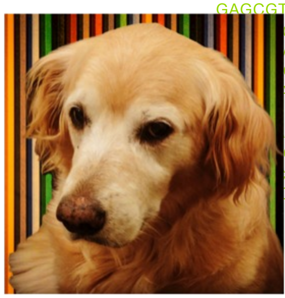
Autoencoder

Yao-Yi Chiang
Computer Science and Engineering
University of Minnesota
yaoyi@umn.edu

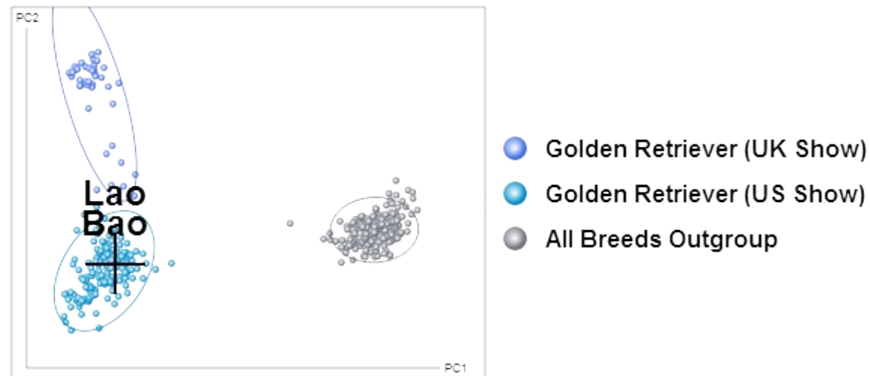
Dimension Reduction

- Principle Component Analysis (PCA)
  - Projecting the data into a new space using linear transformation
  - Using SVD or eigenvalue decomposition to find the new space



GTTTCTCGGTTNNCCGGCGAAAGAAAAGGTCAGAAAAAGACAGCCAAAAAAAGAAAAAGCCCC
AACCACCCCCGGGGAACCTTTTGGGGTTGGAGCCTTAGAATGAGTCTTTTAAGGTTCCGGTTA
GAGCGTGAAACAGAATCTGCCGGTCTCAAAAAAGGTGCGTCTCCCGGTCAGGGAAGGCCNNC
CTCCGAGTCAGAGCCACNNTTTCAGACACTTAGCCCCAGAGGGAATTTGCCTTTTAGT
ATTGGCCAAAGTCAGGGAGAGCGAGTCNNAGGGTTGGAGAAGGACAAGGCCCCTTC
GAGCCCCGGAATTACAAAGTCAGAGTAAGTTAAAGAGTCTCTCGGTCTCTCGGTTAGC
AGCCAAAGAGGGTCGGCCGGTTGGGGAATTGGGGCCAGAAAGTCTTGGTTAGAGAGT
GCCGGGGTTCCCCAGTCAATCTCTCTTTCTTAGGGTTCCTCAATTACGGAGCCAAAAC
GATCCAAGGAACCGGCCGGCCAAGGCCGGAATTGGAGCCGGAGAGAGCCGGAAAG
GGGGTTAGAGAGGGTCGGAAAATCAGAAAATTTCCCTTTTAAGGTTCCTGTCCCTTGG
TTNNAATCGGTCAGTTGGCCTCGGGGGGTTTTAACCAAAAAAAGTGAGGGAGAAACA
GGTCGGAGCGAGAGCCAACCTGAGAGTTCCAATTAGCCNNGGAACCAACCAA

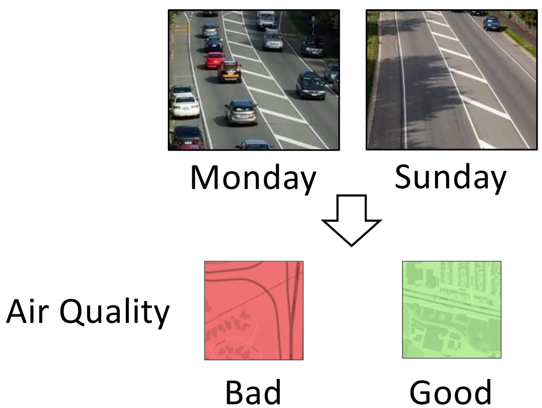
DNA Sequence

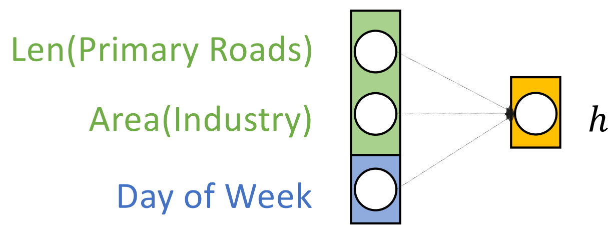


Linear VS. Non-Linear

- What if the underlying low dimensional structure is not linear?
  - PCA would not be able to find good representative basis vectors

- For example,



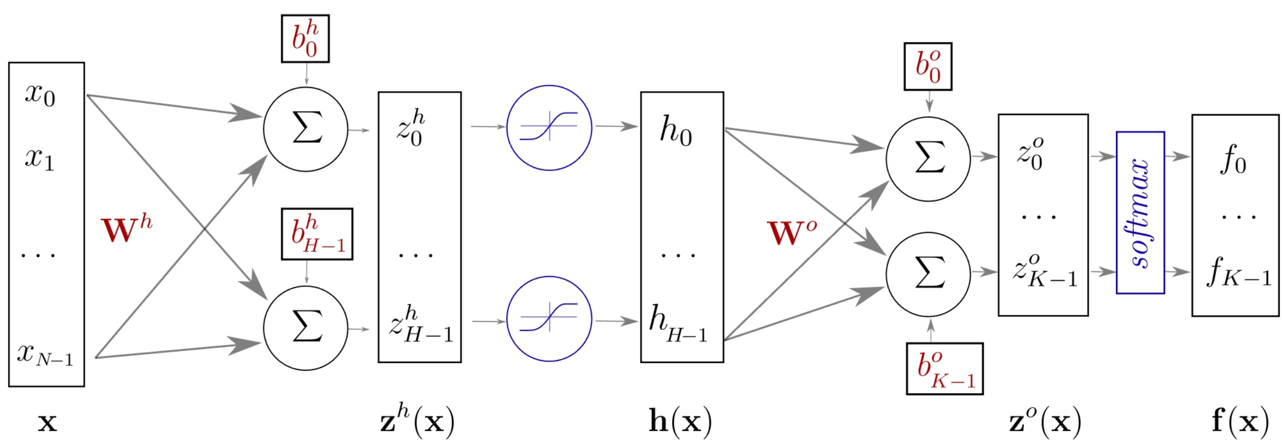


h can be a non-linear combination of three features

Linear VS. Non-Linear

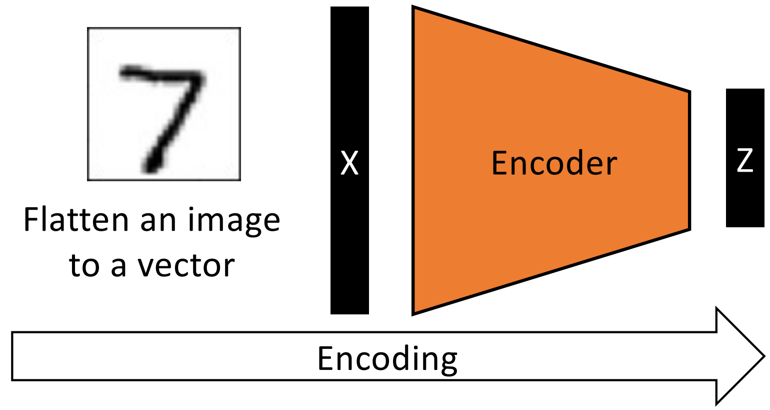
- What if the underlying low dimensional structure is not linear?
  - PCA would not be able to find good representative basis vectors

- Neural Networks?



