Homework 4 Written Part Report
FAI 2024

B11705048, 林杰

The reference of this homework is only ChatGPT. Besides ChatGPT's help, I completed the homework all by myself.

Problem 1 (15 points)

`Swish` is an activation function that can be viewed as a "modification" of the logistic function. Swish has been shown to be better than ReLU in some deep learning models. Recall that the logistic function is defined as

$$\theta(s) = \frac{1}{1+\exp(-s)}$$

Now Swish is defined as

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

What is $\varphi'(s)$?

1.

$$\theta(s) = \frac{1}{1+e^{-s}}, \quad \varphi(s) = s\cdot\theta(s) = \frac{s}{1+e^{-s}}$$

$$\frac{d}{ds}\left[\frac{s}{1+e^{-s}}\right] = \frac{\frac{d}{ds}[s]\cdot(1+e^{-s}) - s\cdot\frac{d}{ds}[1+e^{-s}]}{(1+e^{-s})^2}$$

$$= \frac{(1+e^{-s}) - s(-e^{-s})}{(1+e^{-s})^2} = \frac{1+e^{-s}+se^{-s}}{(1+e^{-s})^2} \quad \#$$

Problem 2 (20 points)

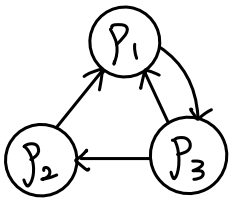
In the class, we briefly talked about the famous **PageRank** algorithm, which assigns a rank to each web page. Given a set of pages: $P = \{P_1, P_2, \cdots, P_n\}$ and their incoming links: $in(P_i) = \{P_j \mid P_j \text{ has a link to } P_i, i \neq j\}$. The basic idea is that a web page should be ranked higher if it has 1) more incoming hyperlinks and 2) incoming links from high-ranked pages. The PageRank can be computed through the following iterative algorithm:

1. Initialize $\text{PageRank}_0(P_i) = \frac{1}{n}$ for each page.
2. Compute the updated PageRank for each page:

$$\text{PageRank}_1(P_i) = \sum_{P_j \in in(P_i)} \frac{\text{PageRank}_0(P_j)}{|out(P_j)|},$$

   where $|out(P_j)|$ denotes the out-degree of P_j.
3. Repeat step 2 until the ranks converge.

Now, consider the following example with 3 pages:



$$P = \{P_1, P_2, P_3\}$$
$$in(P_1) = \{P_2, P_3\}$$
$$in(P_2) = \{P_3\}$$
$$in(P_3) = \{P_1\}$$

We define the following variables:

$$\mathbf{v}_t = \left[\text{PageRank}_t(P_1), \text{PageRank}_t(P_2), \text{PageRank}_t(P_3)\right]^T$$
$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0.5 \\ 0 & 0 & 0.5 \\ 1 & 0 & 0 \end{bmatrix},$$

where $\mathbf{v}_t$ is the ranks after the t-th iteration, and $\mathbf{P}$ is the transition matrix between pages. We can rewrite the update rule above as $\mathbf{v}_t = \mathbf{P}\mathbf{v}_{t-1}$. Answer the following questions:

(A) Based on the iterative algorithm above, please compute the values of $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$, $\mathbf{v}_4$, and $\mathbf{v}_5$.

(B) Recall that in the class, we mentioned that the algorithm converges when $\mathbf{v}^* = \mathbf{P}\mathbf{v}^*$. Solve this equation to derive $\mathbf{v}^*$. Note that you should provide a normalized $\mathbf{v}^*$, i.e., $\sum \mathbf{v}^* = 1$, as the answer.

2.

(A) $\vec{V}_0 = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$, $\vec{V}_1 = \begin{bmatrix} 0 & 1 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{6} \\ \frac{1}{3} \end{bmatrix}$, $\vec{V}_2 = \begin{bmatrix} 0 & 1 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ \frac{1}{6} \\ \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{6} \\ \frac{1}{2} \end{bmatrix}$,

$\vec{V}_3 = \begin{bmatrix} 0 & 1 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{3} \\ \frac{1}{6} \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{5}{12} \\ \frac{1}{4} \\ \frac{1}{3} \end{bmatrix}$, $\vec{V}_4 = \begin{bmatrix} 0 & 1 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{5}{12} \\ \frac{1}{4} \\ \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{5}{12} \\ \frac{1}{6} \\ \frac{5}{12} \end{bmatrix}$,

$\vec{V}_5 = \begin{bmatrix} 0 & 1 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{5}{12} \\ \frac{1}{6} \\ \frac{5}{12} \end{bmatrix} = \begin{bmatrix} \frac{3}{8} \\ \frac{5}{24} \\ \frac{5}{12} \end{bmatrix}$. #

(B) $\begin{bmatrix} 0 & 1 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{cases} y + \frac{1}{2}z = x \\ \frac{1}{2}z = y \\ x = z \end{cases} \rightarrow x : y : z = 2 : 1 : 2$

$\therefore \vec{V}^* = \begin{bmatrix} \frac{2}{5} \\ \frac{1}{5} \\ \frac{2}{5} \end{bmatrix}$ #

Problem 3 (20 points)

We talked about the K-means clustering algorithm in class, which iteratively does partition optimization and prototype optimization until convergence. Given a set of 2-dimensional data points $\mathbf{X} = \{(1,2),(3,4),(7,0),(10,2)\}$, we would like to perform K-means clustering with $K = 2$. Answer the following questions. Note that you should write down the resulting cluster centroids and partitions of each iteration.

(A) Perform the K-means algorithm with $K = 2$ and initial centroids $\{\mu_1, \mu_2\} = \{(1,2),(3,4)\}$ until convergence.

(B) Perform the K-means algorithm with $K = 2$ and initial centroids $\{\mu_1, \mu_2\} = \{(1,2),(7,0)\}$ until convergence. Are the results different from (A)?

(C) Now consider adding a data point $(5,6)$ to $\mathbf{X}$. Show that the K-means algorithm converges to a local minimum with certain intial centroids. You can demonstrate this by showing that there exists a set of initial centroids that does not converges to the global minimum.

3.

