Autoencoders

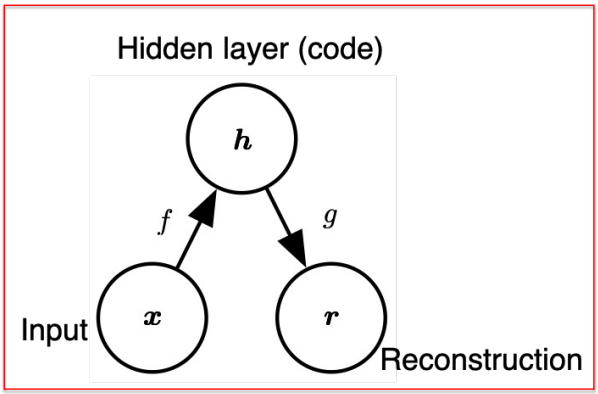
Autoencoder: General Idea

Learn a (compressed and/or sparse) code h for the input x

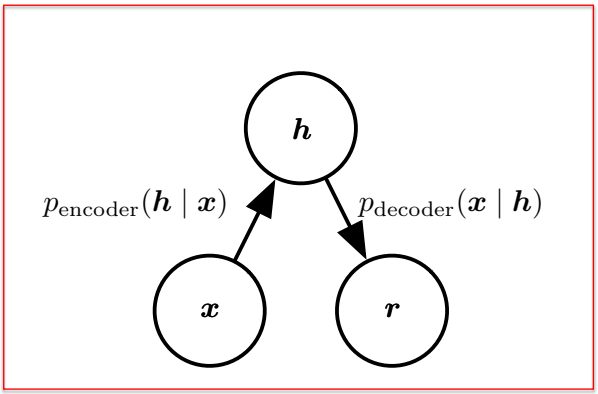
How ? By learning the identity function $\boldsymbol{x} = g(f(\boldsymbol{x}))$ with constraints:

- on the architecture of the network (undercomplete autoencoder)
- adding a regularizing term to the loss (overcomplete autoencoder)

Deterministic



Stochastic



Autoencoder: General Idea

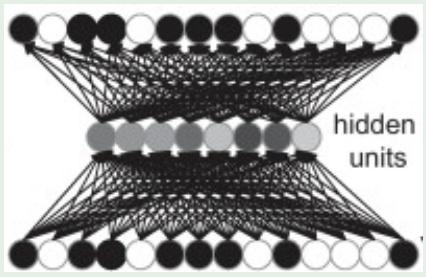
Not useful to learn identity function on the whole input domain!

Identity function only on $\boldsymbol{x}_i \in \mathcal{T}$... and hopefully, similar points $\Rightarrow$ manifold!

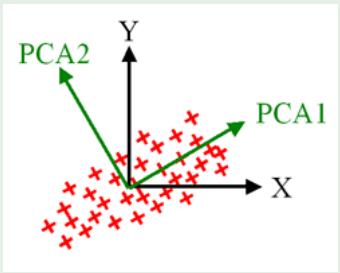
- Undercomplete autoencoders:
  - $\boldsymbol{h}$ has lower dimension than $\boldsymbol{x}$
  - $f(\cdot)$ or $g(\cdot)$ has low capacity (e.g., linear g)
  - must discard some information in $\boldsymbol{h}$
- Overcomplete autoencoders:
  - $\boldsymbol{h}$ has higher dimension than $\boldsymbol{x}$
  - must be regularized

