Dimensionality Reduction

COMP 551 Applied Machine Learning

Isabeau Prémont-Schwarz
School of Computer Science
McGill University

Fall 2024

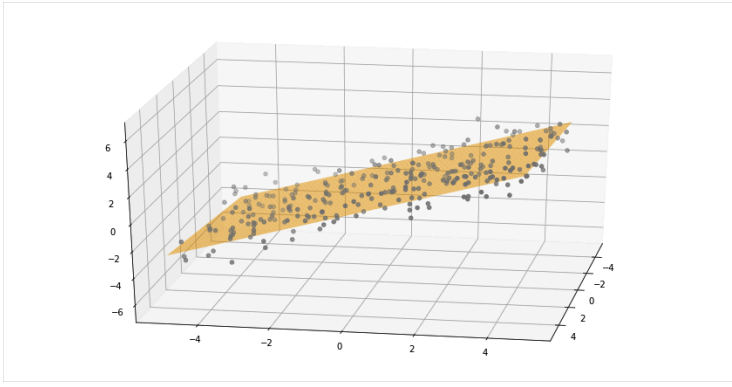
Outline

Acknowledgement

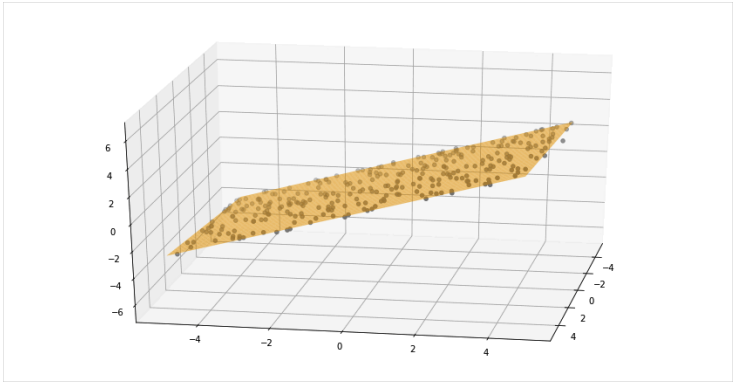
- The slides are adapted from Prof. Yue Li's slides.

Outline

Data Manifold

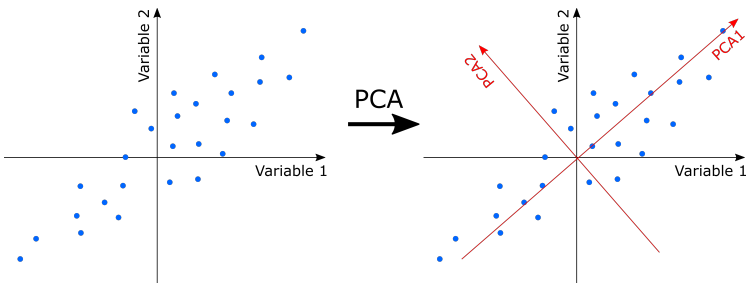


Data Manifold



Principal Component Analysis

PCA main idea: find the directions which encode most of the difference (variance) between datapoints.



Principal Component Analysis

- Suppose we have an unlabelled dataset $\mathbf{X} \in \mathbb{R}^{D \times N}$ for D features and N examples.
- We would like to approximate each data point $\mathbf{x}_n \in \mathbb{R}^{D \times 1}$ by a low dimensional representation $\mathbf{z}_n \in \mathbb{R}^{K \times 1}$, where $K \leq D$.
- The variable $\mathbf{z}_n$ is known as the **latent factor**.
- The error produced by this approximation is called **reconstruction error**:

$$\begin{aligned}\mathcal{L}(\mathbf{W}) &= \frac{1}{N}\sum_{n=1}^{N} ||\mathbf{x}_n - \text{decode}(\text{encode}(\mathbf{x}_n; \mathbf{W}); \mathbf{W})||_2^2 \\ &\triangleq \frac{1}{N}\sum_{n=1}^{N} ||\mathbf{x}_n - \mathbf{W}\mathbf{z}_n||_2^2\end{aligned}$$

where in the context of PCA:

- we assume a *linear* encoder and decoder
- $\mathbf{Z} \in \mathbb{R}^{K \times N}$ is also known as the **loading matrix**.
- $\mathbf{W} \in \mathbb{R}^{D \times K}$ is called the **basis matrix**. It is *orthogonal* matrix: $\mathbf{W}^\top \mathbf{W} = \mathbf{I}$, which means that $\mathbf{w}_k^\top \mathbf{w}_{k'} = 1$ for $k = k'$ or 0 for $k \neq k'$.

PCA derivation

In what follows we assume $\mathbb{E}[\mathbf{x}] = \mathbf{0}$, if it is not true, we can simply redefine $\mathbf{x}_n = \hat{\mathbf{x}}_n - \mathbb{E}[\hat{\mathbf{x}}]$, where $\hat{\mathbf{x}}_n$ is the original data whose mean is not zero.

PCA derivation for the first PC

Let $\mathbf{z}_1 \in \mathbb{R}^{N\times 1}$ and $\mathbf{w}_1 \in \mathbb{R}^{D\times 1}$ be the loading and basis vector of the first PC. The reconstruction error is

$$\mathcal{L}(\mathbf{w}_1, \mathbf{z}_1) = \frac{1}{N}\sum_{n=1}^{N} ||\mathbf{x}_n - z_{1,n}\mathbf{w}_1||^2 = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - z_{1,n}\mathbf{w}_1)^\top(\mathbf{x}_n - z_{1,n}\mathbf{w}_1)$$

$$= \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n^\top\mathbf{x}_n - 2z_{1,n}\mathbf{w}_1^\top\mathbf{x}_n + z_{1,n}^2\underbrace{\mathbf{w}_1^\top\mathbf{w}_1}_{1}) = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n^\top\mathbf{x}_n - 2z_{1,n}\mathbf{w}_1^\top\mathbf{x}_n + z_{1,n}^2)$$

$$\frac{\partial\mathcal{L}(\mathbf{z}_1)}{\partial z_{1,n}} = \frac{1}{N}(-2\mathbf{w}_1^\top\mathbf{x}_n + 2z_{1,n}) \triangleq 0 \quad \Rightarrow \quad z_{1,n} = \mathbf{w}_1^\top\mathbf{x}_n$$

$$\mathcal{L}(\mathbf{w}_1) = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n^\top\mathbf{x}_n - 2z_{1,n}\mathbf{w}_1^\top\mathbf{x}_n + z_{1,n}^2) = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n^\top\mathbf{x}_n - 2z_{1,n}^2 + z_{1,n}^2) = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n^\top\mathbf{x}_n - z_{1,n}^2)$$

$$\propto -\frac{1}{N}\sum_{n=1}^{N} z_{1,n}^2 = -\frac{1}{N}\sum_{n=1}^{N}\mathbf{w}_1^\top\mathbf{x}_n\mathbf{x}_n^\top\mathbf{w}_1 = -\mathbf{w}_1^\top\hat{\Sigma}\mathbf{w}_1 \quad \text{where} \quad \hat{\Sigma} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top$$

We see that minimizing the reconstruction error is equivalent to maximizing the variance of the latent representation w.r.t. $\mathbf{w}_1$ (since $\mathrm{Var}[z] = \mathbb{E}[z^2] - \mathbb{E}^2[z] = \mathbb{E}[z^2]$, where $\mathbb{E}[z] = \mathbf{w}_1^\top\mathbb{E}[\mathbf{x}]_n = 0$).