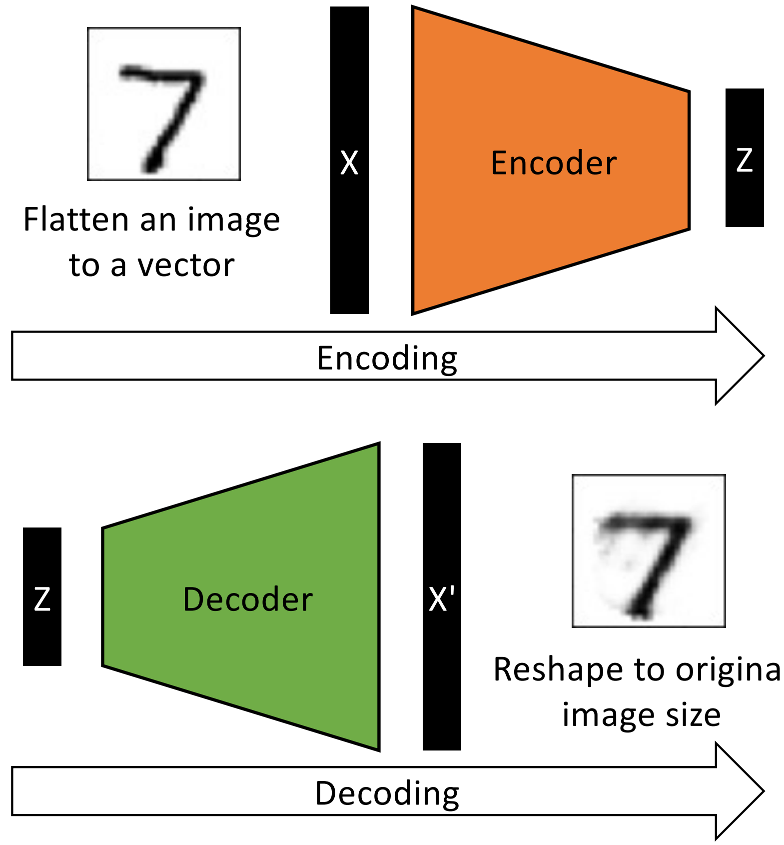
Autoencoder – Encoder and Decoder

- Encoder
  - Encoding the input X into a hidden representation Z



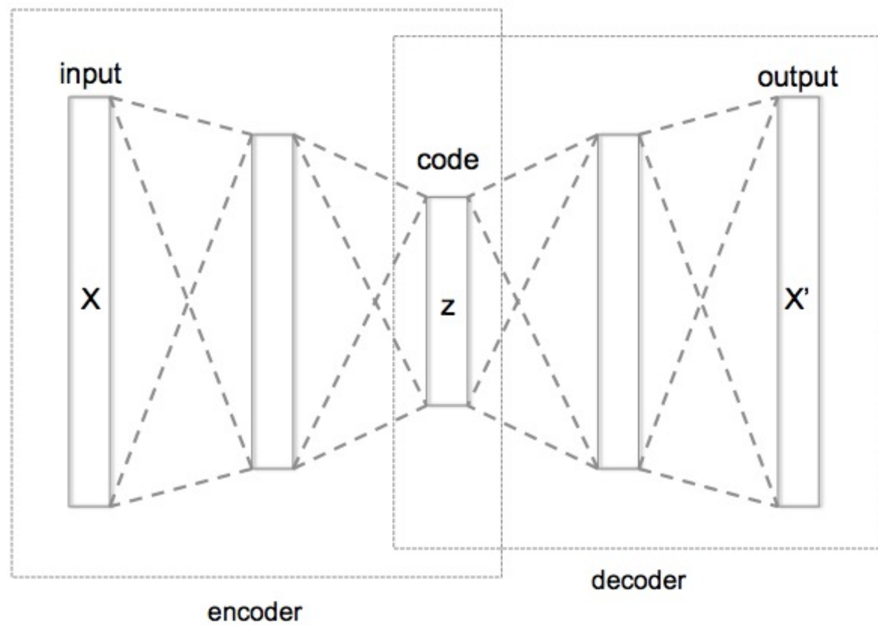
Autoencoder – Encoder and Decoder

- Encoder
  - Encoding the input X into a hidden representation Z
- Decoder
  - Decoding the input X' from the hidden representation Z



Autoencoder – Encoder and Decoder

- Encoder
  - Encoding the input X into a hidden representation Z
- Decoder
  - Decoding the input X' from the hidden representation Z
- Usually, $\mathrm{Dim}(Z) < \mathrm{Dim}(X)$, also called undercomplete AE

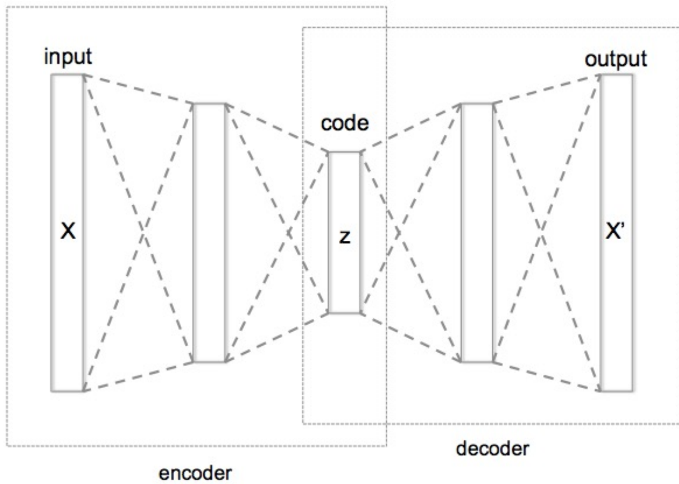


Autoencoder – Encoder and Decoder

- Encoder
  - $Z = f(X) = \sigma(WX + b)$
- Decoder
  - $X' = g(Z) = \sigma'(W'Z + b')$

- σ and σ' are activation functions
- σ' depends on the input type
  - e.g., if the inputs have values between 0 and 1, we can use a Sigmoid function

For example:
W 32x64; X 64x1,000;
Z 32x1,000;
W' 64x32; X' 64x1,000



Autoencoder – Objective Function

- $X' = f(g(X))$
- The model is trained to minimize a certain loss function which will ensure that X' is close to X

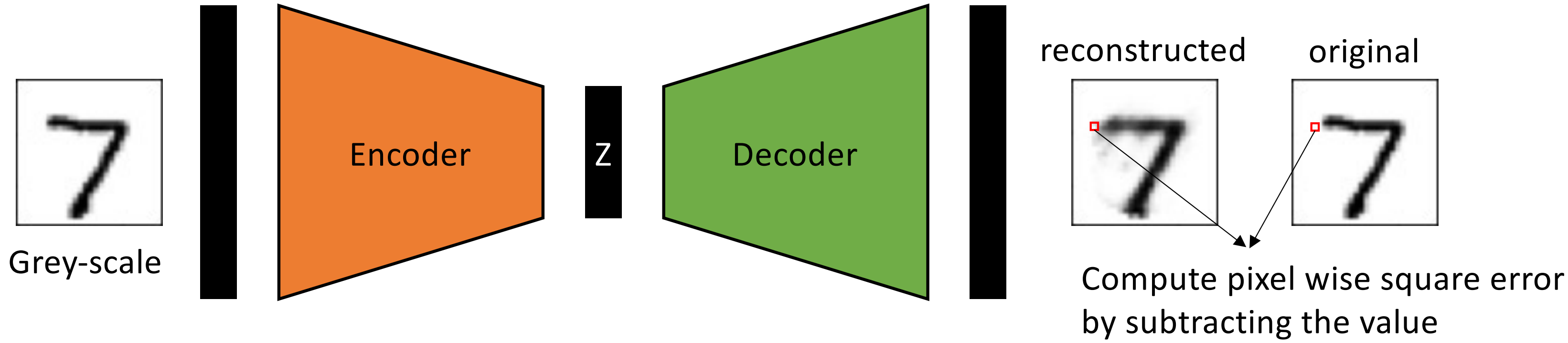
- Loss function depends on the inputs

Autoencoder – Objective Function

- When the inputs are real values, we can use Mean Square Error (MSE) as the loss function

$$\min_{W, W', b, b'} \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (x'_{ij} - x_{ij})^2$$

where m is the number of samples, and n is the number of features

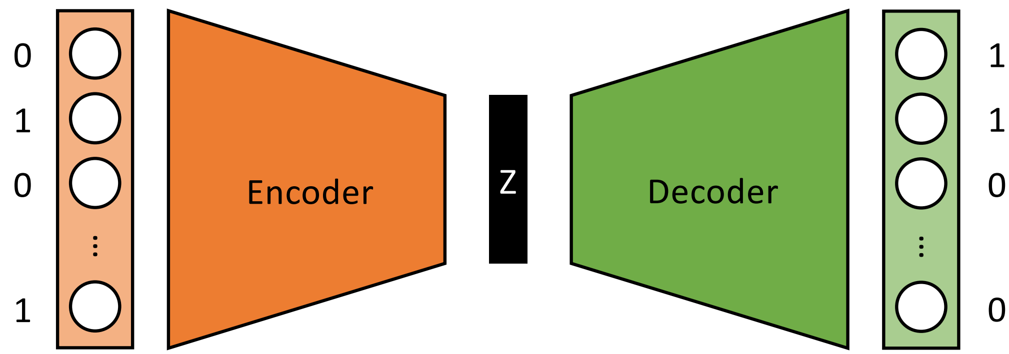


Autoencoder – Objective Function

- When the inputs are binary, we can use Binary Cross Entropy (BCE) as the loss function

$$\min_{W, W', b, b'} \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} -(x_{ij} \log x'_{ij} + (1 - x_{ij}) \log(1 - x'_{ij}))$$

where m is the number of samples, and n is the number of features



Learn more about BCE: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a

Link Between PCA and Autoencoder

- The encoder part of an autoencoder is equivalent to PCA if
  - the encoder is a one-layer linear transformation, no bias term
  - the decoder is a one-layer linear transformation, no bias term
  - using the squared error loss function
  - normalizing the input to 0 mean along each dimension
  - also divide each input element by the square root of m so that $\tilde{X}^T\tilde{X}$ is the covariance matrix of the 0 mean data

$$\tilde{x}_{ij} = \frac{1}{\sqrt{m}}\left(x_{ij} - \frac{1}{m}\sum_{k=1}^{m} x_{kj}\right)$$

where x_i is the input, j is the feature dimension, and m is the number of samples