(A) Centroid 1 = (1,2), Centroid 2 = (3,4)

Itetration 1 :

For (1,2) → Centroid 1
For (3,4) → Centroid 2
For (7,0) → Centroid 2
For (10,2) → Centroid 2

⇒ Group 1 : {(1,2)}
Group 2 : {(3,4),(7,0),(10,2)}
New Centroid 1 : (1,2)
New Centroid 2 : $(\frac{20}{3}, 2)$

Iteration 2 :

For (1,2) → Centroid 1
For (3,4) → Centroid 1
For (7,0) → Centroid 2
For (10,2) → Centroid 2

⇒ Group 1 : {(1,2),(3,4)}
Group 2 : {(7,0),(10,2)}
New Centroid 1 : (2,3)
New Centroid 2 : $(\frac{17}{2}, 1)$

Iteration 3:

For (1,2) → Centroid 1
For (3,4) → Centroid 1
For (7,0) → Centroid 2
For (10,2) → Centroid 2

⇒ Group 1 : {(1,2), (3,4)}
Group 2 : {(7,0), (10,2)}
New Centroid 1 : (2,3)
New Centroid 2 : $(\frac{17}{2}, 1)$

Since the assignments do not change from the previous iteration, we stop here.

(B) ✿ Centroid 1 = (1,2), Centroid 2 = (7,0).

Itetration 1:

For (1,2) → Centroid 1
For (3,4) → Centroid 1
For (7,0) → Centroid 2
For (10,2) → Centroid 2

⇒ Group 1 : {(1,2), (3,4)}
Group 2 : {(7,0), (10,2)}
New Centroid 1 : (2,3)
New Centroid 2 : $(\frac{17}{2}, 1)$

Iteration 2:

For (1,2) → Centroid 1
For (3,4) → Centroid 1
For (7,0) → Centroid 2
For (10,2) → Centroid 2

⇒ Group 1 : {(1,2), (3,4)}
Group 2 : {(7,0), (10,2)}
New Centroid 1 : (2,3)
New Centroid 2 : $(\frac{17}{2}, 1)$

Since the assignments do not change from the previous iteration, we stop here.
The final result is same from (A), but converges faster.

(c) $X = \{(1,2), (3,4), (7,0), (10,2), (5,6)\}$
Consider initial centroids $\{\mu_1, \mu_2\} = \{(1,2), (3,4)\}$.
✿ Centroid 1 = (1,2), Centroid 2 = (3,4)

Itetration 1:

For (1,2) → Centroid 1
For (3,4) → Centroid 2
For (7,0) → Centroid 2
For (10,2) → Centroid 2
For (5,6) → Centroid 2

⇒ Group 1 : {(1,2)}
Group 2 : {(3,4), (7,0), (10,2), (5,6)}
New Centroid 1 : (1,2)
New Centroid 2 : $(\frac{25}{4}, 3)$

Iteration 2:

For (1,2) → Centroid 1
For (3,4) → Centroid 1
For (7,0) → Centroid 2
For (10,2) → Centroid 2
For (5,6) → Centroid 2

⇒ Group 1 : {(1,2), (3,4)}
Group 2 : {(7,0), (10,2), (5,6)}
New Centroid 1 : (2,3)
New Centroid 2 : $(\frac{22}{3}, \frac{8}{3})$

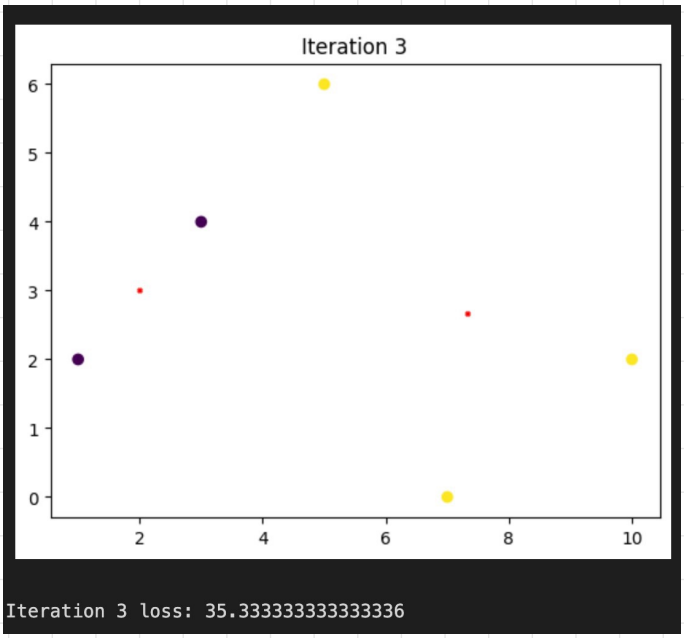
Iteration 3:

For (1,2) → Centroid 1
For (3,4) → Centroid 1
For (7,0) → Centroid 2
For (10,2) → Centroid 2
For (5,6) → Centroid 2

⇒ Group 1 : {(1,2), (3,4)}
Group 2 : {(7,0), (10,2), (5,6)}
New Centroid 1 : (2,3)
New Centroid 2 : $(\frac{22}{3}, \frac{8}{3})$

Since the assignments do not change from the previous iteration, we stop here.

Loss = Sum of distance square to the centroid.
Using the help of Python, loss ≈ 35.33.



Consider initial centroids $\{\mu_1, \mu_2\} = \{(3,4), (7,0)\}$.
✿ Centroid 1 = (3,4), Centroid 2 = (7,0)

Itetration 1 :

For (1,2) → Centroid 1
For (3,4) → Centroid 1
For (7,0) → Centroid 2
For (10,2) → Centroid 2
For (5,6) → Centroid 1

