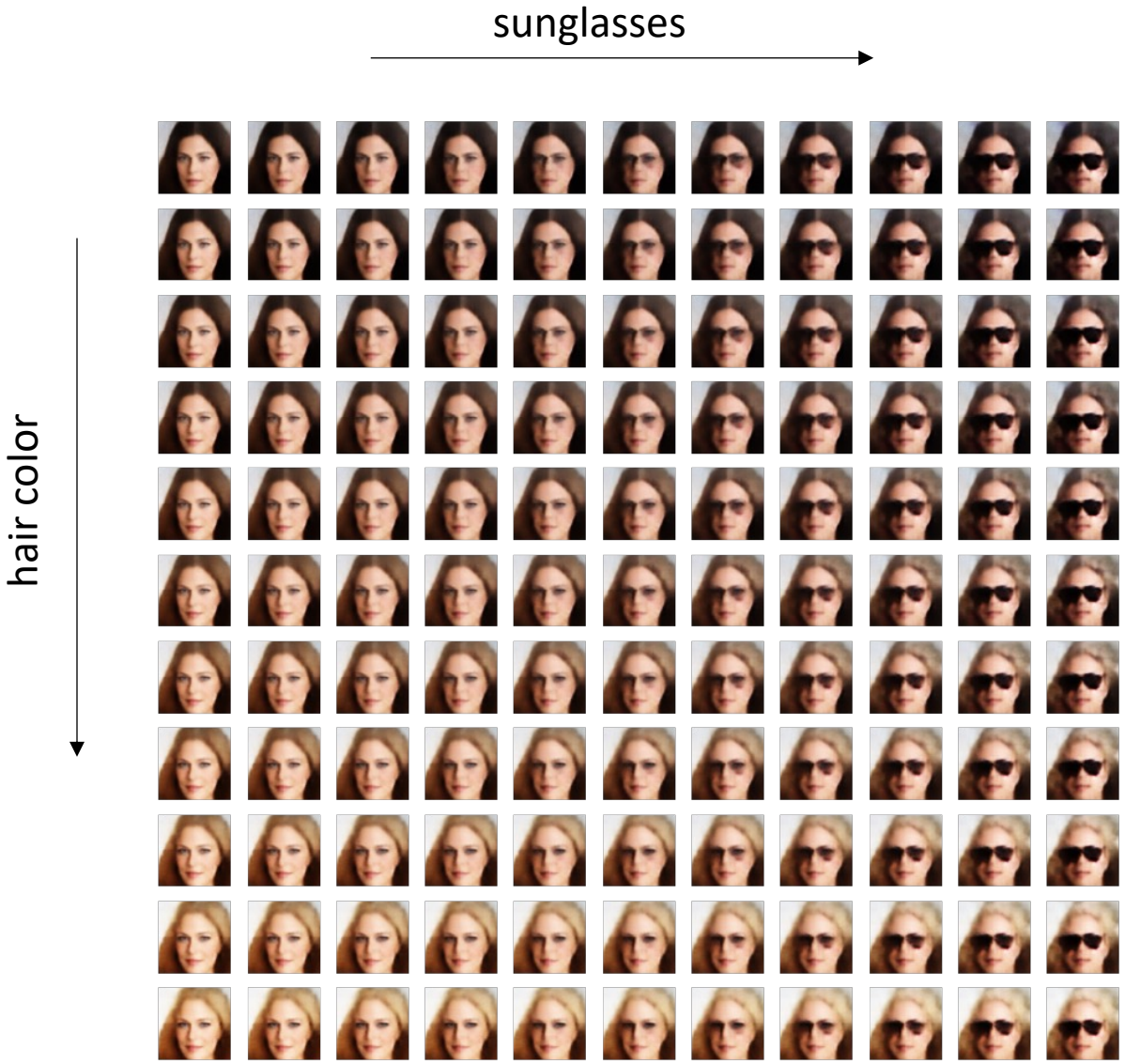


https://www.jeremyjordan.me/variational-autoencoders/

Vector arithmetic



Adding new features to samples



https://www.compthree.com/blog/autoencoder/

https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

Notebook: VAE

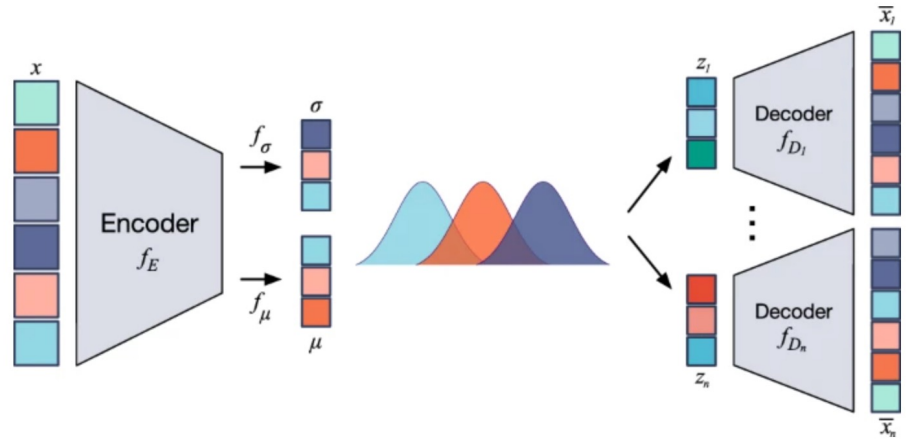
Applications: scRNA –seq analysis

Article | Open Access | Published: 15 February 2021

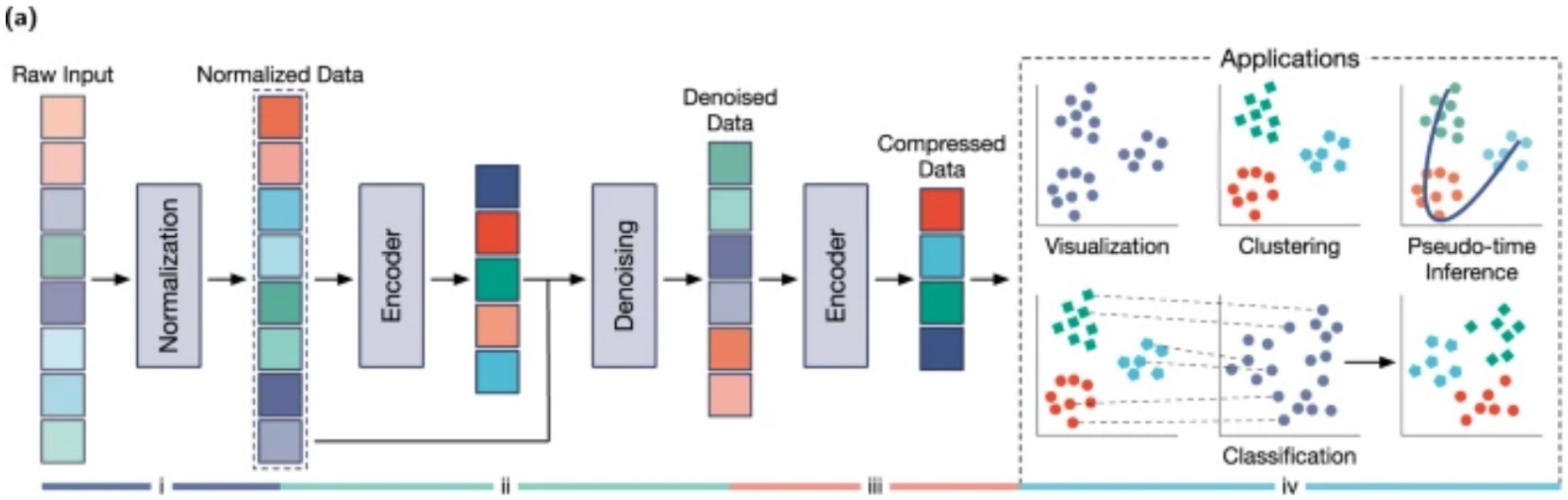
Fast and precise single-cell data analysis using a hierarchical autoencoder

Duc Tran, Hung Nguyen, Bang Tran, Carlo La Vecchia, Hung N. Luu & Tin Nguyen

Nature Communications **12**, Article number: 1029 (2021) | Cite this article



The encoder projects input data to multiple low-dimensional latent spaces (outputs of z_1 to z_n layers). The decoders infer original data from these latent data. Minimizing the difference between inferred data and original one leads to a high quality representation of the original data at bottleneck layer (outputs of μ layer).