Link Between PCA and Autoencoder

- We will show that if
  - using a linear decoder and a squared error loss function
  - the optimal solution to the following objective function is obtained when using a linear encoder

$$\min_{W, W', b, b'} \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (x'_{ij} - \tilde{x}_{ij})^2$$

- The above loss function is equivalent to

$$\min(\|\tilde{X} - ZW'\|_F)^2$$

where $\|A\|_F$ is the Frobenius Norm of matrix A, $\|A\|_F \equiv \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$

Link Between PCA and Autoencoder

- The optimal solution to the problem

$$\min(\|\tilde{X} - ZW'\|_F)^2$$

is given by

$$\tilde{X} = ZW' = U\Sigma V^T$$ Recall: from SVD

where U and V are orthogonal matrices and Σ is a diagonal matrix with non-negative values on diagonal

orthogonal matrices:
$(V^T V = \mathrm{I})$
$(V^T = V^{-1})$

- By matching variables one possible solution is

$$Z = U\Sigma$$
$$W' = V^T$$

Link Between PCA and Autoencoder

- We will now show that Z is a linear encoding and find an expression for the encoder weight W

$$Z = U\Sigma$$

Link Between PCA and Autoencoder

- We will now show that Z is a linear encoding and find an expression for the encoder weight W

$$Z = U\Sigma$$
$$Z = (\tilde{X}\tilde{X}^T)(\tilde{X}\tilde{X}^T)^{-1}U\Sigma$$

Link Between PCA and Autoencoder

- We will now show that Z is a linear encoding and find an expression for the encoder weight W

$$Z = U\Sigma$$
$$Z = (\tilde{X}\tilde{X}^T)(\tilde{X}\tilde{X}^T)^{-1}U\Sigma$$
$$Z = \left(\tilde{X}V\Sigma^T U^T\right)(U\Sigma V^T V\Sigma^T U^T)^{-1}U\Sigma \quad \tilde{X} = ZW' = U\Sigma V^T$$

Link Between PCA and Autoencoder

- We will now show that Z is a linear encoding and find an expression for the encoder weight W

$$Z = U\Sigma$$
$$Z = (\tilde{X}\tilde{X}^T)(\tilde{X}\tilde{X}^T)^{-1}U\Sigma$$
$$Z = \left(\tilde{X}V\Sigma^T U^T\right)(U\Sigma V^T V\Sigma^T U^T)^{-1}U\Sigma \qquad \tilde{X} = ZW' = U\Sigma V^T$$
$$Z = \tilde{X}V\Sigma^T U^T(U\Sigma\Sigma^T U^T)^{-1}U\Sigma \qquad (V^T V = \mathrm{I})$$

Link Between PCA and Autoencoder

- We will now show that Z is a linear encoding and find an expression for the encoder weight W

$$Z = U\Sigma$$

$$Z = (\tilde{X}\tilde{X}^T)(\tilde{X}\tilde{X}^T)^{-1}U\Sigma$$

$$Z = \left(\tilde{X}V\Sigma^T U^T\right)(U\Sigma V^T V\Sigma^T U^T)^{-1}U\Sigma \qquad \tilde{X} = ZW' = U\Sigma V^T$$

$$Z = \tilde{X}V\Sigma^T U^T(U\Sigma\Sigma^T U^T)^{-1}U\Sigma \qquad (V^T V = \mathrm{I})$$

$$Z = \tilde{X}V\Sigma^T U^T U(\Sigma\Sigma^T)^{-1}U^T U\Sigma \qquad ((ABC)^{-1} = C^{-1}B^{-1}A^{-1})$$

$$(U^T = U^{-1})$$

Link Between PCA and Autoencoder

- We will now show that Z is a linear encoding and find an expression for the encoder weight W

$$Z = U\Sigma$$

$$Z = (\tilde{X}\tilde{X}^T)(\tilde{X}\tilde{X}^T)^{-1}U\Sigma$$

$$Z = \left(\tilde{X}V\Sigma^T U^T\right)(U\Sigma V^T V\Sigma^T U^T)^{-1}U\Sigma \qquad \tilde{X} = ZW' = U\Sigma V^T$$

$$Z = \tilde{X}V\Sigma^T U^T (U\Sigma\Sigma^T U^T)^{-1}U\Sigma \qquad (V^T V = \mathrm{I})$$

$$Z = \tilde{X}V\Sigma^T U^T U(\Sigma\Sigma^T)^{-1}U^T U\Sigma \qquad ((ABC)^{-1} = C^{-1}B^{-1}A^{-1})$$

$$Z = \tilde{X}V\Sigma^T(\Sigma^T)^{-1}(\Sigma)^{-1}\Sigma \qquad (U^T = U^{-1})$$

Link Between PCA and Autoencoder

- We will now show that Z is a linear encoding and find an expression for the encoder weight W

$$Z = U\Sigma$$
$$Z = (\tilde{X}\tilde{X}^T)(\tilde{X}\tilde{X}^T)^{-1}U\Sigma$$
$$Z = \left(\tilde{X}V\Sigma^T U^T\right)(U\Sigma V^T V\Sigma^T U^T)^{-1}U\Sigma \qquad \tilde{X} = ZW' = U\Sigma V^T$$
$$Z = \tilde{X}V\Sigma^T U^T(U\Sigma\Sigma^T U^T)^{-1}U\Sigma \qquad (V^T V = \mathrm{I})$$
$$Z = \tilde{X}V\Sigma^T U^T U(\Sigma\Sigma^T)^{-1}U^T U\Sigma \qquad ((ABC)^{-1} = C^{-1}B^{-1}A^{-1})$$
$$Z = \tilde{X}V\Sigma^T(\Sigma^T)^{-1}(\Sigma)^{-1}\Sigma = \tilde{X}V \qquad (U^T = U^{-1})$$

Link Between PCA and Autoencoder

- We will now show that Z is a linear encoding and find an expression for the encoder weight W

$$Z = U\Sigma$$
$$Z = (\tilde{X}\tilde{X}^T)(\tilde{X}\tilde{X}^T)^{-1}U\Sigma$$
$$Z = \left(\tilde{X}V\Sigma^T U^T\right)(U\Sigma V^T V\Sigma^T U^T)^{-1}U\Sigma \qquad \tilde{X} = ZW' = U\Sigma V^T$$
$$Z = \tilde{X}V\Sigma^T U^T(U\Sigma\Sigma^T U^T)^{-1}U\Sigma \qquad (V^T V = \mathrm{I})$$
$$Z = \tilde{X}V\Sigma^T U^T U(\Sigma\Sigma^T)^{-1}U^T U\Sigma \qquad ((ABC)^{-1} = C^{-1}B^{-1}A^{-1})$$
$$Z = \tilde{X}V\Sigma^T(\Sigma^T)^{-1}(\Sigma)^{-1}\Sigma = \tilde{X}V \qquad (U^T = U^{-1})$$

- Thus, Z is a linear transformation of $\tilde{X}$ and $W = V$

Link Between PCA and Autoencoder

- We have encoder $W = V$
- With SVD, $\tilde{X} = U\Sigma V^T$, the columns of V are the orthonormal eigenvectors of $\tilde{X}^T\tilde{X}$