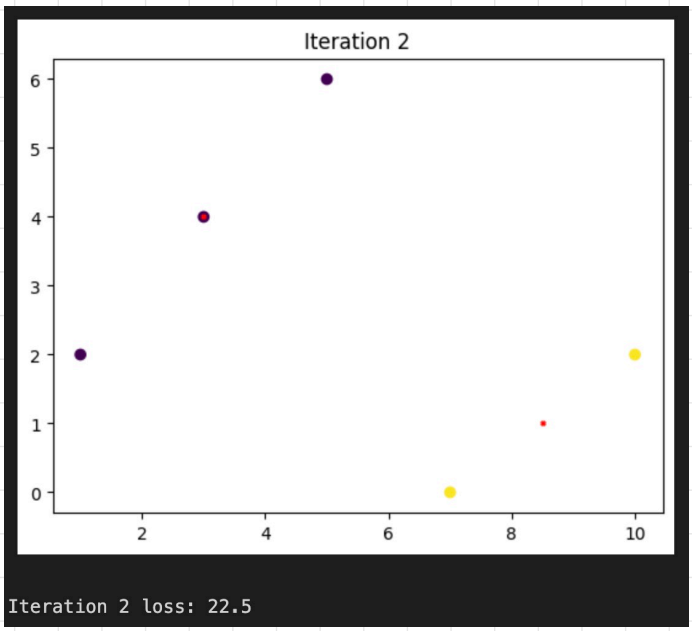
⇒ Group 1 : $\{(1,2), (3,4), (5,6)\}$
Group 2 : $\{(7,0), (10,12)\}$
New Centroid 1 : $(3,4)$
New Centroid 2 : $(\frac{17}{2}, 6)$

Itetration 2 :

For (1,2) → Centroid 1
For (3,4) → Centroid 1
For (7,0) → Centroid 2
For (10,2) → Centroid 2
For (5,6) → Centroid 1

⇒ Group 1 : $\{(1,2), (3,4), (5,6)\}$
Group 2 : $\{(7,0), (10,12)\}$
New Centroid 1 : $(3,4)$
New Centroid 2 : $(\frac{17}{2}, 6)$

Since the assignments do not change from the previous iteration, we stop here
Loss = Sum of distance square to the centroid.
Using the help of Python, loss ≈ 22.5.



Conclusion: initial centroid $\{(3,4), (7,0)\}$ can end up with lower loss than initial centroid $\{(12), (3,4)\}$. Therefore, there exists at least a pair of initial centroid that does not converges to the global minimum.
⇒ K-means converges to a local minimum with certain initial points.

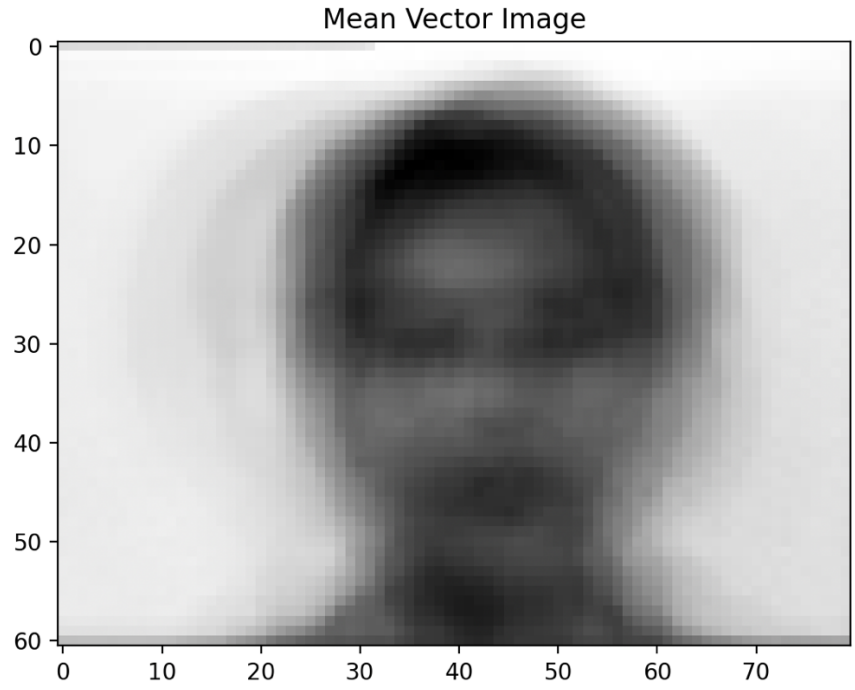
Homework 4 Programming Part Report

Foundations of Artificial Intelligence (CSIE3005)

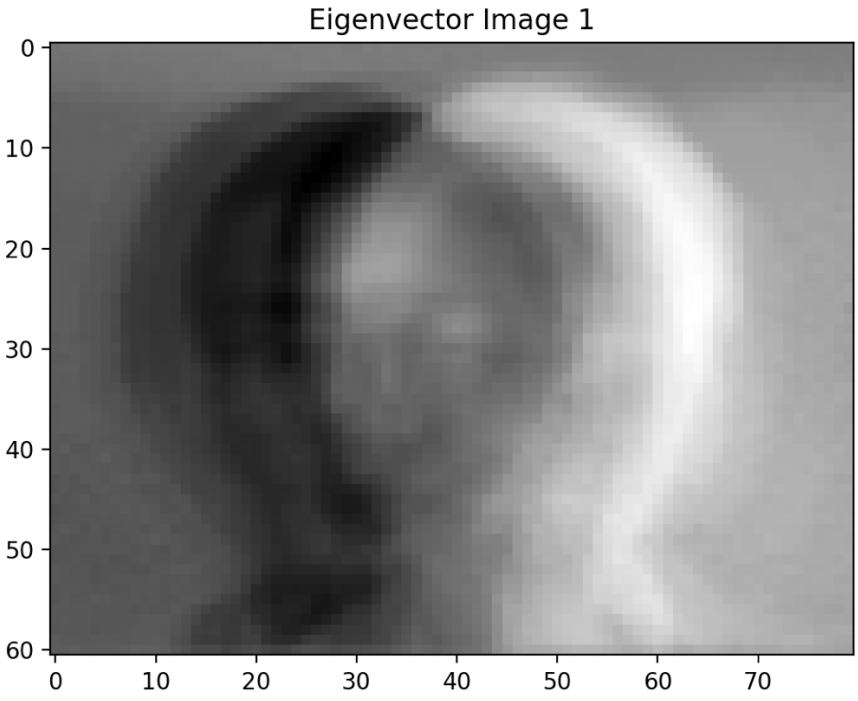
Department of Information Management, NTU - B11705048 林杰

