Foundations of Artificial Intelligence: Homework 4

b11902038 鄭博允

May 19, 2025

Problem 1

First we use the product rule:

$$\varphi'(s) = \theta(s) + s \cdot \theta'(s)$$

Since

$$\theta'(s) = \theta(s)(1 - \theta(s))$$

therefore we have

$$\varphi'(s) = \theta(s) + s \cdot \theta(s)(1 - \theta(s)) = \theta(s)\,[1 + s(1 - \theta(s))]$$

Problem 2

(a)

Given the transition matrix:

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0.5 \\ 0 & 0 & 0.5 \\ 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{v}_0 = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$$

We compute $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4$ and $\mathbf{v}_5$ with the equation $\mathbf{v}_t = \mathbf{P}\mathbf{v}_{t-1}$:

$$\mathbf{v}_1 = \mathbf{P}\mathbf{v}_0 = \begin{bmatrix} 1 \cdot \frac{1}{3} + 0.5 \cdot \frac{1}{3} \\ 0.5 \cdot \frac{1}{3} \\ 1 \cdot \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} + \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{6} \\ \frac{1}{3} \end{bmatrix}$$

$$\mathbf{v}_2 = \mathbf{P}\mathbf{v}_1 = \begin{bmatrix} 1 \cdot \frac{1}{6} + 0.5 \cdot \frac{1}{3} \\ 0.5 \cdot \frac{1}{3} \\ 1 \cdot \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{6} + \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{6} \\ \frac{1}{2} \end{bmatrix}$$

$$\mathbf{v}_3 = \mathbf{P}\mathbf{v}_2 = \begin{bmatrix} 1 \cdot \frac{1}{6} + 0.5 \cdot \frac{1}{2} \\ 0.5 \cdot \frac{1}{2} \\ 1 \cdot \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{1}{6} + \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{5}{12} \\ \frac{1}{4} \\ \frac{1}{3} \end{bmatrix}$$

$$\mathbf{v}_4 = \mathbf{P}\mathbf{v}_3 = \begin{bmatrix} 1 \cdot \frac{1}{4} + 0.5 \cdot \frac{1}{3} \\ 0.5 \cdot \frac{1}{3} \\ 1 \cdot \frac{5}{12} \end{bmatrix} = \begin{bmatrix} \frac{1}{4} + \frac{1}{6} \\ \frac{1}{6} \\ \frac{5}{12} \end{bmatrix} = \begin{bmatrix} \frac{5}{12} \\ \frac{1}{6} \\ \frac{5}{12} \end{bmatrix}$$

$$\mathbf{v}_5 = \mathbf{P}\mathbf{v}_4 = \begin{bmatrix} 1 \cdot \frac{1}{6} + 0.5 \cdot \frac{5}{12} \\ 0.5 \cdot \frac{5}{12} \\ 1 \cdot \frac{5}{12} \end{bmatrix} = \begin{bmatrix} \frac{1}{6} + \frac{5}{24} \\ \frac{5}{24} \\ \frac{5}{12} \end{bmatrix} = \begin{bmatrix} \frac{7}{12} \\ \frac{5}{24} \\ \frac{5}{12} \end{bmatrix}$$

(b)

Let:

$$\mathbf{v}^* = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Find $\mathbf{v}^*$ such that $\mathbf{v}^* = \mathbf{P}\mathbf{v}^*$ and $\sum \mathbf{v}^* = 1$.
From $\mathbf{v}^* = \mathbf{P}\mathbf{v}^*$:

$$\begin{aligned} a &= b + \frac{1}{2}c \\ b &= \frac{1}{2}c \\ c &= a \end{aligned}$$

Since $a + b + c = 1$, we can derive:

$$\mathbf{v}^* = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \frac{2}{5} \\ \frac{1}{5} \\ \frac{2}{5} \end{bmatrix}$$

Problem 3

Given: $X = \{(1,2),(3,4),(7,0),(10,2)\}$, $K = 2$

(a)

initial centroids: $\{\mu_1, \mu_2\} = \{(1,2),(3,4)\}$.