$$\tilde{X}^T\tilde{X} = V\Sigma^T U^T\ U\Sigma V^T$$

$$\tilde{X}^T\tilde{X} = V\Sigma^T\Sigma V^T \qquad (V^T = V^{-1})$$

$$\tilde{X}^T\tilde{X}\ V = V(\Sigma^T\Sigma)$$

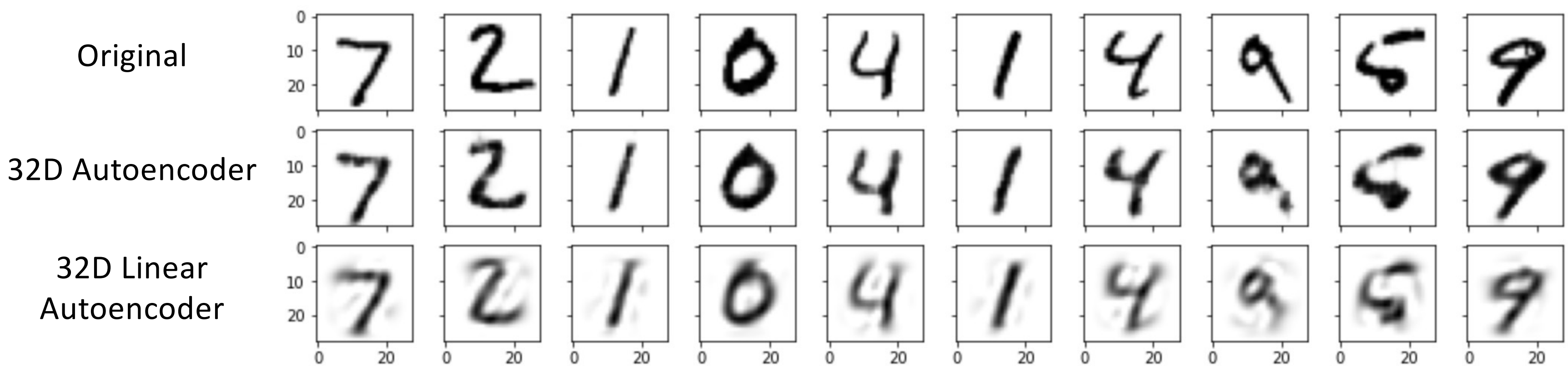
https://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm#:~:text=The%20SVD%20represents%20an%20expansion,up%20the%20columns%20of%20U.

Link Between PCA and Autoencoder

- We have encoder $W = V$
- With SVD, $\tilde{X} = U\Sigma V^T$, the columns of V are the orthonormal eigenvectors of $\tilde{X}^T\tilde{X}$
- From PCA, we know that the projection matrix is the matrix of eigenvectors of the covariance matrix
- Since the entries of X are normalized by $\tilde{x}_{ij} = \frac{1}{\sqrt{m}}\left(x_{ij} - \frac{1}{m}\sum_{k=1}^{m} x_{kj}\right)$, $\tilde{X}^T\tilde{X}$ is the covariance matrix
- Thus, the linear encoder W and the projection matrix for PCA could be the same

Link Between PCA and Autoencoder

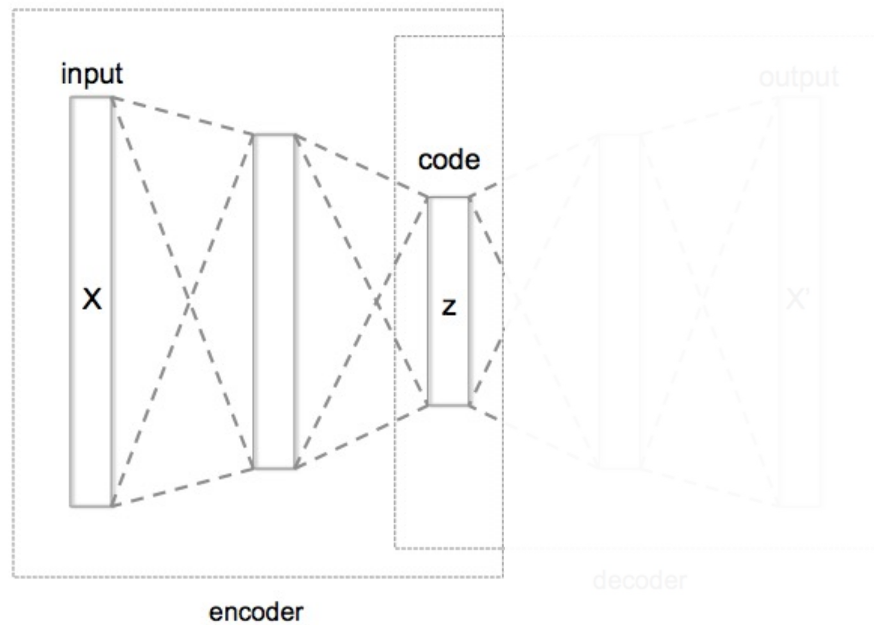
- Nonlinear autoencoder can learn more powerful codes for a given dimensionality (e.g., 32), compared with linear autoencoder (PCA)



Autoencoder Applications

- Using the hidden representation as the input to classic machine learning methods e.g., SVM, KNN
- The latent space can be used for visualization (e.g., clustering)
- Anomaly detection

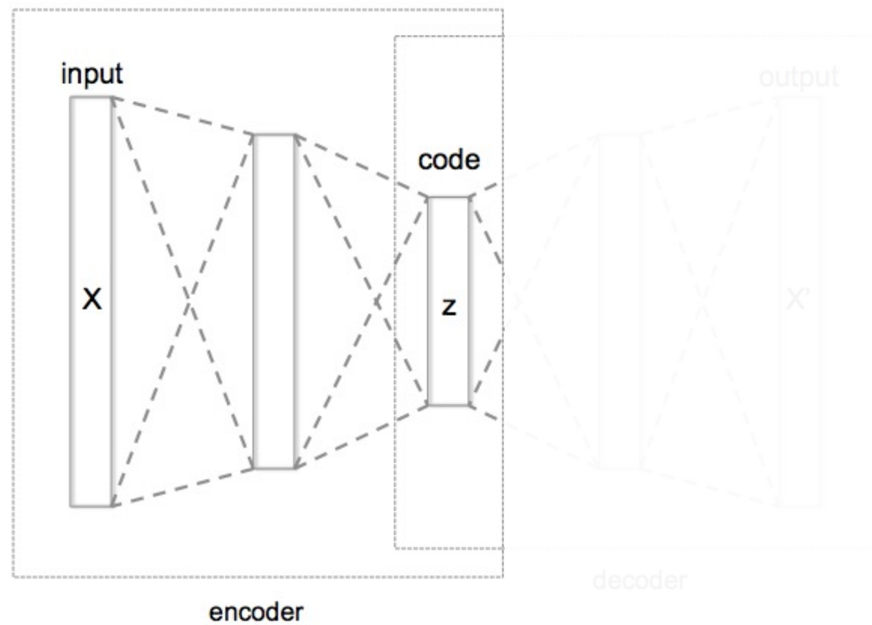
After training, disregarding the decoder



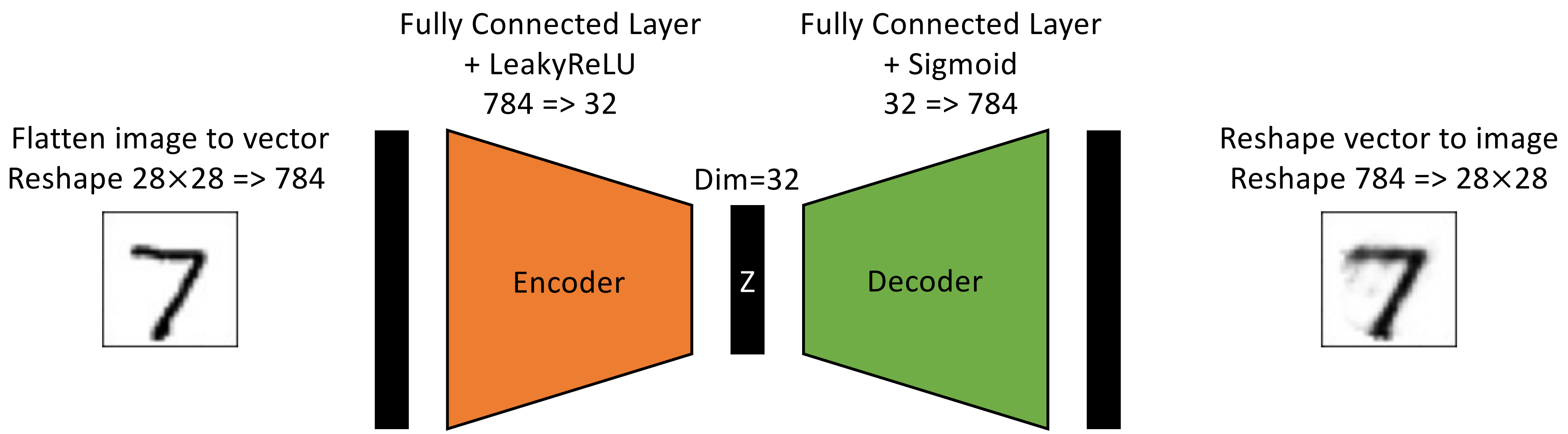
Autoencoder Applications

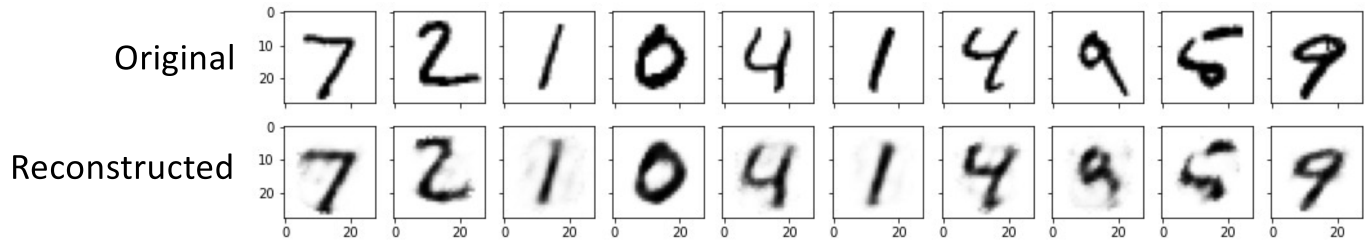
- Training an autoencoder on a large dataset, then fine tune the encoder part on your own smaller dataset and/or provide your own output layers (e.g., classification)