Undercomplete: Linear Autoencoder Networks

Autoencoder Networks



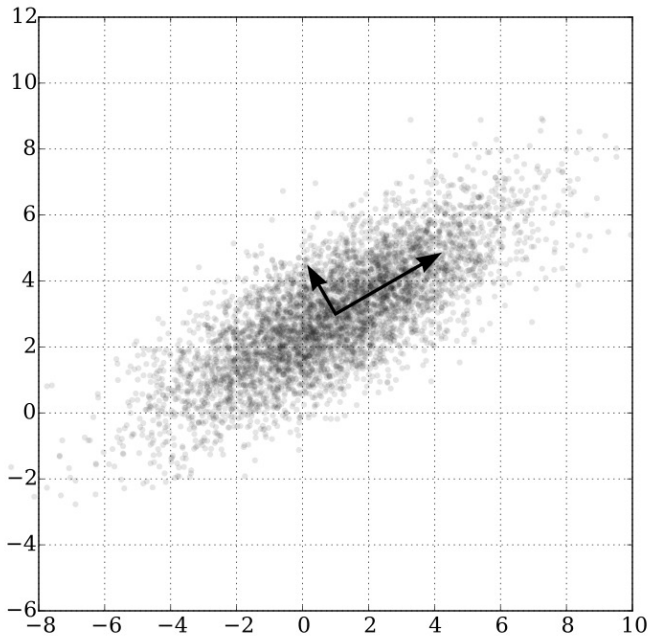
Linear Units $\Longleftrightarrow$



- Linear autoencoder networks are actually closely linked to PCA [Bourlard and Kamp, 1988; Baldi and Hornik, 1989]
- Principal Components: (orthogonal) directions of maximum variance of data
- Compression of information: define the optimal sub-space where to project data
- Used in Pattern Recognition to reduce the dimension of data and/or noise (see image processing)

How to Compute Linear Autoencoders

Compute an orthogonal basis (principal directions) minimizing the average squared distance from a point $\boldsymbol{x}$ in the dataset to the spanned subspace



How to Compute Linear Autoencoders

Singular Value Decomposition (SVD)

- Collect all the data $\boldsymbol{x}_i$ as rows (columns) in a matrix $\boldsymbol{\Xi} \in \mathbb{R}^{n \times m}$
- Compute the SVD decomposition of $\boldsymbol{\Xi}$

$$\boldsymbol{\Xi} = \boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^\top$$

where

- $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ is an orthonormal matrix (columns are a basis for the rows of $\boldsymbol{\Xi}$)
- $\boldsymbol{S} \in \mathbb{R}^{n \times m}$ is a diagonal matrix with non-negative real numbers on the diagonal (*singular values*)
- $\boldsymbol{U} \in \mathbb{R}^{m \times m}$ is an orthonormal matrix (columns are a basis for the columns of $\boldsymbol{\Xi}$)

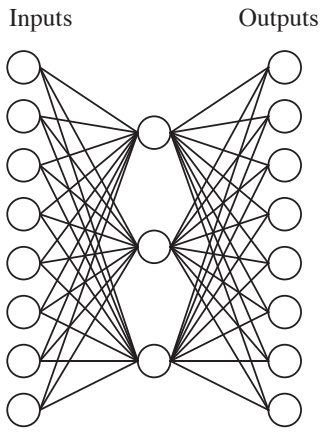
Example:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \end{bmatrix}}_{\boldsymbol{\Xi}} = \underbrace{\begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{bmatrix}}_{\boldsymbol{V}} \underbrace{\begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & \sqrt{5} & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\boldsymbol{S}} \underbrace{\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ -\sqrt{0.2} & 0 & 0 & 0 & -\sqrt{0.8} \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -\sqrt{0.8} & 0 & 0 & 0 & \sqrt{0.2} \end{bmatrix}}_{\boldsymbol{U}^\top}$$

$\boldsymbol{x}_i$ as rows: reduced representations $\Rightarrow \boldsymbol{V}\boldsymbol{S}\underbrace{\boldsymbol{U}^\top\boldsymbol{U}}_{=\boldsymbol{I}} = \boldsymbol{V}\boldsymbol{S}$

Undercomplete: Shallow Autoencoder Networks

Learning the identity function:
(lossy) compression of data

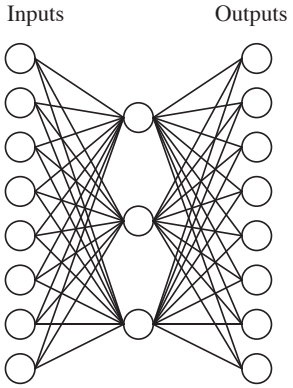


Supervised learning for..
Unsupervised learning!!