The first PC is the eigenvector of $\hat{\Sigma}$ with the largest eigenvalue

Because we want the projection to be orthonormal such that $\mathbf{w}_1^\top \mathbf{w} = 1$, we introduce the following constraint with the *Lagrange multiplier* λ_1 to the loss function:

$$\tilde{\mathcal{L}}(\mathbf{w}_1) = -\mathbf{w}_1^\top \hat{\Sigma} \mathbf{w}_1 + \lambda_1 (\mathbf{w}_1^\top \mathbf{w}_1 - 1) \tag{1}$$

$$\frac{\partial \tilde{\mathcal{L}}(\mathbf{w}_1)}{\partial \mathbf{w}_1} = -2\hat{\Sigma}\mathbf{w}_1 + 2\lambda_1 \mathbf{w}_1 \stackrel{\Delta}{=} 0 \quad \Rightarrow \quad \hat{\Sigma}\mathbf{w}_1 = \lambda_1 \mathbf{w}_1$$

Therefore, the optimal solution for $\mathbf{w}_1$ is an *eigenvector* of $\hat{\Sigma}$ and λ_1 corresponds to the *eigenvalue*. Multiplying $\mathbf{w}_1^\top$ on both side, we have

$$\hat{\Sigma}\mathbf{w}_1 = \lambda_1 \mathbf{w}_1 \quad \Rightarrow \quad \mathbf{w}_1^\top \hat{\Sigma} \mathbf{w}_1 = \lambda_1 \mathbf{w}_1^\top \mathbf{w}_1 \quad \Rightarrow \quad \mathbf{w}_1^\top \hat{\Sigma} \mathbf{w}_1 = \lambda_1$$

Since we want to maximize $\mathbf{w}_1^\top \hat{\Sigma} \mathbf{w}_1$ (i.e., minimizing the loss in Eq (1)), we pick the eigenvector that corresponds to the *largest* eigenvalue:

$$\mathbf{w}_1^* \leftarrow \arg\max_{\mathbf{w}_1} \mathbf{w}_1^\top \hat{\Sigma} \mathbf{w}_1 = \arg\max_{\mathbf{w}_1} \mathrm{Var}[\mathbf{z}_1] = \arg\min_{\mathbf{w}_1} \mathcal{L}(\mathbf{w}_1)$$

Computing the second PC

We can find the second PC to further minimize the reconstruction error:

$$\mathcal{L}(\mathbf{w}_2) = \frac{1}{N}\sum_{n=1}^{N} ||\mathbf{x}_n - z_{1,n}\hat{\mathbf{w}}_1 - z_{2,n}\mathbf{w}_2||^2 = \frac{1}{N}\sum_{n=1}^{N} ||\tilde{\mathbf{x}}_n - z_{2,n}\mathbf{w}_2||^2$$

$$= \frac{1}{N}\sum_{n=1}^{N} \tilde{\mathbf{x}}_n^\top \tilde{\mathbf{x}}_n - \mathbf{w}_2^\top \hat{\tilde{\Sigma}} \mathbf{w}_2 \quad \text{where}$$

$$\hat{\tilde{\Sigma}} = \frac{1}{N}\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - z_{1,n}\hat{\mathbf{w}}_1)(\mathbf{x}_n - z_{1,n}\hat{\mathbf{w}}_1)^\top$$

$$= \underbrace{\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^\top}_{\hat{\Sigma}} - \hat{\mathbf{w}}_1\frac{1}{N}\sum_{n=1}^{N} z_{1,n}\mathbf{x}_n^\top - \left(\frac{1}{N}\sum_{n=1}^{N} z_{1,n}\mathbf{x}_n\right)\hat{\mathbf{w}}_1^\top + \left(\frac{1}{N}\sum_{n=1}^{N} z_{1,n}^2\right)\hat{\mathbf{w}}_1\hat{\mathbf{w}}_1^\top$$