https://www.nature.com/articles/s41467-021-21312-2

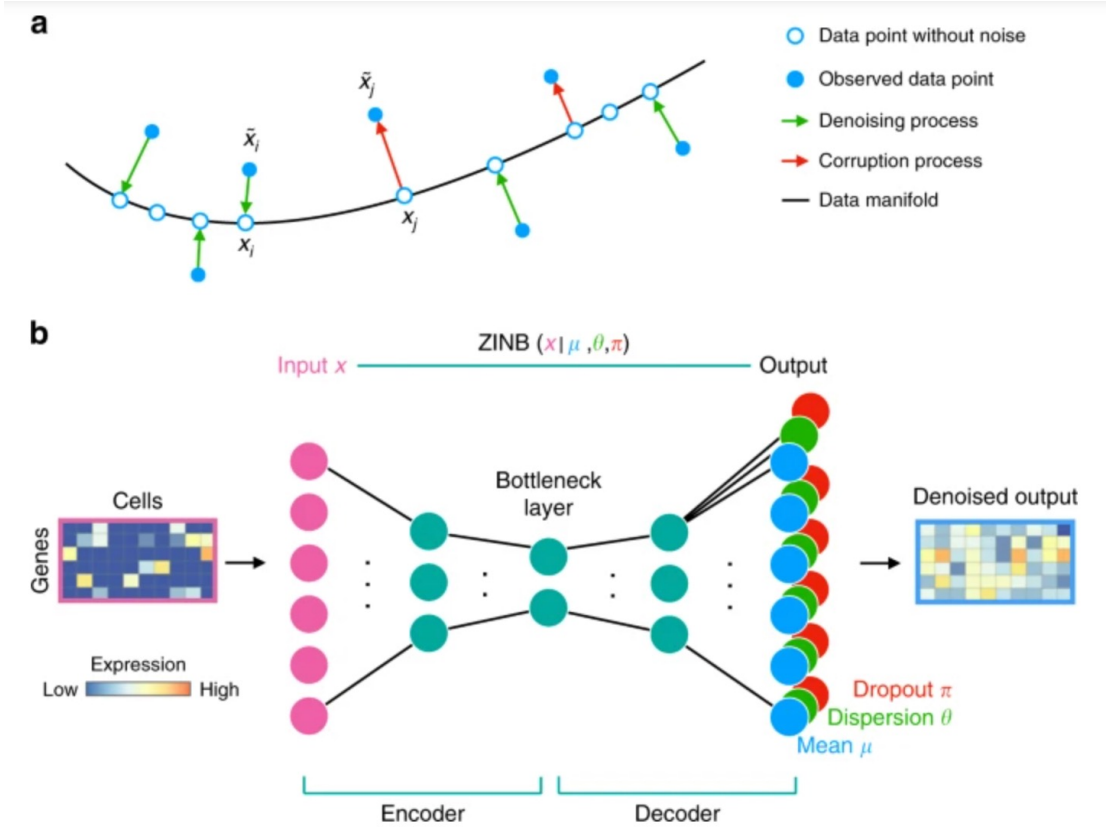
Applications: scRNA –seq analysis

Article | Open Access | Published: 23 January 2019

Single-cell RNA-seq denoising using a deep count autoencoder

Gökcen Eraslan, Lukas M. Simon, Maria Mircea, Nikola S. Mueller & Fabian J. Theis