Input	Output
00000001	00000001
00000010	00000010
00000100	00000100
00001000	00001000
00010000	00010000
00100000	00100000
01000000	01000000
10000000	10000000

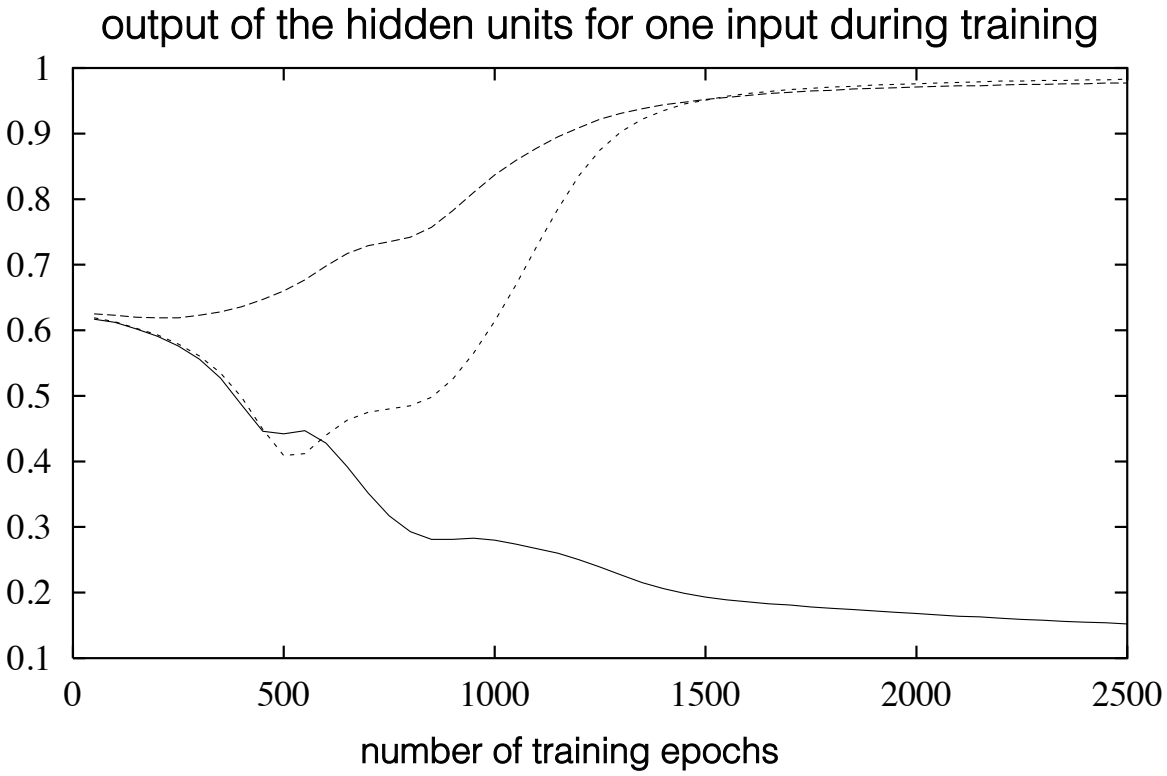
Undercomplete: Shallow Autoencoders

After learning…

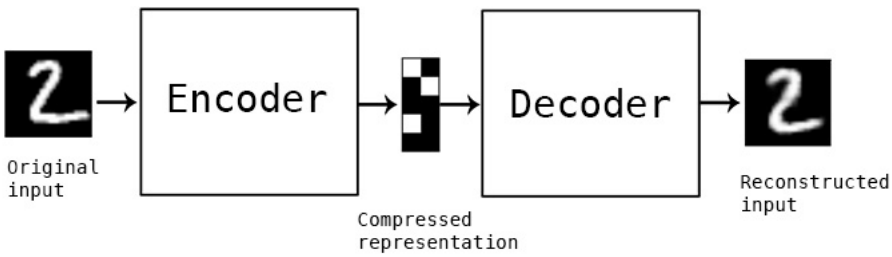


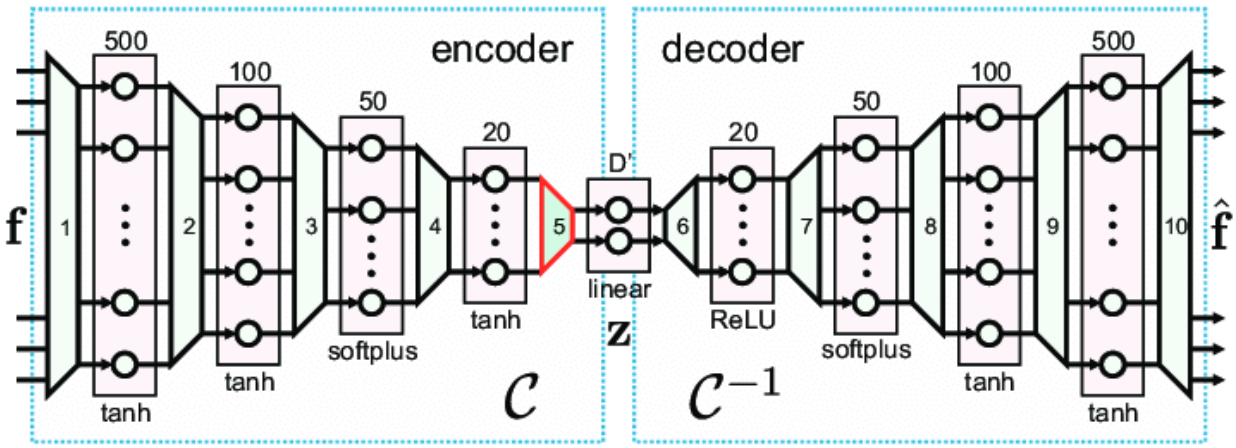
Input		Hidden values		Output
10000000	→	0.89 0.04 0.08	→	10000000
01000000	→	0.01 0.11 0.88	→	01000000
00100000	→	0.01 0.97 0.27	→	00100000
00010000	→	0.99 0.97 0.71	→	00010000
00001000	→	0.03 0.05 0.02	→	00001000
00000100	→	0.22 0.99 0.99	→	00000100
00000010	→	0.80 0.01 0.98	→	00000010