Therefore

$$\mathbf{w}_2^\top \hat{\tilde{\Sigma}} \mathbf{w}_2 = \mathbf{w}_2^\top \hat{\Sigma}\mathbf{w}_2 - \underbrace{\mathbf{w}_2^\top\hat{\mathbf{w}}_1}_{0}\frac{1}{N}\sum_{n=1}^{N} z_{1,n}\mathbf{x}_n^\top\mathbf{w}_2 - \mathbf{w}_2^\top\left(\frac{1}{N}\sum_{n=1}^{N} z_{1,n}\mathbf{x}_n\right)\underbrace{\hat{\mathbf{w}}_1^\top\mathbf{w}_2}_{0} + \left(\frac{1}{N}\sum_{n=1}^{N} z_{1,n}^2\right)\underbrace{\mathbf{w}_2^\top\hat{\mathbf{w}}_1}_{0}\underbrace{\hat{\mathbf{w}}_1^\top\mathbf{w}_2}_{0}$$

$$= \mathbf{w}_2^\top \hat{\Sigma}\mathbf{w}_2$$

Computing the second PC

We can find the second PC to further minimize the reconstruction error:

$$\mathcal{L}(\mathbf{w}_2) = \frac{1}{N}\sum_{n=1}^{N} ||\mathbf{x}_n - z_{1,n}\hat{\mathbf{w}}_1 - z_{2,n}\mathbf{w}_2||^2 = \frac{1}{N}\sum_{n=1}^{N} ||\tilde{\mathbf{x}}_n - z_{2,n}\mathbf{w}_2||^2$$

$$= \frac{1}{N}\sum_{n=1}^{N} \tilde{\mathbf{x}}_n^\top \tilde{\mathbf{x}}_n - \mathbf{w}_2^\top \hat{\tilde{\Sigma}} \mathbf{w}_2 = \frac{1}{N}\sum_{n=1}^{N} \tilde{\mathbf{x}}_n^\top \tilde{\mathbf{x}}_n - \mathbf{w}_2^\top \hat{\Sigma} \mathbf{w}_2$$

Adding the orthogonal constraints $\mathbf{w}_1^\top \mathbf{w}_2 = 0$ and orthonormal constraint $\mathbf{w}_2^\top \mathbf{w}_2 = 1$:

$$\tilde{\mathcal{L}}(\mathbf{w}_2) = -\mathbf{w}_2^\top \hat{\Sigma} \mathbf{w}_2 + \lambda_2(\mathbf{w}_2^\top \mathbf{w}_2 - 1) + \lambda_{12}\mathbf{w}_2^\top \mathbf{w}_1$$

$$\frac{\partial \tilde{\mathcal{L}}(\mathbf{w}_2)}{\partial \mathbf{w}_2} = -2\hat{\Sigma}\mathbf{w}_2 + 2\lambda_2 \mathbf{w}_2 + \lambda_{12}\mathbf{w}_1 \stackrel{\Delta}{=} 0$$

Solving for $\mathbf{w}_2$:

$$2\hat{\Sigma}\mathbf{w}_2 = 2\lambda_2\mathbf{w}_2 + \lambda_{12}\mathbf{w}_1 \quad \overset{\mathbf{w}_1^\top \cdot}{\Rightarrow} \quad 2\overbrace{\mathbf{w}_1^\top \hat{\Sigma}\mathbf{w}_2}^{\lambda_1 \mathbf{w}_1^\top} = 2\lambda_2 \overbrace{\mathbf{w}_1^\top \mathbf{w}_2}^{0} + \lambda_{12}\overbrace{\mathbf{w}_1^\top \mathbf{w}_1}^{1} \Rightarrow 0 = \lambda_{12}$$

$$\Rightarrow \hat{\Sigma}\mathbf{w}_2 = \lambda_2 \mathbf{w}_2 \quad \Rightarrow \quad \mathbf{w}_2^\top \hat{\Sigma}\mathbf{w}_2 = \lambda_2$$

Therefore, the solution for $\mathbf{w}_2$ for the second PC is the second largest eigenvector.