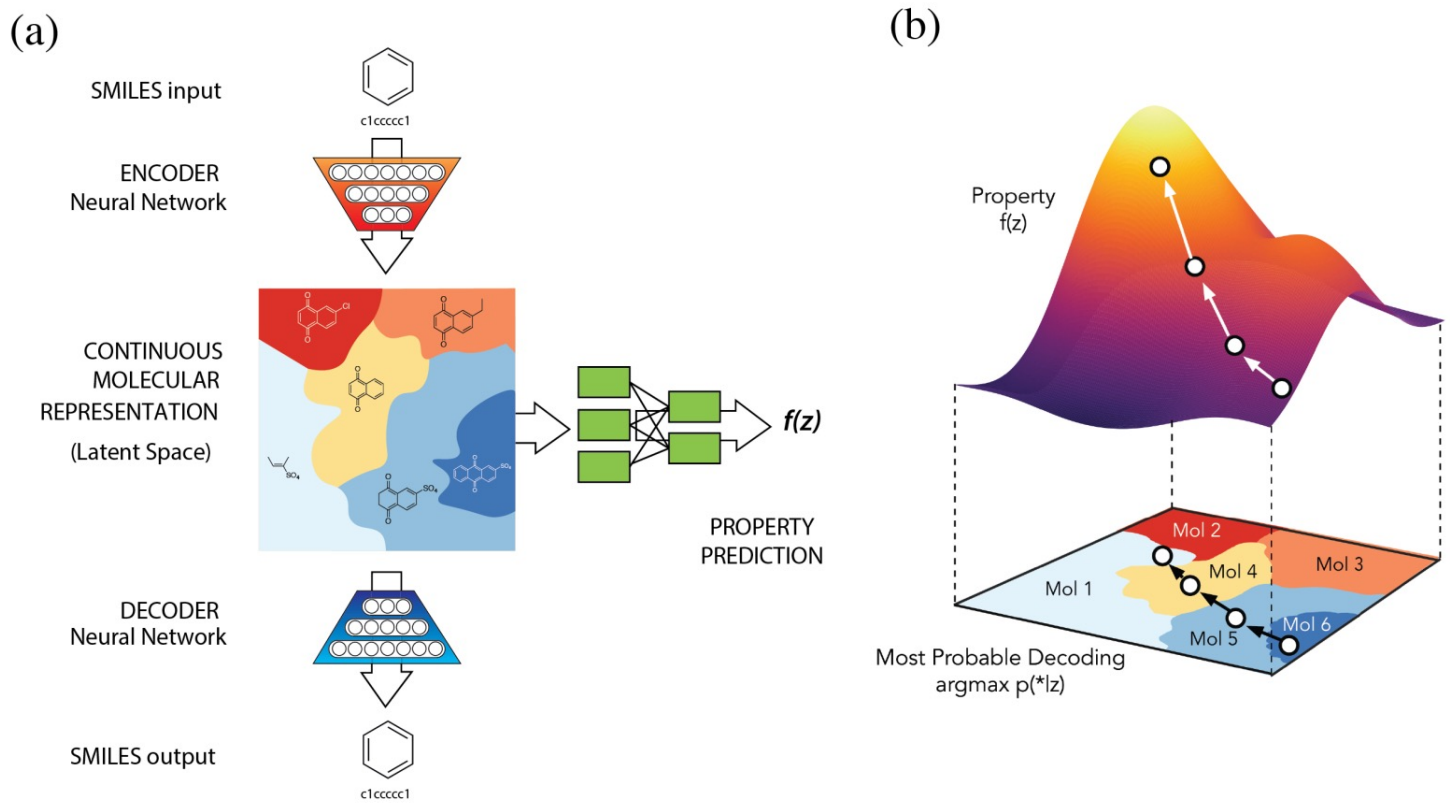
Nature Communications **10**, Article number: 390 (2019) | Cite this article



https://www.nature.com/articles/s41467-018-07931-2

Applications: Drug design

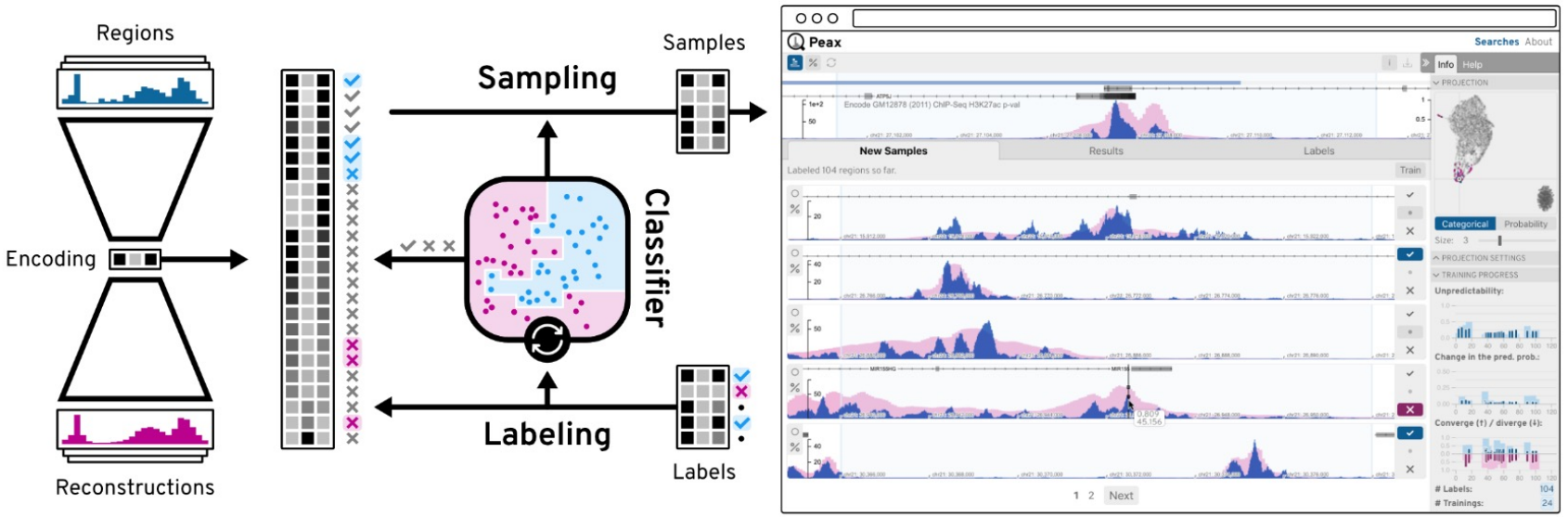
Encoder converts the discrete representation of a molecule (SMILES) into a real-valued continuous vector, and the Decoder converts these continuous vectors back to discrete molecule representations.