(A) Mean Vector and Eigenvectors

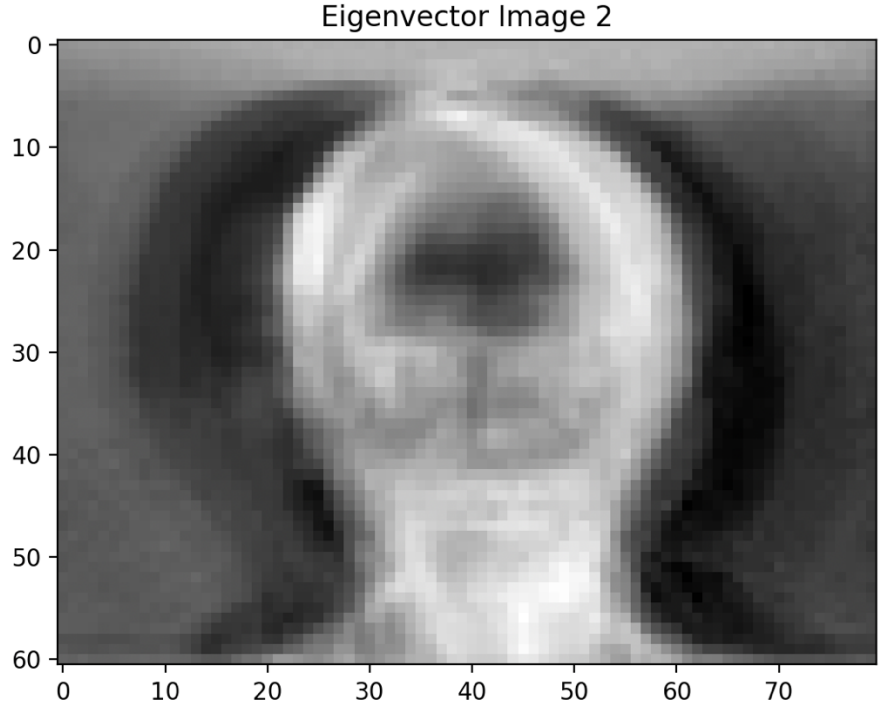
Mean Vector Image:



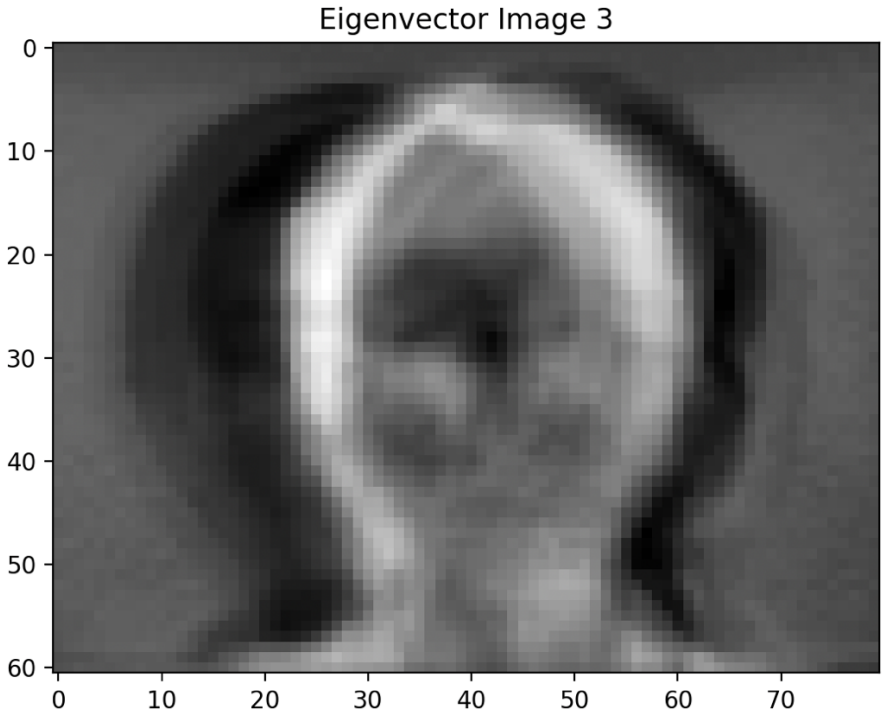
Eigenvector with 1st largest eigenvalue (as image):



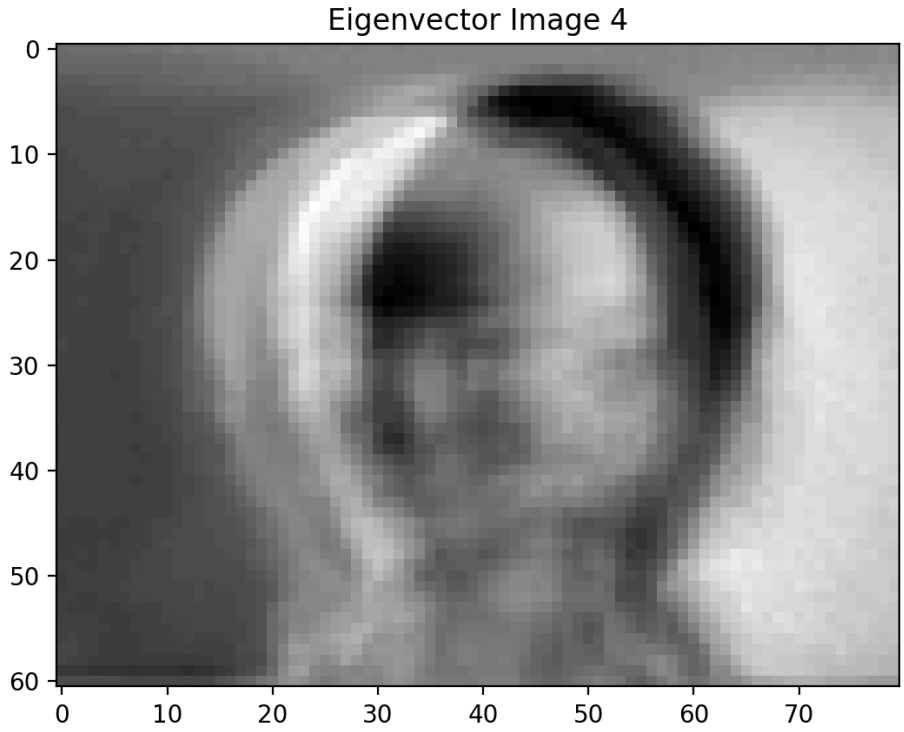
Eigenvector with 2nd largest eigenvalue (as image):