After training, disregarding the decoder



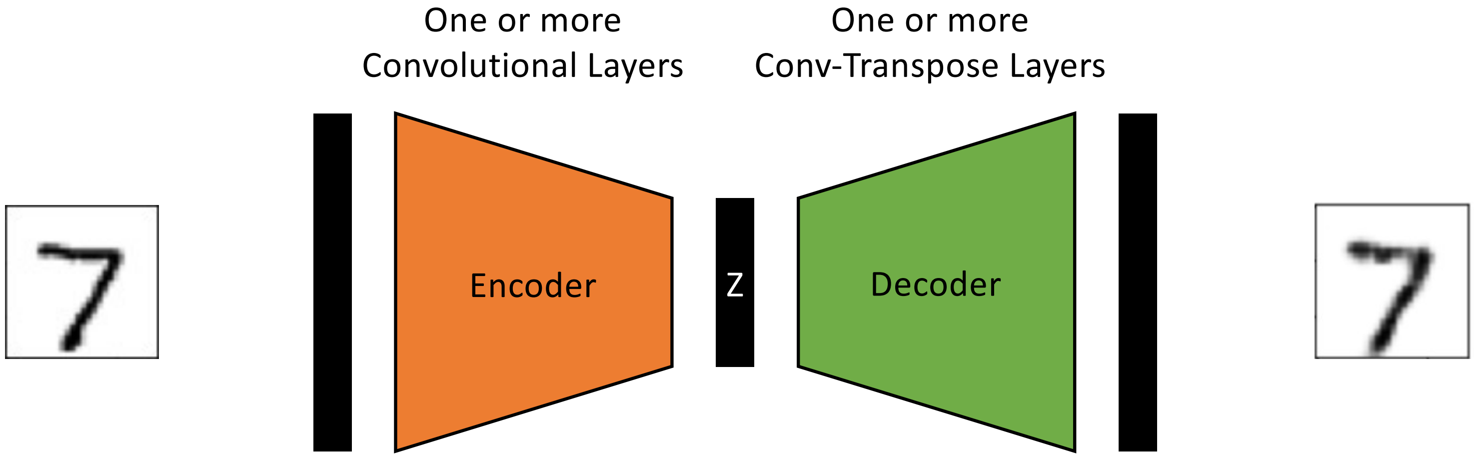
A Fully-Connected Autoencoder on Images

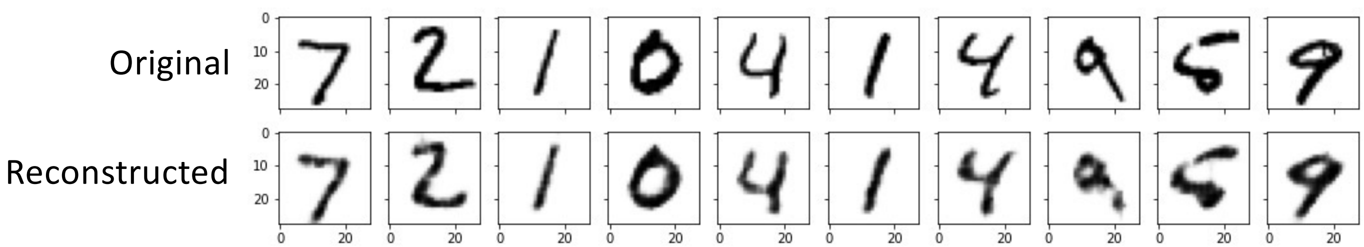




Code: https://github.com/rasbt/stat479-deep-learning-ss19/tree/master/L15_autoencoder/code

A Convolutional Autoencoder on Images

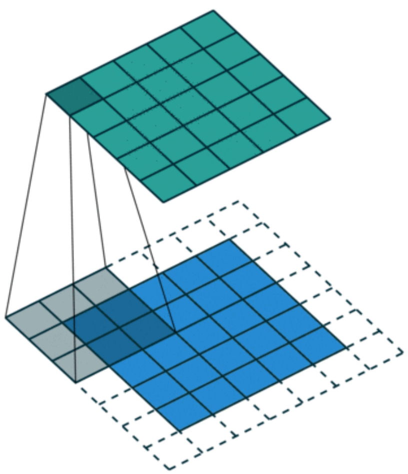




Code: https://github.com/rasbt/stat479-deep-learning-ss19/tree/master/L15_autoencoder/code

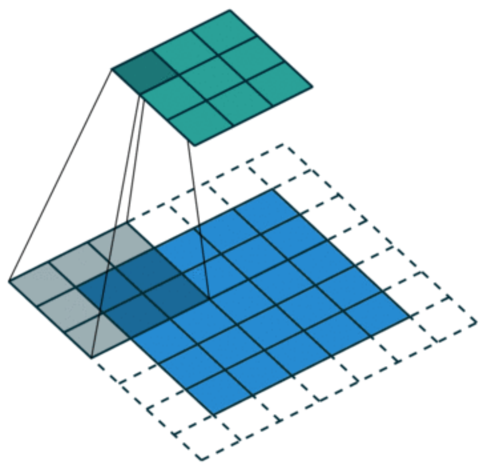
Regular and Transposed Convolution

Regular Convolution
filter size = 3 x 3
padding = 1
stride = 1



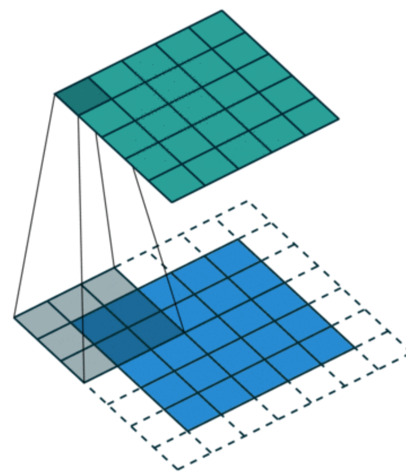
input image size = 5 x 5
output image size =5 x 5

Regular Convolution
filter size = 3 x 3
padding = 1
stride = 2



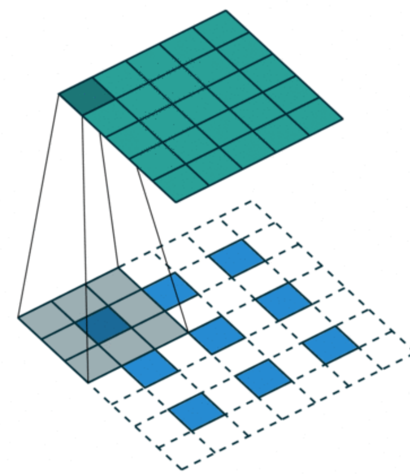
input image size = 5 x 5
output image size = 3 x 3