00000001	→	0.60 0.94 0.01	→	00000001

Undercomplete: Shallow Autoencoders



Undercomplete: Deep Autoencoders





Overcomplete Autoencoders

Regularized autoencoders

- Sparse autoencoders
- Denoising autoencoders
- Contractive autoencoders
- Autoencoders with Dropout on the hidden layer

Sparse Autoencoders

Limit the capacity of autoencoder

- By adding a term to the cost function penalizing the entries of the code $\boldsymbol{h} = f(\boldsymbol{x})$ for being larger

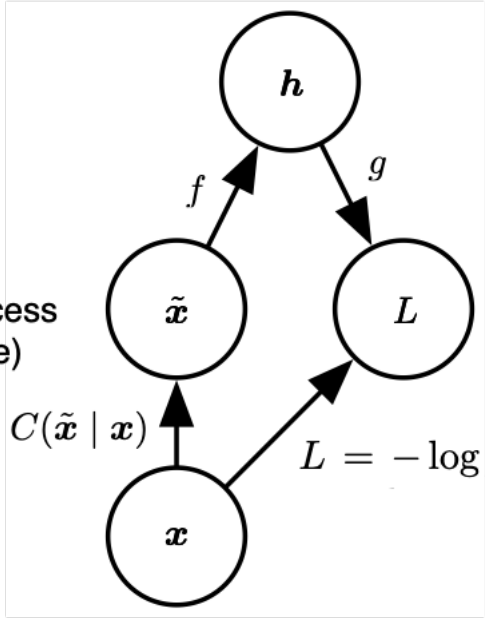
$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \boldsymbol{\Omega}(\boldsymbol{h})$$