Eigenvector with 3rd largest eigenvalue (as image):

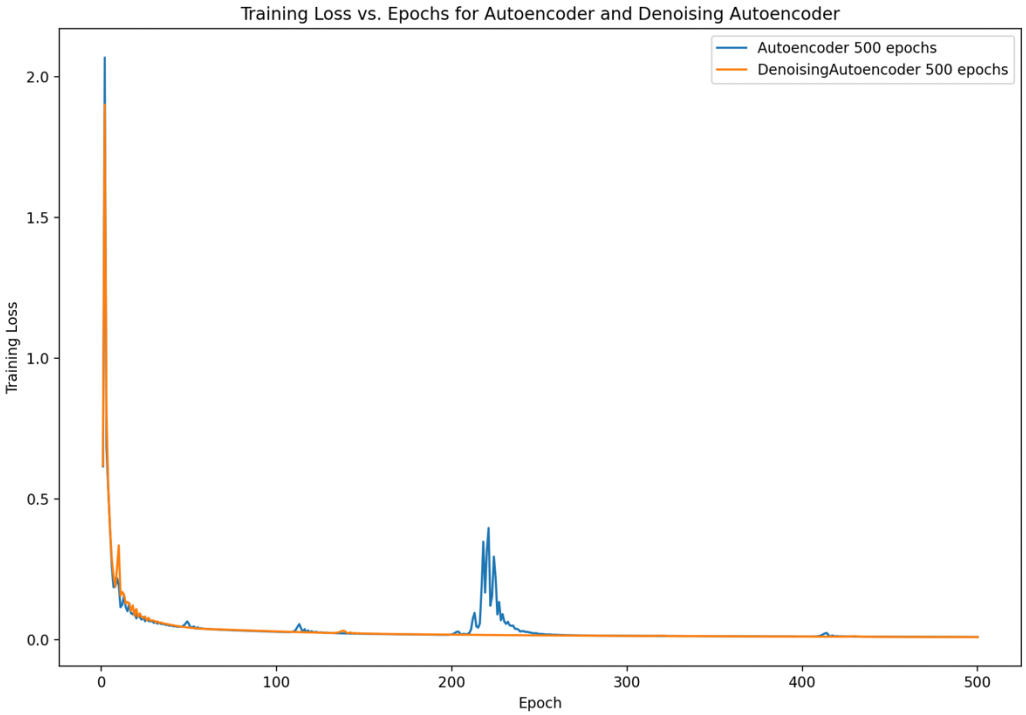


Eigenvector with 4th largest eigenvalue (as image):



(B) Training Curve for Autoencoder and Denoising Autoencoder

Below, I plot the training curve for autoencoder and denoising autoencoder (original network in the sample code). The y-axis is the training loss, and the x-axis is the number of epochs. I use Adam optimizer (lr = 0.001) in this problem.

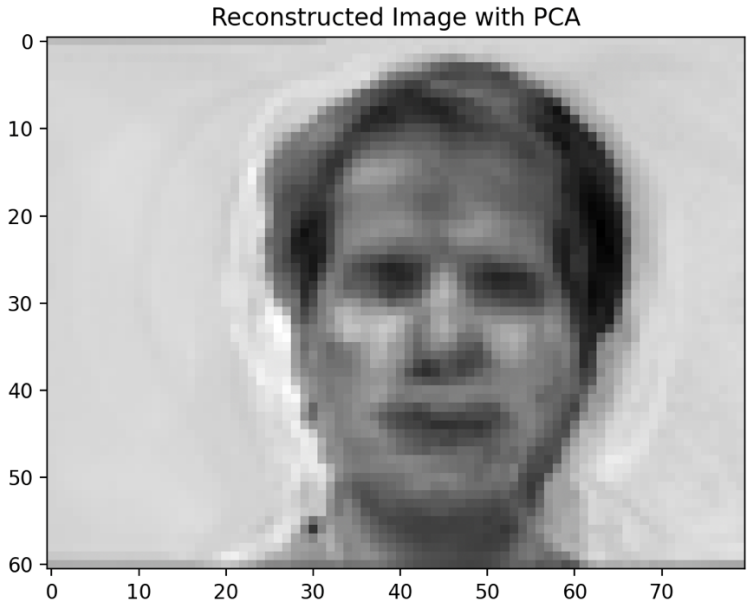


We can see from the above graph that autoencoder and denoising autoencoder can have a trend of decreasing training loss when epoch increases. However, the Autoencoder's loss has a little peak when epoch is around 215. Therefore, the conclusion is that training with denoising autoencoder is much more stable than original autoencoder.

(C) Image Comparison

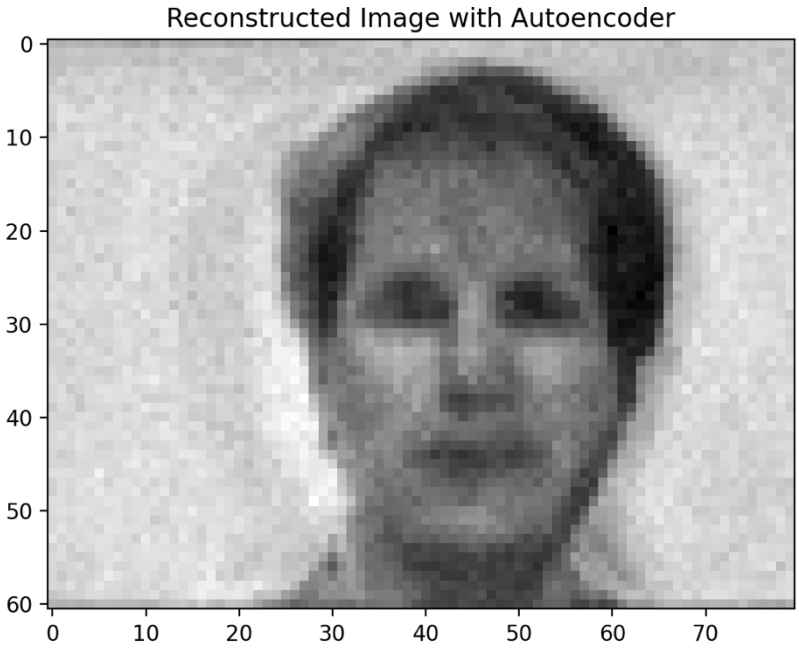
Original Image:

Reconstructed with PCA:



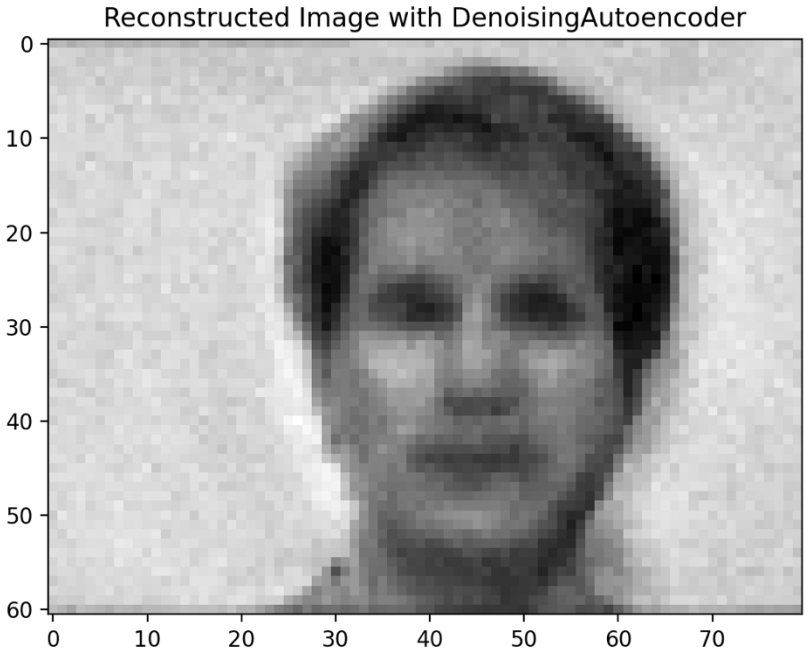
Reconstruction loss (MSE) with PCA: 0.01071046968805632.

Reconstructed with autoencoder (500 epochs, original network in the sample code):



Reconstruction loss (MSE) with autoencoder: 0.01357628019669383.

Reconstructed with denoising autoencoder (500 epochs, original network in the sample code):



Reconstruction loss with denoising autoencoder: 0.013515440799139819.

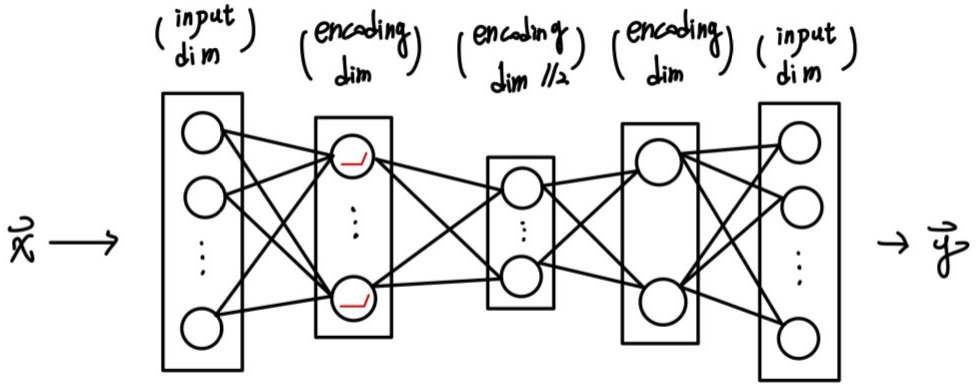
From experiment I conducted, we can find out that PCA can yield the best result

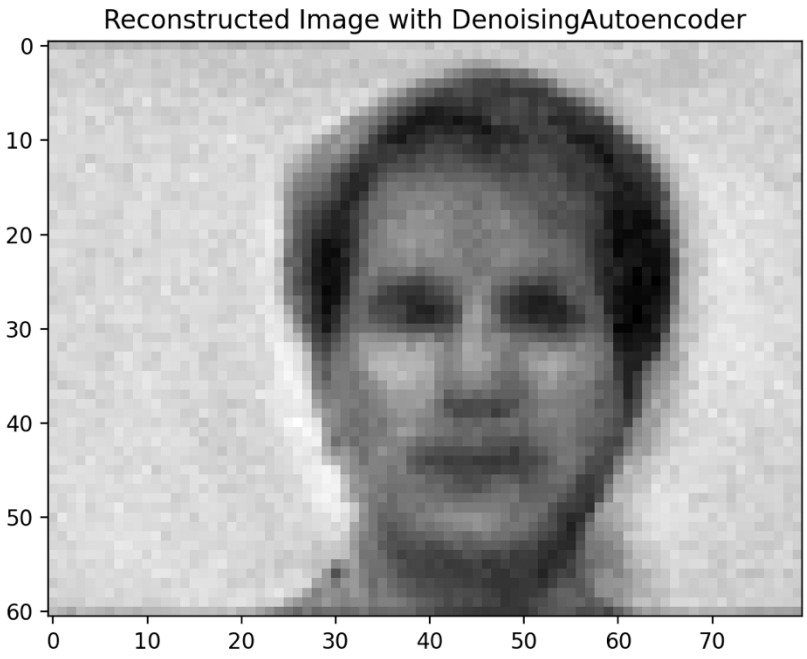
(lowest MSE). Reconstruction loss with denoising autoencoder is a little bit less than the reconstruction loss with original autoencoder.