Transposed Convolution
filter size = 3 x 3
padding = 1
stride = 1



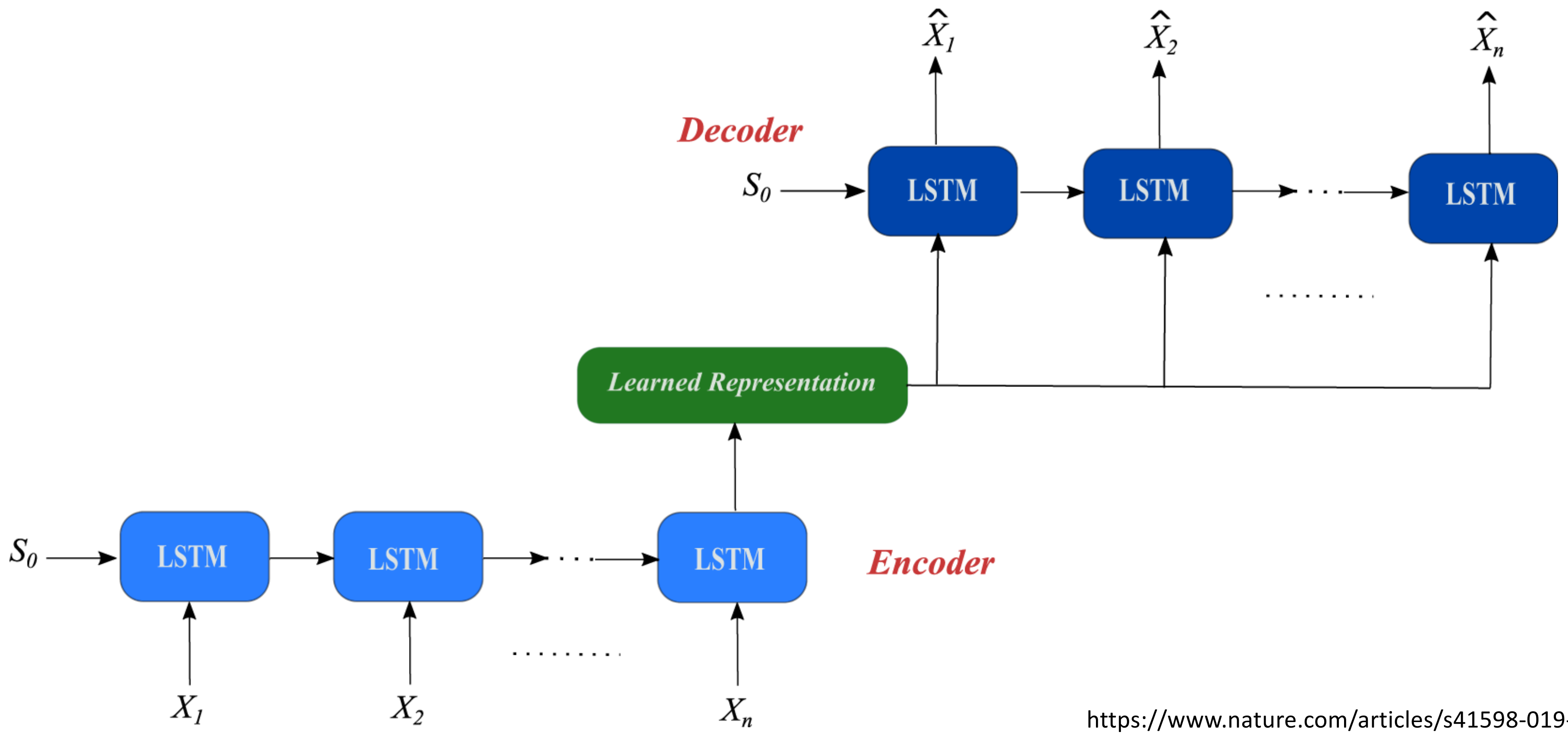
input image size = 5 x 5
output image size =5 x 5

Transposed Convolution
filter size = 3 x 3
padding = 1
stride = 2



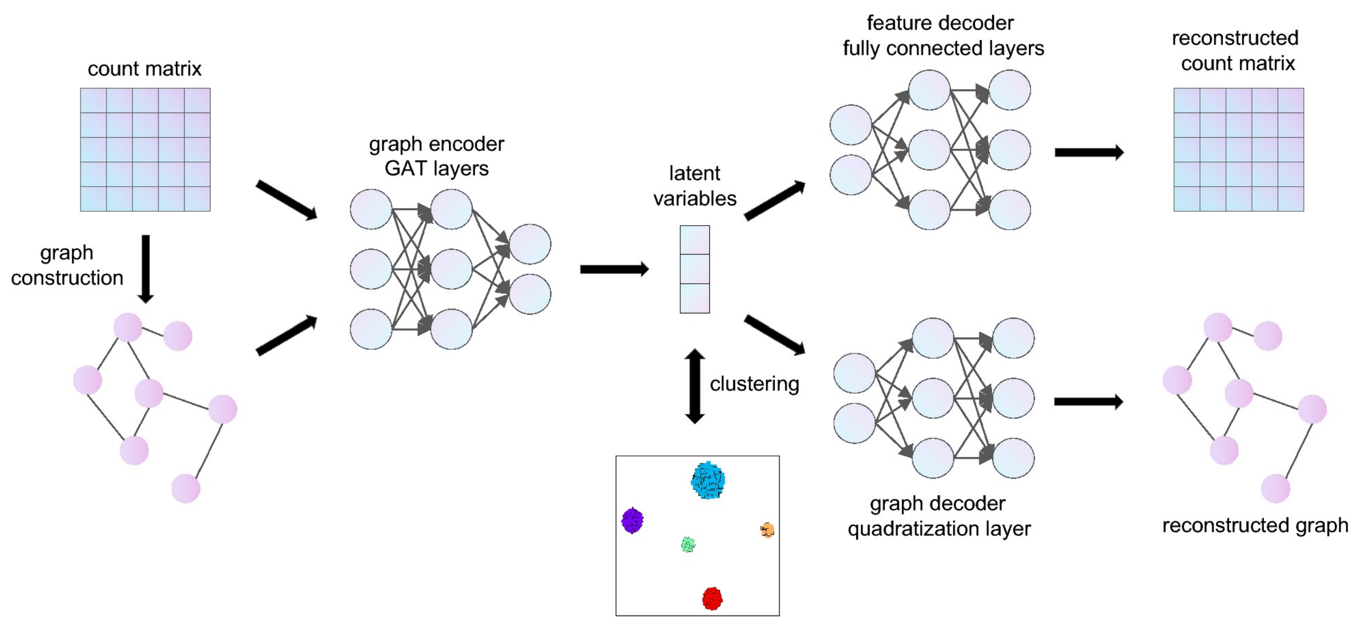
input image size = 3 x 3
output image size = 5 x 5

RNN Autoencoder for Sequence Data



https://www.nature.com/articles/s41598-019-55320-6

GCN Autoencoder for Graph



- The normalized count matrix represents the gene expression level in each cell. The adjacency matrix is constructed by connecting each cell to its K nearest neighbors.
- The encoder takes the count matrix and the adjacency matrix as inputs and generates low-dimensional latent variables.
- The feature decoder reconstructs the count matrix.
- The graph decoder reconstructs the adjacency matrix.
- Clustering is performed on the latent variables.
-

https://www.nature.com/articles/s41598-021-99003-7/figures/1

Regularized Autoencoder

- Autoencoders are trained to preserve as much information as possible of the input data with smaller vectors

Regularized Autoencoder

- Autoencoders are trained to preserve as much information as possible of the input data with smaller vectors

- Moreover, autoencoders are to create meaningful representations of the input
  - More neurons (i.e., hidden size) than the input size allow the network to compute powerful representations of the input

Regularized Autoencoder

- However, when the hidden dimension is higher than the input
  - No compression needed, also called overcomplete AE
  - The network trivially learns to just copy, not learning meaningful features

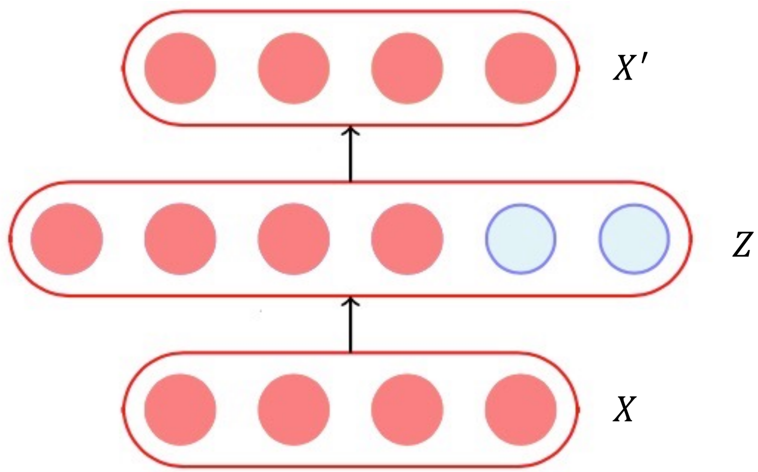


Image by Mitesh M. Khapra

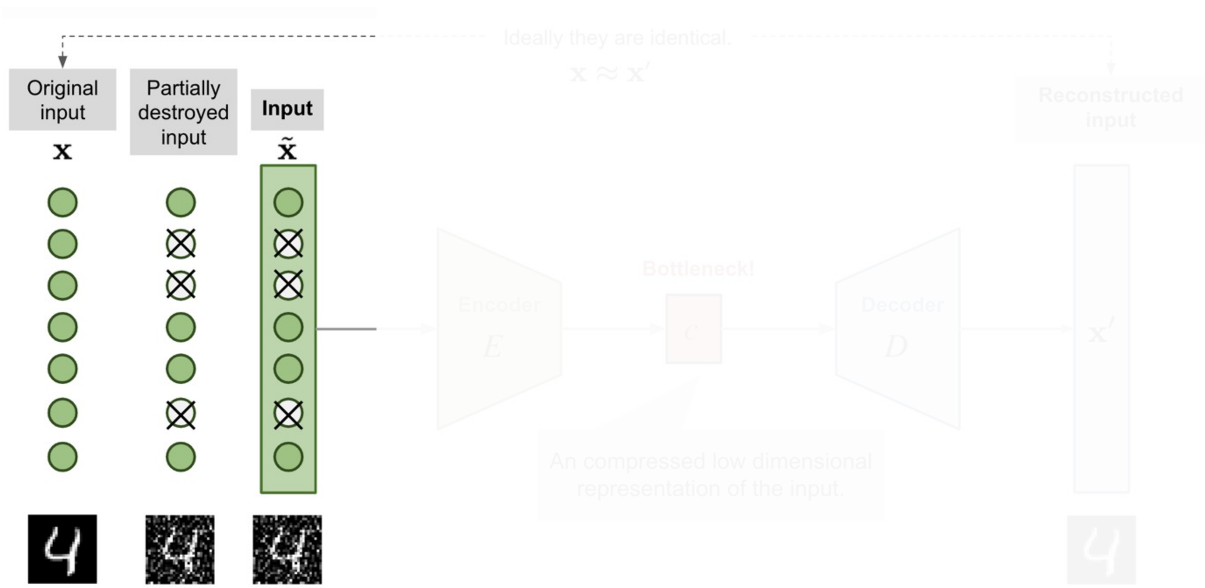
Regularized Autoencoder

- Regularized autoencoders aim to avoid overfitting and improve robustness
  - Denoise Autoencoder [1]
  - Sparse Autoencoder [2]