Generalizing to computing all K PCs

Find the k^{th} PC can be done in the same way

$$\mathcal{L}(\mathbf{w}_k) = \frac{1}{N}\sum_{n=1}^{N} ||\mathbf{x}_n - \sum_{s=1}^{k-1} z_{s,n}\mathbf{w}_s - z_{k,n}\mathbf{w}_k||^2 = \frac{1}{N}\sum_{n=1}^{N} ||\tilde{\mathbf{x}}_n - z_{k,n}\mathbf{w}_k||^2$$

Iteratively computing the k^{th} eigenvector is slow.
For $N << D$ (e.g., $N = 100$ samples versus $D = 20,000$ genes), we can efficiently compute all eigenvectors by solving the *eigendecomposition* of the square and symmetric covariance matrix:

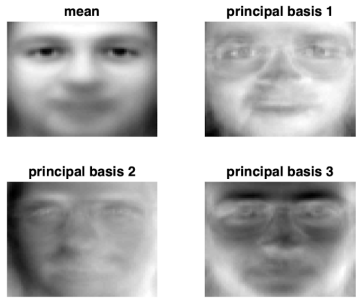
$$\underbrace{\mathbf{X}^\top\mathbf{X}}_{N\times N} = \mathbf{U}\Lambda\mathbf{U}^{-1}$$

where $\mathbf{U}$ contains all of the N eigenvectors and Λ is the diagonal matrix with the diagonal elements being the eigenvalues. Because $\mathbf{U}$ is a orthogonal matrix, $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$ and $\mathbf{U}^\top = \mathbf{U}^{-1}$.
More efficiently, we can compute the truncated Singular Vector Decomposition (SVD) to get only the first $K < \min(N, D)$ PCs by $\mathbf{X} = \mathbf{USV}^\top$ (details omitted).

Eigen faces (Murphy22 Chapter 20.1)

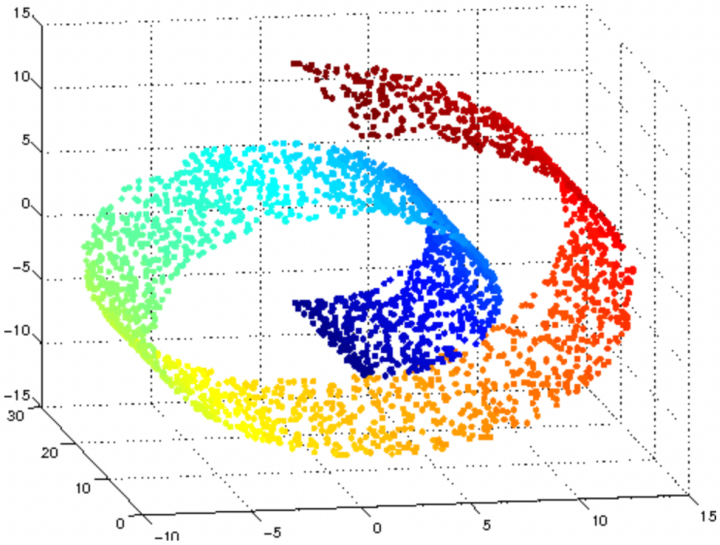
PCA were performed on 64×64 pxiel images from the Olivetti face database (panel a). Mean and the first 3 PCA components $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3$ are displayed in panel b.

(a)



(b)

Data Manifold

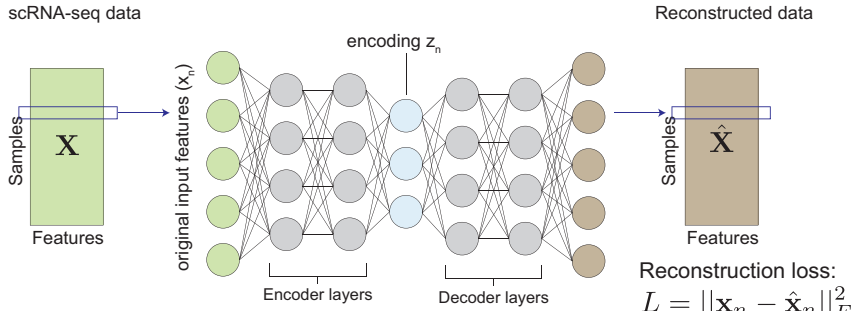


Outline

Autoencoders

AE main idea: same goal as PCA to describe data with as few parameters as possible, but allow for complex non-linear relationship between those parameters and the input data.