- Special case of variational autoencoder (will present in the future)
  - Probabilistic model $p_{\text{model}}(\boldsymbol{x}, \boldsymbol{h}) = p_{\text{model}}(\boldsymbol{h}) p_{\text{model}}(\boldsymbol{x}|\boldsymbol{h})$
    - $\boldsymbol{h}$ latent variables *generating* $\boldsymbol{x}$
    - $\log p_{\text{model}}(\boldsymbol{x}, \boldsymbol{h}) = \log p_{\text{model}}(\boldsymbol{h}) + \log p_{\text{model}}(\boldsymbol{x}|\boldsymbol{h})$
  - Laplace prior corresponds to L1 sparsity penalty
    - $p_{\text{model}}(h_i) = \frac{\lambda}{2} \exp^{-\lambda|h_i|}$

$$-\log p_{\text{model}}(\boldsymbol{h}) = \sum_i \left(\lambda |h_i| - \log \frac{\lambda}{2} \right) = \lambda \sum_i |h_i| + \text{const}$$

  - Other sparsity penalty terms have been proposed in the literature

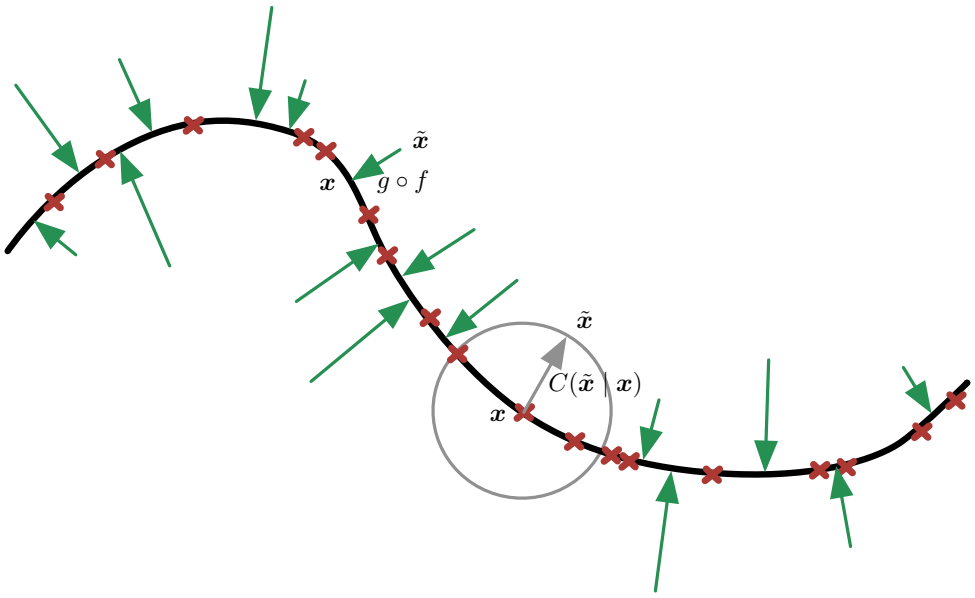
Denoising Autoencoder



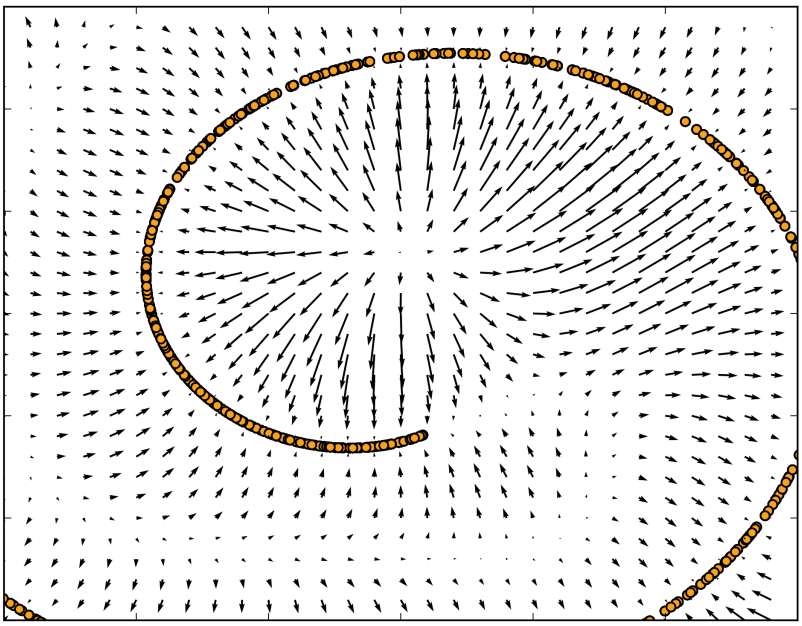
C: corruption process (introduce noise)