Iteration 1:

1. Assign points based on Euclidean distance
   - cluster 1: $(1,2)$
   - cluster 2: $(3,4),(7,0),(10,2)$
2. Update centroids
   - $\mu_1 = (1,2)$
   - $\mu_2 = (\frac{20}{3}, 2)$

Iteration 2:

1. Assign points based on Euclidean distance
   - cluster 1: $(1,2),(3,4)$
   - cluster 2: $(7,0),(10,2)$
2. Update centroids
   - $\mu_1 = (2,3)$
   - $\mu_2 = \frac{17}{2}, 1)$

Iteration 3:

1. Assign points based on Euclidean distance
   - cluster 1: $(1,2),(3,4)$
   - cluster 2: $(7,0),(10,2)$
2. Same assignment $\rightarrow$ converge

(b)

initial centroids: $\{\mu_1, \mu_2\} = \{(1,2),(7,0)\}$.

Iteration 1:

1. Assign points based on Euclidean distance
   - cluster 1: $(1,2),(3,4)$
   - cluster 2: $(7,0),(10,2)$
2. Update centroids
   - $\mu_1 = (2,3)$
   - $\mu_2 = (\frac{17}{2},1)$

Iteration 2:

1. Assign points based on Euclidean distance
   - cluster 1: $(1,2),(3,4)$
   - cluster 2: $(7,0),(10,2)$
2. Same assignment $\rightarrow$ converge

The result is as same as (a).

(C)

New set: $\{(1,2),(3,4),(7,0),(10,2),(5,6)\}$

If we set initial centroids to $\{\mu_1, \mu_2\} = \{(1,2),(3,4)\}$.

Iteration 1:

1. Assign points based on Euclidean distance
   - cluster 1: $(1,2)$
   - cluster 2: $(3,4),(5,6),(7,0),(10,2)$
2. Update centroids

- $\mu_1 = (1, 2)$
- $\mu_2 = (\frac{25}{4}, 3)$

Iteration 2:

1. Assign points based on Euclidean distance
   - cluster 1: $(1,2), (3,4)$
   - cluster 2: $(5,6), (7,0), (10,2)$
2. Update centroids
   - $\mu_1 = (2, 3)$
   - $\mu_2 = (\frac{22}{3}, \frac{8}{3})$

Iteration 3:

1. Assign points based on Euclidean distance
   - cluster 1: $(1,2), (3,4)$
   - cluster 2: $(5,6), (7,0), (10,2)$
2. Same assignment $\rightarrow$ converge

Thus K-means can converge to local minima depending on initialization.

Foundations of Artificial Intelligence: Homework 4 Programming Part Report

b11902038 鄭博允

May 19, 2025

(a) Mean Vector and Eigenfaces

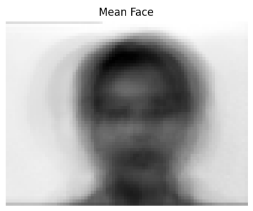


Figure 1: Mean face

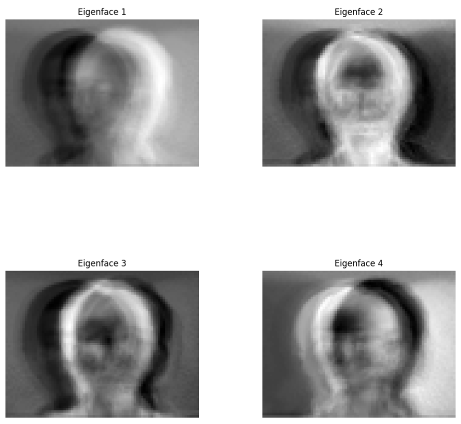
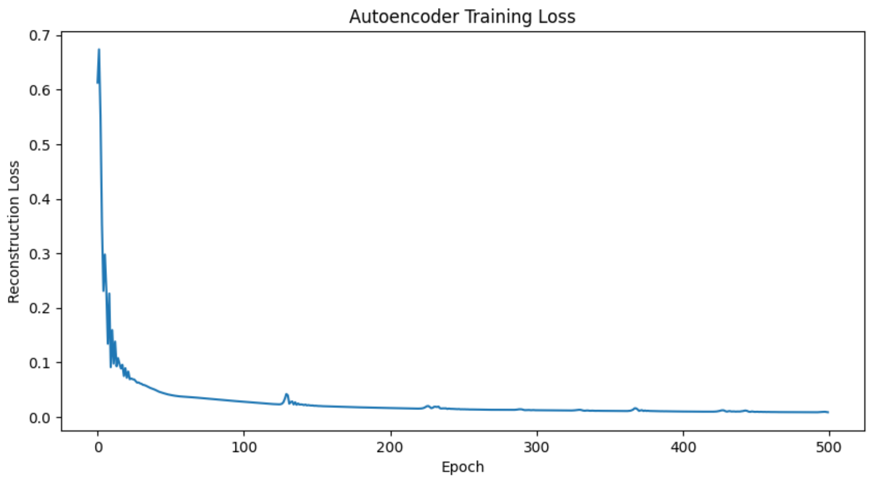