MLP autoencoder



Encoding by a 3-layer feedforward network (i.e., encoder):

$$z_n = f(f(\mathbf{x}_n \mathbf{W}_E^{(0)}) \mathbf{W}_E^{(1)}) \mathbf{W}_E^{(2)}$$

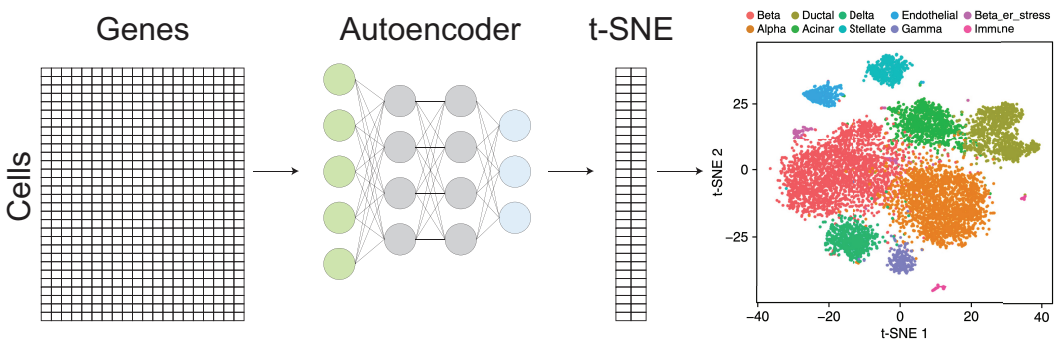
Decoding by another 3-layer feedforward network (i.e., decoder):

$$\hat{\mathbf{x}}_d = f(f(\mathbf{z}_n \mathbf{W}_D^{(0)}) \mathbf{W}_D^{(1)}) \mathbf{W}_D^{(2)}$$

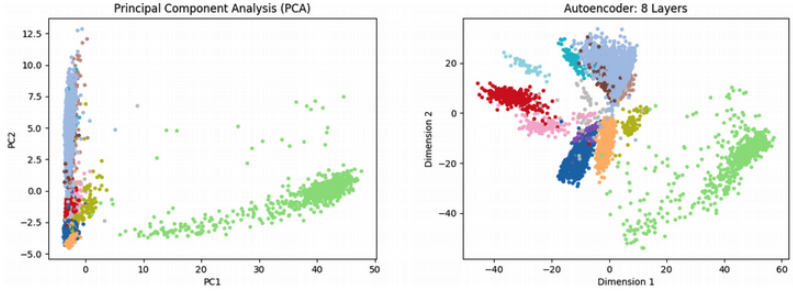
Loss: $L = \frac{1}{N} \sum_n ||\mathbf{x} - \hat{\mathbf{x}}_n||^2$; Backpropagation: $\mathbf{W}_\cdot^{(\ell)} \leftarrow \mathbf{W}_\cdot^{(\ell)} - \epsilon \nabla L(\mathbf{W}_\cdot^{(\ell)})$

Discovering cell types from single-cell gene expression data

- When dealing with high-dimensional data, it is useful to reduce the dimensionality to a lower dimensional subspace to capture the "essence" of the data.
- Below is an example of applying **Autoencoder** followed by **t-distributed stochastic neighbour embedding** (t-SNE) to thousands of cells, each having the expression of 20,000 genes