$$L = -\log p_{\text{decoder}}(x \mid h = f(\tilde{x}))$$

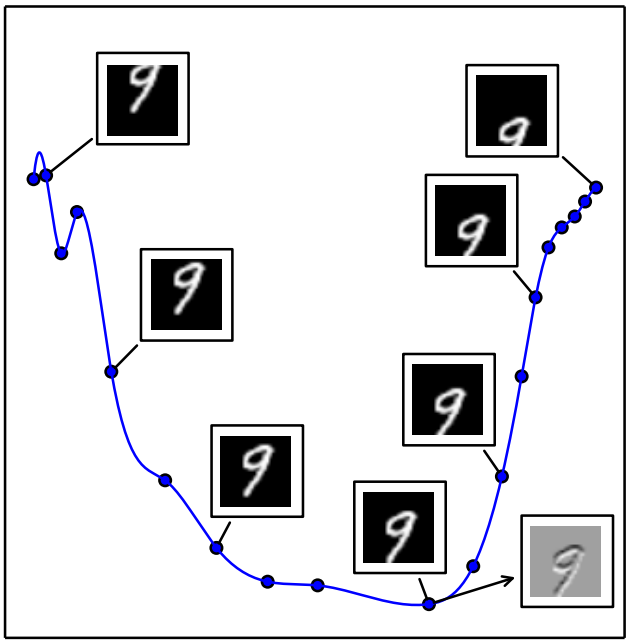
Denoising Autoencoder: Learn a Manifold



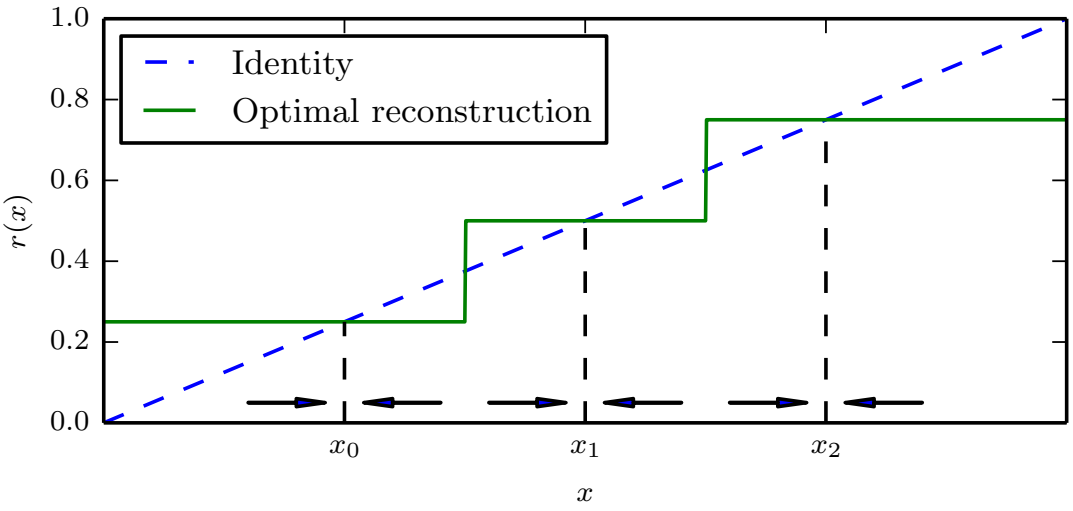
Vector Field Learned by a Denoising Autoencoder



Tangent Hyperplane of a Manifold



A Simple Example



Contractive Autoencoders

$$\Omega(\boldsymbol{h}) = \lambda \left\| \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_F^2$$ ← Frobenius norm (sum of squared elements)

Practical issues

- heavy computation of Jacobian for deep networks: build one layer at time!
- scale of decoder should be the same as encoder: impose weight matrix of decoder to be the transpose of weight matrix of encoder!