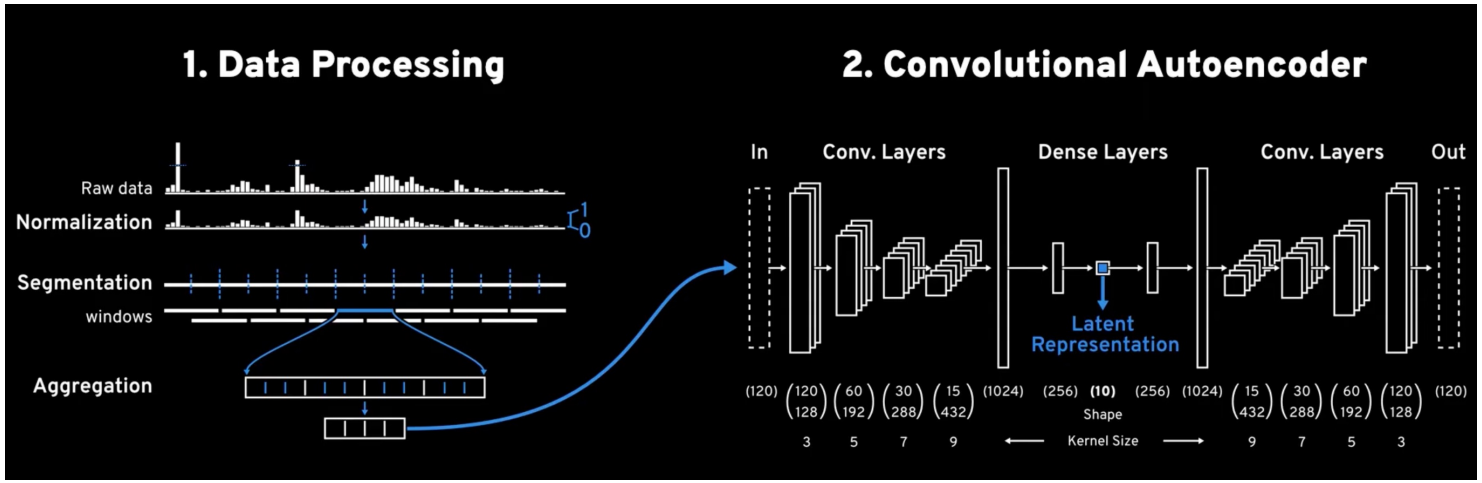


https://keras.io/examples/generative/molecule_generation/

https://arxiv.org/abs/1610.02415

Applications: Epigenomic pattern recommendation engine





https://peax.lekschas.de/

Resources

- Chapters 14 and 15 in Goodfellow et al. Text
- https://www.jeremyjordan.me/autoencoders/
- https://neptune.ai/blog/autoencoders-case-study-guide
- https://www.jeremyjordan.me/variational-autoencoders/
- https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf
- https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73
- https://www.compthree.com/blog/autoencoder/