PCA vs Autoencoder



https://towardsdatascience.com/deep-learning-for-single-cell-biology-935d45064438

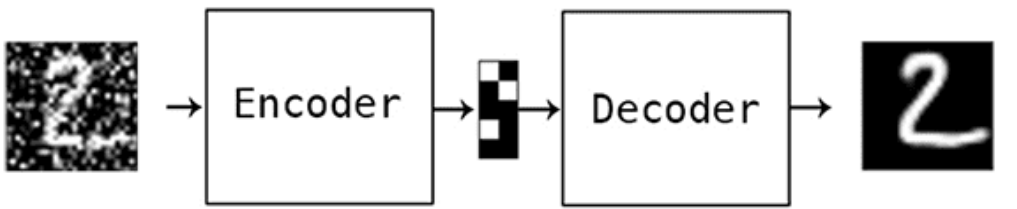
Denoising autoencoder (DAE) (?)

DAE takes the original input with added Gaussian or Bernoulli noise (for binary image):

$$\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \sigma^2 \mathbf{I}), \quad \text{or} \quad \tilde{\mathbf{x}} \sim \mathbf{x}\mathcal{B}(p, 1-p)$$

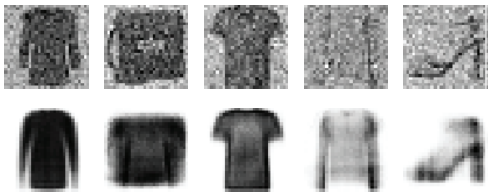
DAE are an extension of simple autoencoders to help:

- The hidden layers of the autoencoder learn more robust filters
- Reduce the risk of overfitting in the autoencoder
- Prevent the autoencoder from learning a simple identify function



Reconstructed Fashion MNIST images from validation set by DAE

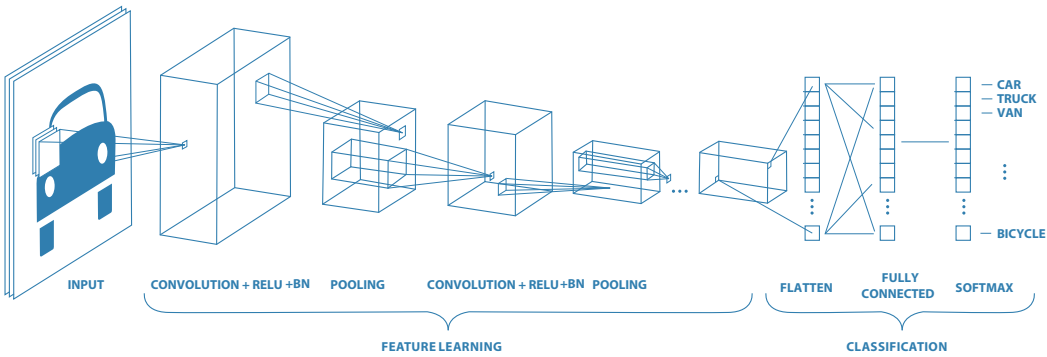
with Gaussian noise



with Bernoulli dropout noise

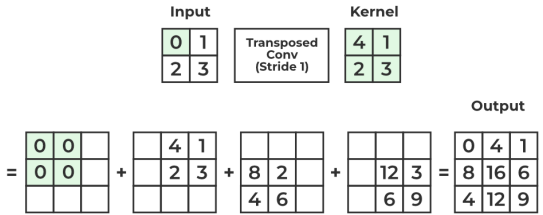
(Murphy22 Chapter 20.3 Figure 20.19)

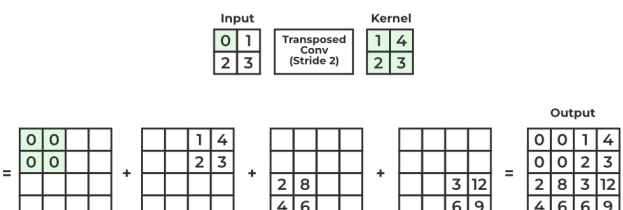
Convolutional neural network for classification on images (Lecture 3)



We can use CNN architectures in our autoencoders!