Figure 2: top 4 eigenfaces

(b) Training Curves



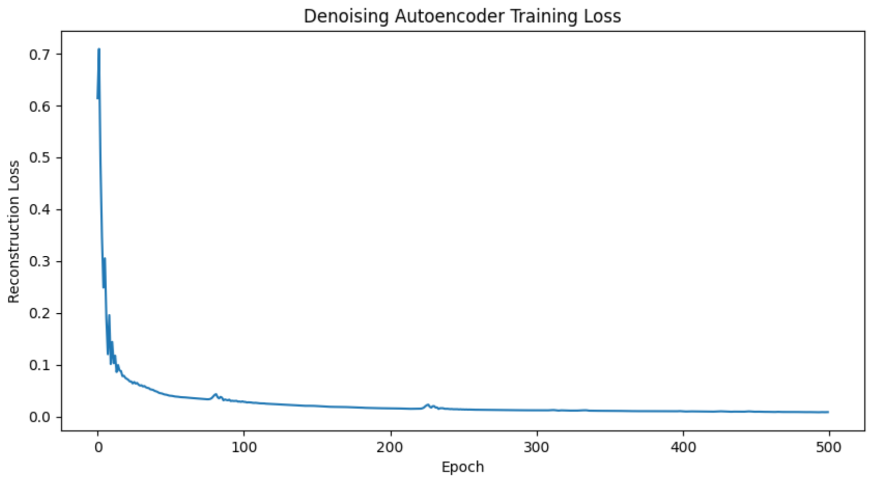


Figure 3: Training curve of *Autoencoder* and *Denoising Autoencoder*

(c) Image Reconstruction and Mean Squared Error

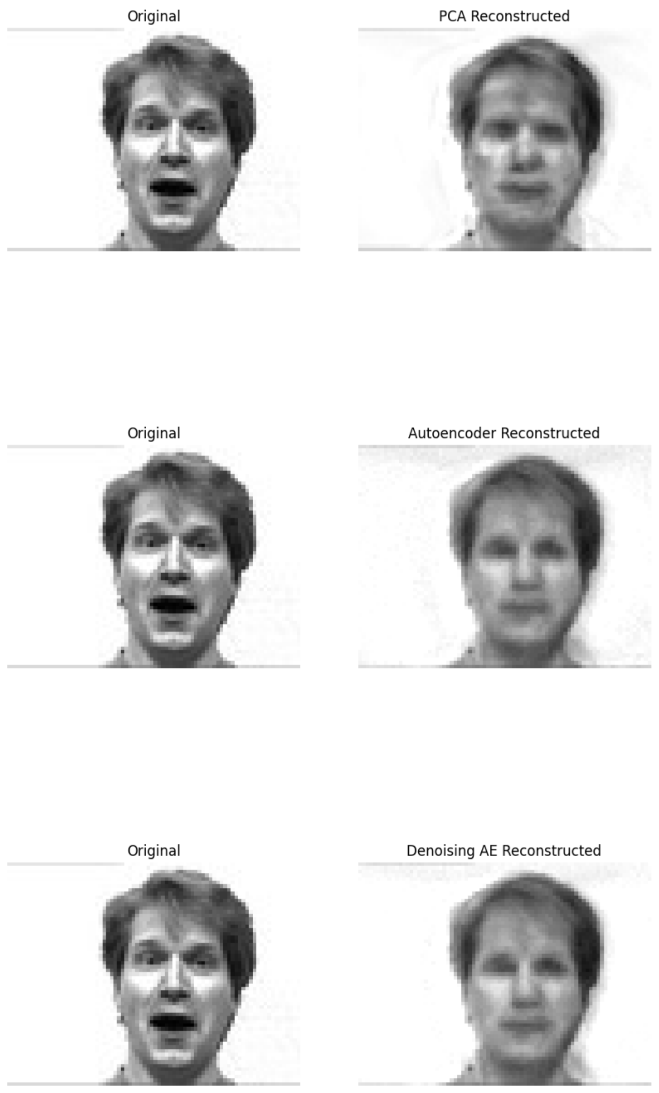


Figure 4: Original and reconstruction image for different models

The mean squared error of each model:

- *PCA*: 0.010710
- *Autoencoder*: 0.013231
- *Denoising Autoencoder*: 0.013431

(d) Denoising Autoencoder Architecture Modification

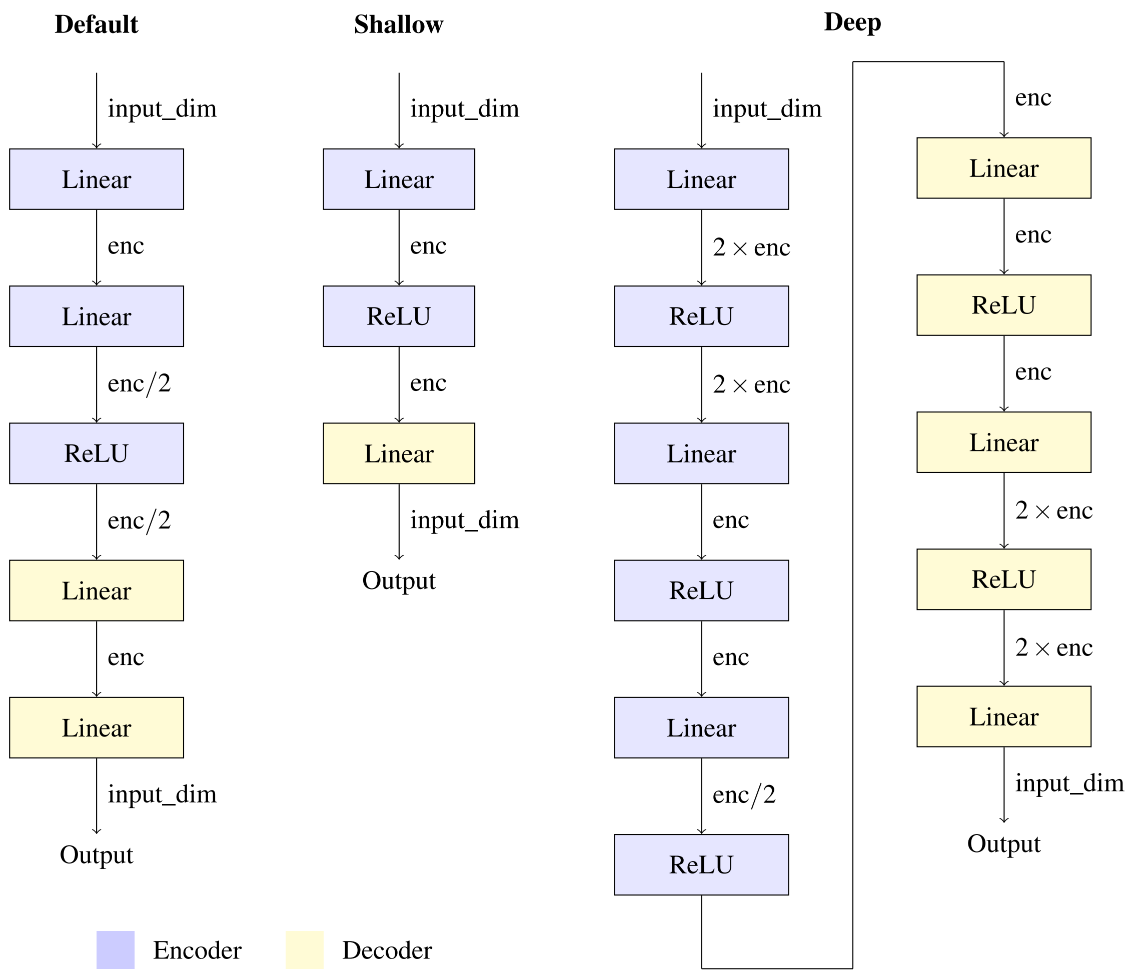


Figure 5: Diagram of different architectures

Three different architectures were tested: *Default*, *Shallow*, and *Deep*. The architecture of each is shown as Figure 5. All three architectures were tested twice under 500 epochs, as shown in table 1. The training curve for different Denoising Autoencoder architectures is shown in Figure 6.