[1] Pascal Vincent, et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.". Journal of machine learning research 11.Dec (2010): 3371-3408.
[2] Ng, Andrew. "Sparse autoencoder." CS294A Lecture notes 72.2011 (2011): 1-19.

Denoise Autoencoder

- The input is partially corrupted by adding noises to or masking some values of the input vector in a stochastic manner



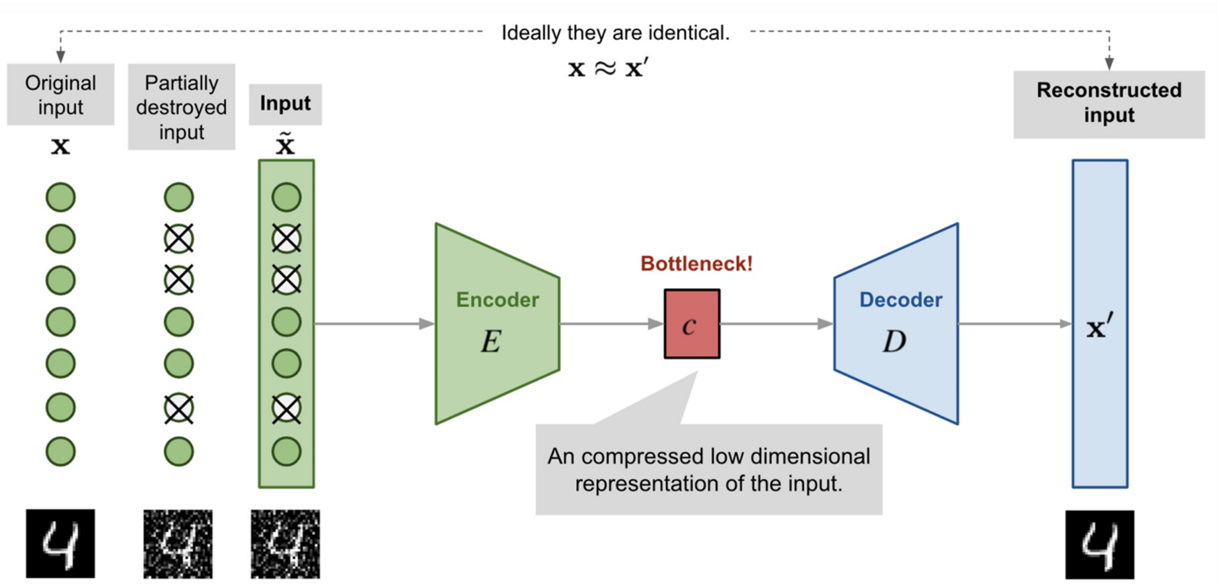
$$\tilde{x}^{(i)} \sim \mathcal{M}_D\left(\tilde{x}^{(i)} \middle| x^{(i)}\right)$$

where $\mathcal{M}_D$ defines the mapping from the true data samples to the noisy or corrupted ones, e.g., masking noise, Gaussian noise

Denoising AE architecture by Lilian Weng

Denoise Autoencoder

- Then the model is trained to recover the original input (note: not the corrupt one)



$$\min_{\theta, \phi} \frac{1}{n} \sum_{i=1}^{n} (x^{(i)} - f_\theta(g_\phi(\tilde{x}^{(i)})))^2$$

Denoising AE architecture by Lilian Weng

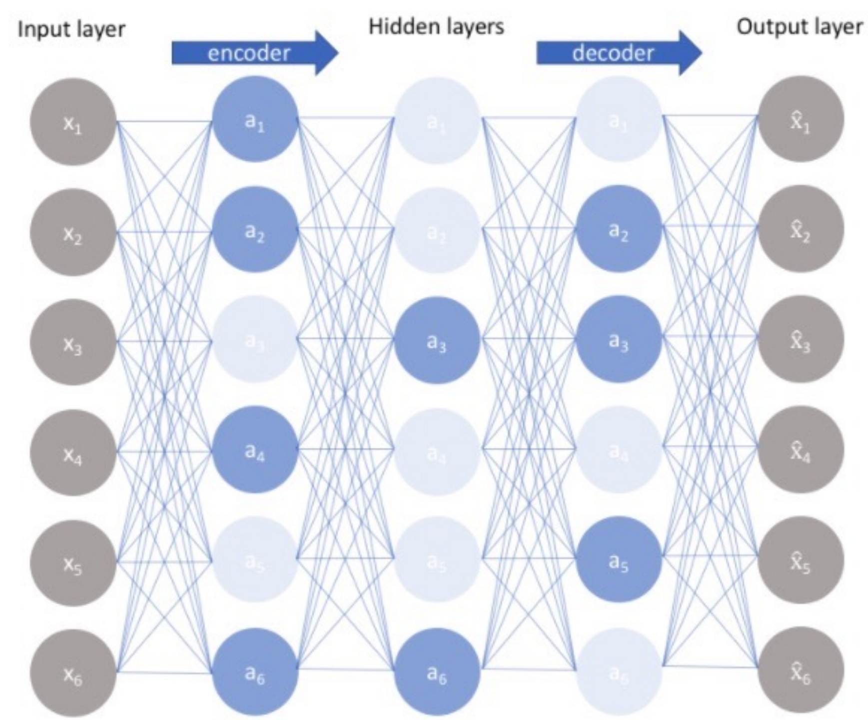
Denoise Autoencoder – Experiment Results

- The model learns a combination of many input dimensions to recover the denoised version rather than to overfit one dimension, which helps learn robust latent representation

Original input, corrupted data, and reconstructed data. Copyright by opendeep.org.

Sparse Autoencoder

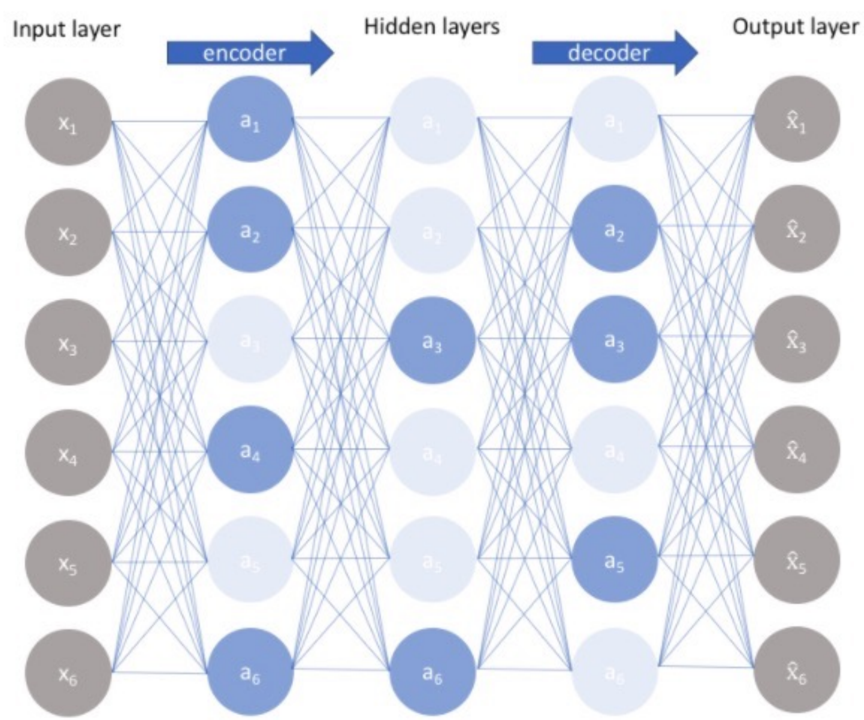
- Sparse autoencoder forces the model to only have a small number of hidden units being activated at the same time