Transposed convolutional aka Deconvolution layer

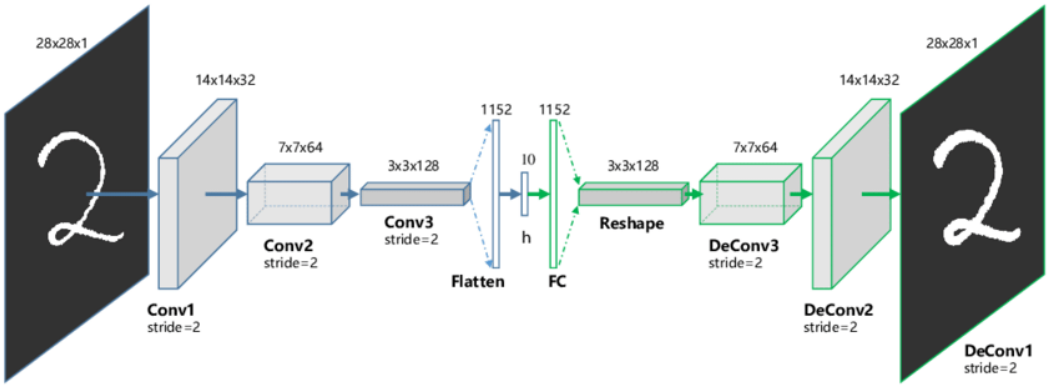




- Instead of sliding the kernel over the input pixels and performing element-wise multiplication and summation, a transposed convolutional layer **slides the input pixel over the kernel** and performs element-wise multiplication and summation.
- This results in an output that is larger than the input, and the size of the output can be controlled by the stride and padding parameters of the layer.

(source)

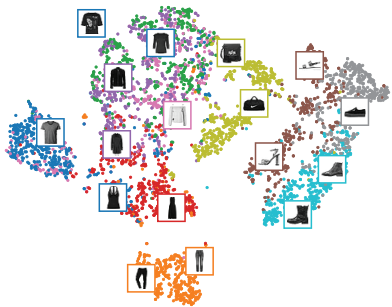
CNN autoencoder



source

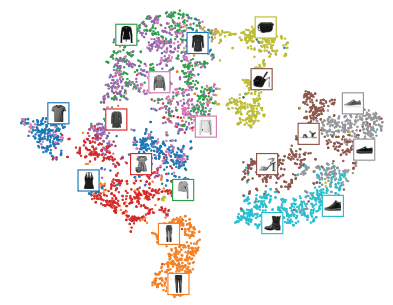
MLP AE vs CNN AE on Fashion MNIST (Murphy22 Fig 20.17 & 18)

MLP AE (784-100-30)



CNN AE

3x3 Conv (16), MaxPool (2x2), Conv (32, 3x3), MaxPool (2x2), Conv (64, 3x3), MaxPool (2x2)



Summary of Autoencoder

- Using network encoder and network decoder, we can train non-linear function that can project high-dimensional data onto low-dimensional latent space by minimizing the reconstruction loss via stochastic gradient decent.
- AE can have MLP or CNN architectures. When applied to images, CNN architecture works better because it benefits from the same induction bias as in the CNN classifiers.
- Denoising autoencoder learns more robust representation of the data than the vanilla autoencoder
- VAE can generate new samples by inferring the distribution of the latent embedding.

Summary of Unsupervised Learning

Goal: Find the regularities in the Data. That often means finding a compressed way to represent the data.

- Clustering: represent all data as a few different clusters.
- PCA & AE: represent the data as a low dimensional manifold of the main directions of variation.
  - PCA: linear submanifold, can find optimal sub-manifold analytically.
  - AE: non-linear, optimality not guaranteed.

Congratulations!