Deep achieves the lowest error, indicating better learning and generalization capacity due to higher representation power. While *Shallow* underperforms, likely

Architecture	**Trial 1 Loss**	**Trial 2 Loss**	**Avg. Loss**
Default	0.0157	0.0131	0.0144
Shallow	0.0197	0.0184	0.0190
Deep	0.0138	0.0142	0.0140

Table 1: Reconstruction loss of different architectures

due to its limited capacity. This indicates that a deeper network can potentially learn more complex representations and achieve better reconstruction performance. However, if we consider the training time, the *Default* might be better choice to achieve favorable result with less time.

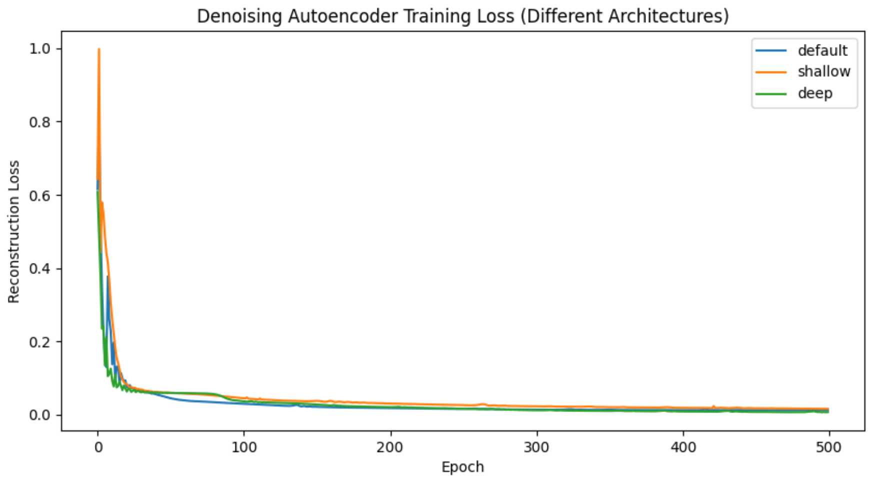


Figure 6: Training curve for different Denoising Autoencoder architectures

(e) Optimizer Comparison for Denoising Autoencoder

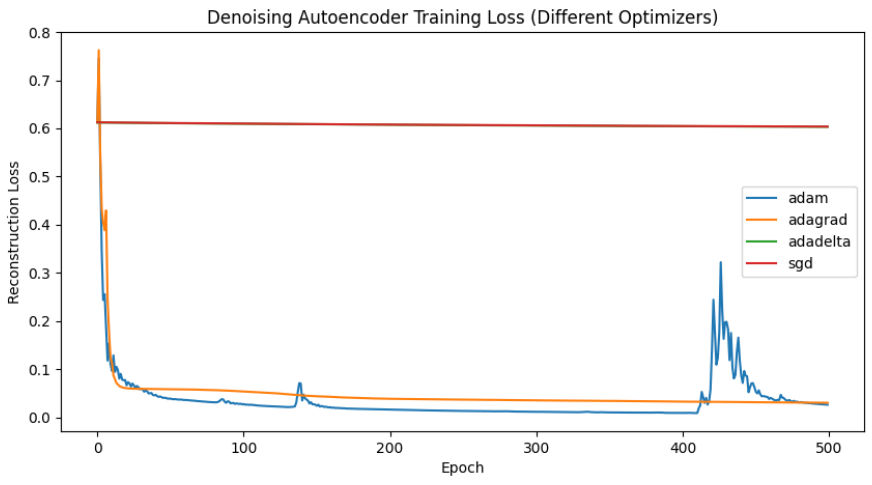


Figure 7: Training curve for different Denoising Autoencoder optimizers

Four different optimizers were tested: *Adam*, *Adadrad*, *Adadelta* and *SGD* (vanilla). All optimizers have a learning rate = 1. The training curve for different optimizers is shown in Figure 6.

In terms of convergence speed, *Adam* converges fastest while *SGD* and *Adadelta* seem to have failed at learning as their loss nearly shows no decrease through epochs. The **Yale Face dataset** is composed of grayscale facial images with limited variation. The low variance in pixel intensity and the subtle features in facial images demand an optimizer capable of handling small gradient updates effectively—something *Adam* excels at but *Adadelta* and *SGD* do not. *Adadelta* might be over-adapting to small gradients, and *SGD* might have no momentum since the learning rate is unsuitable.