(D) Network Architecture for the Denoising Autoencoder

Original denoising autoencoder (in the given sample code):

Structure:

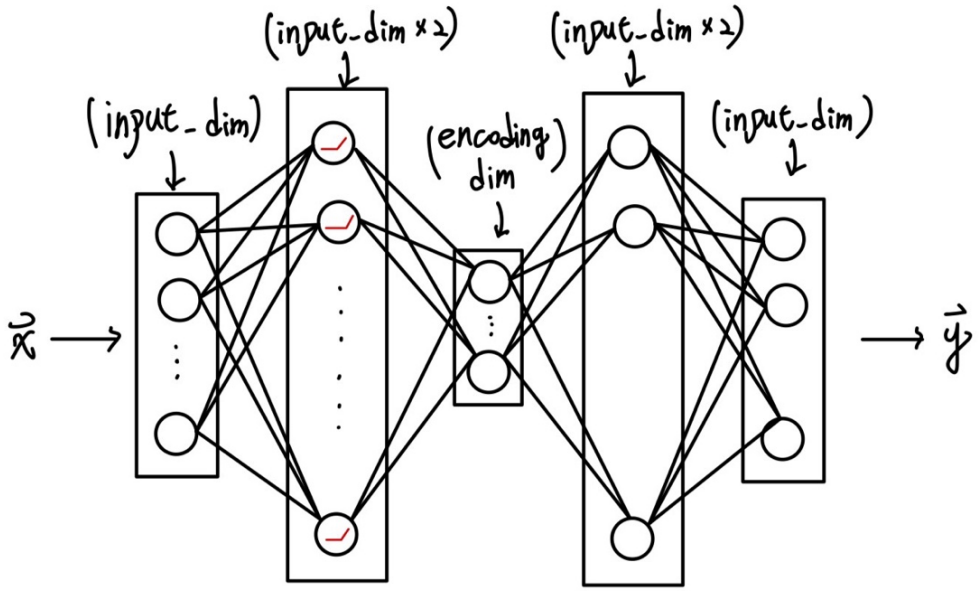


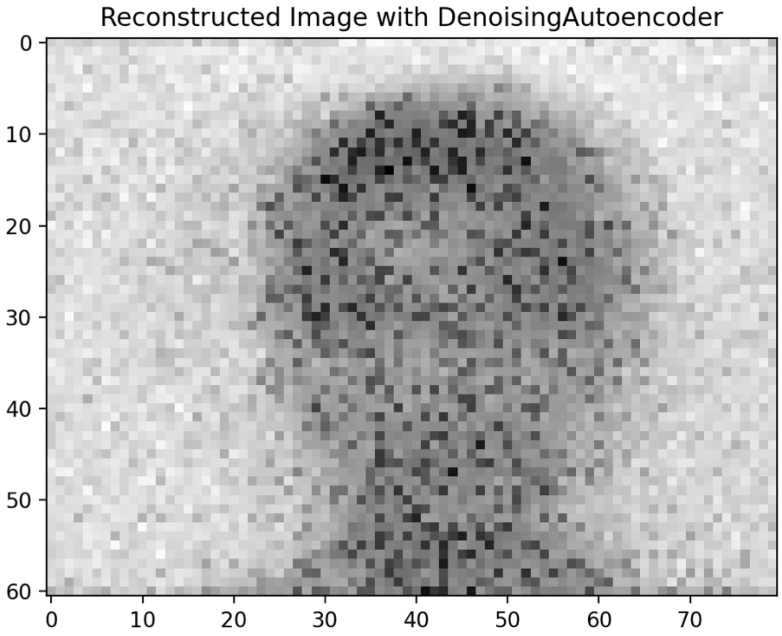


Reconstruction loss with original denoising autoencoder: 0.013515440799139819.

A fatter network:

Structure:



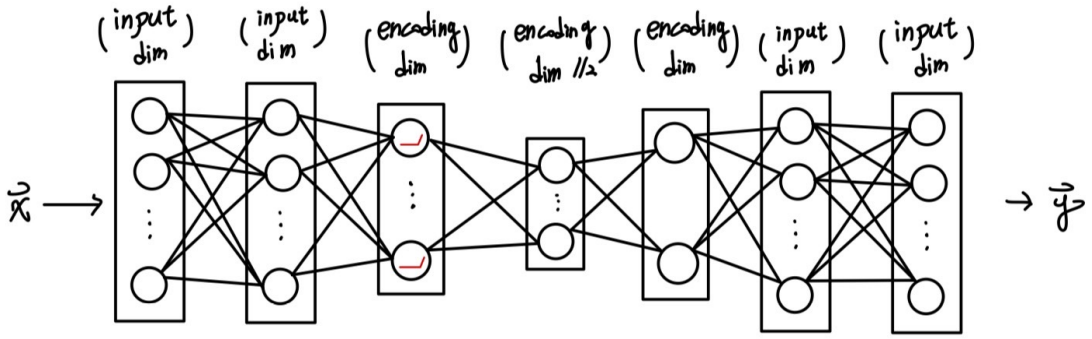


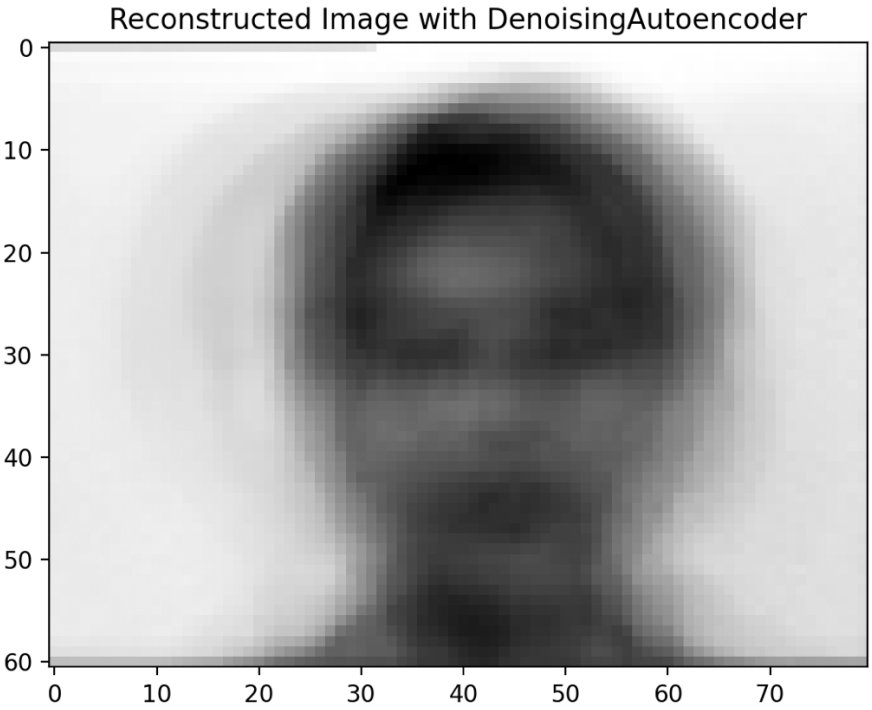
Reconstruction loss with fatter denoising autoencoder: 0.05010385245262015.