Sparse Autoencoder image by [Syoya Zhou](#)

Sparse Autoencoder

- Sparse autoencoder forces the model to only have a small number of hidden units being activated at the same time
- Loss = reconstruction loss + regularization loss
- There are two ways to construct sparsity penalty
  - L1 regularization
  - KL-divergence



Sparse Autoencoder image by Syoya Zhou

Sparse Autoencoder with KL-divergence

- Let's say there are s_l neurons in the l^{th} hidden layer and the activation function for the j^{th} neuron in this layer is labelled as $a_j^{(l)}(.)$

Sparse Autoencoder with KL-divergence

- Let's say there are s_l neurons in the l^{th} hidden layer and the activation function for the j^{th} neuron in this layer is labelled as $a_j^{(l)}(.)$
- The average activation of neuron $\hat{\rho}_j$ is expected to be a small number ρ, known as *sparsity parameters*

$$\hat{\rho}_j^{(l)} = \frac{1}{n}\sum_{i=1}^{n}[a_j^{(l)}(x^{(i)})] \approx \rho$$

$\left[a_j^{(l)}\left(x^{(i)}\right)\right] = 1$ if the neuron is activated, 0 otherwise
n is the number of input sample

Sparse Autoencoder with KL-divergence

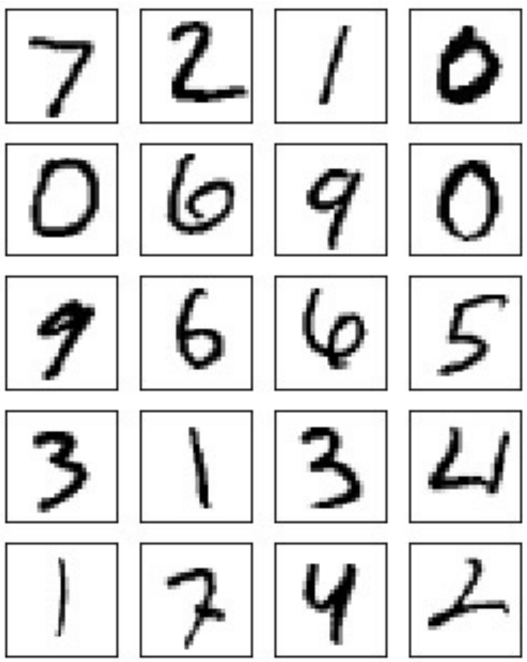
- The KL-divergence measures the difference between two probability distributions,[1] one with mean ρ and the other with mean $\rho_j^{(l)}$

$$L_{SAE} = L_{MSE} + \beta \sum_{l=1}^{L} \sum_{j=1}^{s_l} D_{KL}(\rho \,||\, \hat{\rho}_j^{(l)})$$

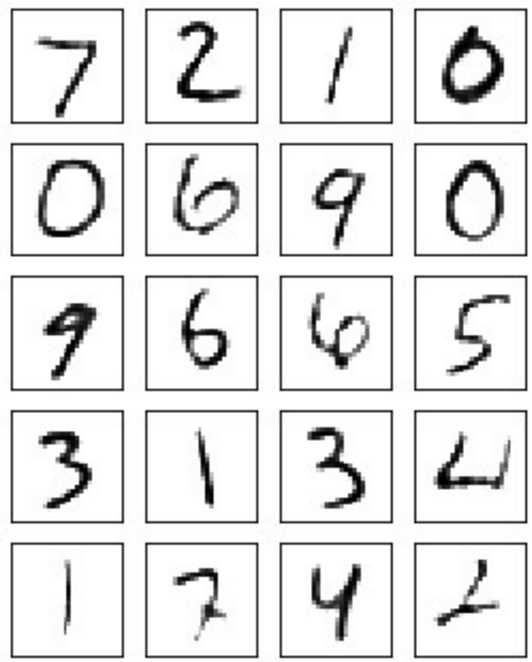
- The hyperparameter β controls how strong the penalty applying on the sparsity loss

1. The probability distribution here can be viewed as Bernoulli distribution, the discrete probability distribution of a random variable which takes the value 1 with probability p and the value 0 with probability q = p -1: https://en.wikipedia.org/wiki/Bernoulli_distribution

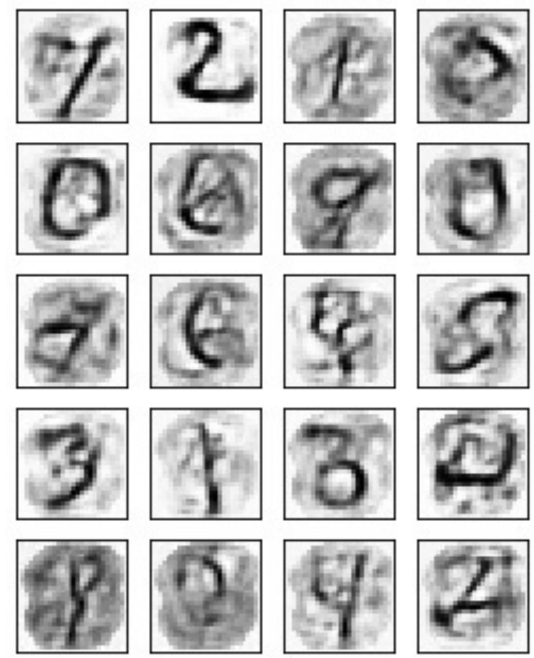
Sparse Autoencoder – Experiment Results



Original input



Reconstructed data



Reconstructed from latent space with zeroed "inactive" neurons (activation < 0.5)

code: https://github.com/AntonP999/Sparse_autoencoder/blob/master/Sparse_autoencoder.ipynb

Other Autoencoders

- Variational Autoencoder (VAE)
- Beta-VAE

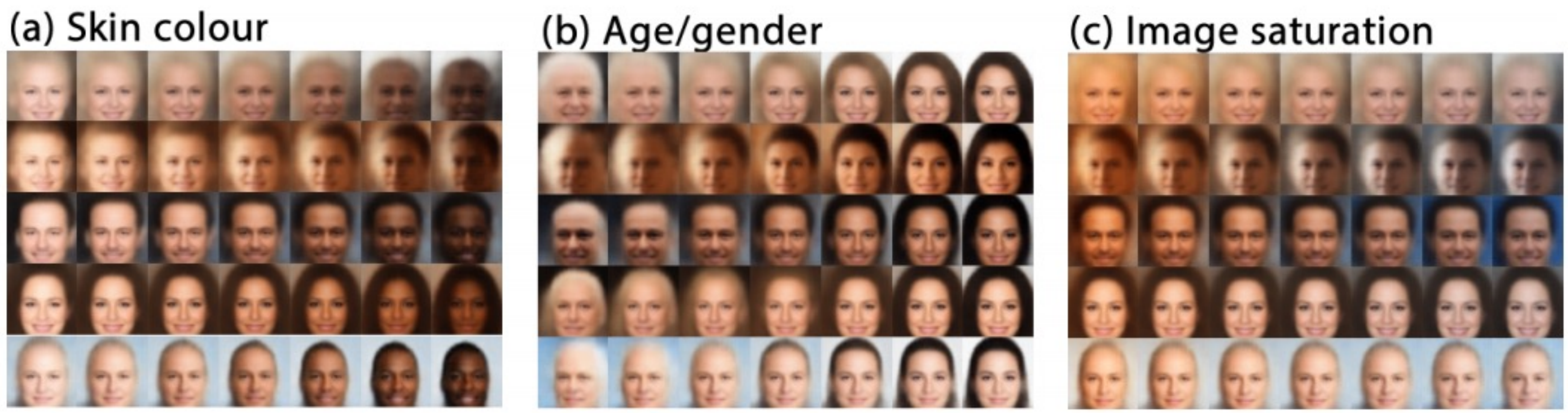


Figure 4: **Latent factors learnt by β-VAE on celebA:** traversal of individual latents demonstrates that β-VAE discovered in an unsupervised manner factors that encode skin colour, transition from an elderly male to younger female, and image saturation.

Higgins, I. et al.(2016). beta-vae: Learning basic visual concepts with a constrained variational framework.

Autoencoder Summary

- Autoencoder is a neural network architecture designed to learn an identity function in an unsupervised way to reconstruct the original input
- Autoencoders can compress the data in a non-linear way
- Autoencoders create meaningful representations of the input
- Autoencoders with regularization strategy overcome overfitting and improve the robustness when there are more neurons in the network than the input
- Many different types of autoencoder structures exist to accommodate various data representations

Acknowledgements

- Deep learning slides adapted from https://m2dsupsdlclass.github.io/lectures-labs/ by Olivier Grisel and Charles Ollion (CC-By 4.0 license)
- Gil, Yolanda (Ed.) Introduction to Computational Thinking and Data Science. Available from http://www.datascience4all.org
- https://lilianweng.github.io/posts/2018-08-12-vae/