From the above experiment, making the model more complicated by making the second layer and the encoding layer two times fatter (comparing to the given sample code) will not yield to a better result with lower loss. This may be caused by overfitting since the model complexity increases.

A deeper network:

Structure:





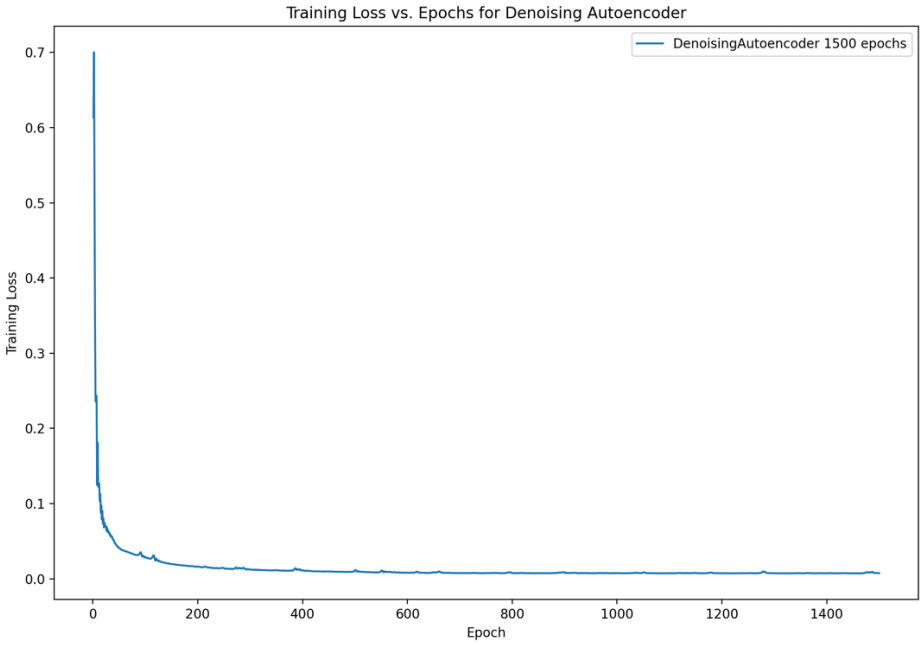
Reconstruction loss with deeper denoising autoencoder: 0.04208998344927163.

From the above experiment, making the model more complicated by adding one more layer with the same dimension of input (comparing to the given sample code) will not yield to a better result with lower loss. Again, this may be caused by overfitting since the model complexity increases.

(E) Different Optimizers on Denoising Autoencoder

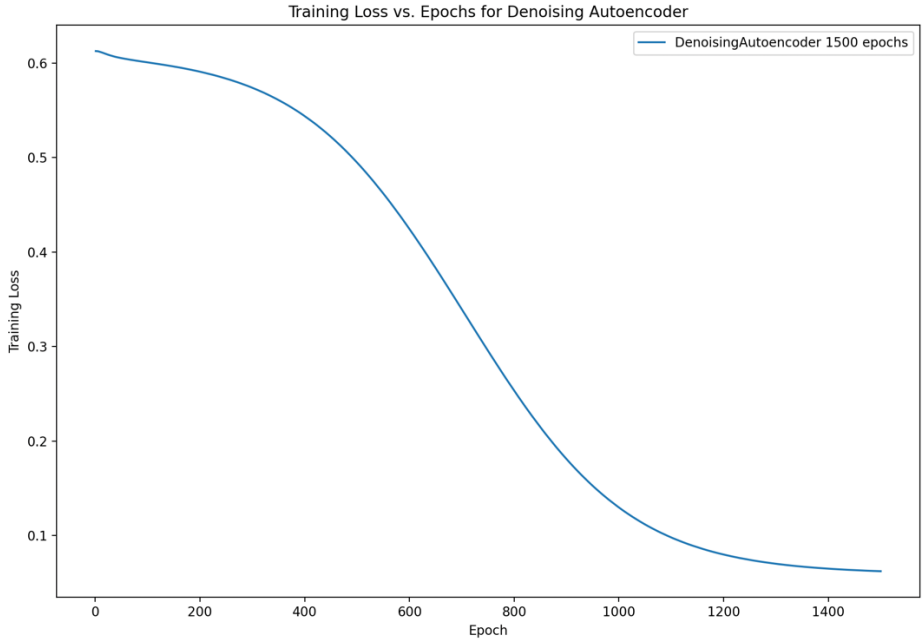
In this subproblem, the network structure is the same as the given sample code.

Adam optimizer (learning rate = 0.001):



Reconstruction loss with denoising autoencoder: 0.012514606808341346.

SGD (Stochastic Gradient Descent) optimizer (learning rate = 0.001, momentum = 0.9):



Reconstruction loss with denoising autoencoder: 0.03902042927189168.

From the above experiment, I used two different optimizers. One is the Adam optimizer; the other is the SGD optimizer. Both optimizers have learning rate = 0.001, and run 1500 epochs. We can see that Adam optimizer converges at around 300 epochs, and the SGD optimizer converges at around 1400 epochs, but with a smoother training curve.

In terms of reconstruction performance, the Adam optimizer have a better MSE loss.

If we use the denoising autoencoder with Adam optimizer to train the logistic regression model to make predictions, we can reach a 93.33% accuracy. However, if we use the denoising autoencoder with AGD optimizer to train the logistic regression model, we can reach a lower accuracy of 86.66%. But the difference is not significant.