Written Part

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 $\varphi'(s)$?

$\varphi'(s) = \theta(s) + s \cdot \theta'(s) = \frac{1}{1+e^{-s}} + s \cdot \frac{e^{-s}}{(1+e^{-s})^2} = \frac{1+e^{-s}+se^{-s}}{(1+e^{-s})^2}$ ✱

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.

(a.) At first, we have $v_0 = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]^T$, Therefore

$v_1 = Pv_0 = \begin{bmatrix} 0 & 1 & 1/2 \\ 0 & 0 & 1/2 \\ 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/6 \\ 1/3 \end{bmatrix}$,

$v_2 = Pv_1 = \begin{bmatrix} 0 & 1 & 1/2 \\ 0 & 0 & 1/2 \\ 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} 1/2 \\ 1/6 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/6 \\ 1/2 \end{bmatrix}$,

$v_3 = Pv_2 = \begin{bmatrix} 0 & 1 & 1/2 \\ 0 & 0 & 1/2 \\ 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} 1/3 \\ 1/6 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 5/12 \\ 1/4 \\ 1/3 \end{bmatrix}$,

$v_4 = Pv_3 = \begin{bmatrix} 0 & 1 & 1/2 \\ 0 & 0 & 1/2 \\ 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} 5/12 \\ 1/4 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 5/12 \\ 1/4 \\ 5/12 \end{bmatrix}$,

$v_5 = Pv_4 = \begin{bmatrix} 0 & 1 & 1/2 \\ 0 & 0 & 1/2 \\ 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} 5/12 \\ 1/6 \\ 5/12 \end{bmatrix} = \begin{bmatrix} 3/8 \\ 5/24 \\ 5/12 \end{bmatrix}$ ✱

(b.) Let $V^* = [a, b, c]^T$. Then $\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1/2 \\ 0 & 0 & 1/2 \\ 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} a \\ b \\ c \end{bmatrix} \Leftrightarrow \begin{cases} a = b + c/2 \\ b = c/2 \\ c = c \end{cases}$

which gives us $a = 2b = c$ after simplification. Since V^* is normalized, we have $V^* = [2/5, 1/5, 2/5]^T$ ✱

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.

(a.)

Cluster 1 : $\{(1,2)\}$ Cluster 2 : $\{(3,4),(7,0),(10,2)\}$ → $\{M_1, M_2\} = \{(1,2), (\frac{20}{3}, 2)\}$

Cluster 1 : $\{(1,2),(3,4)\}$ Cluster 2 : $\{(7,0),(10,2)\}$ → $\{M_1, M_2\} = \{(2,3), (8.5, 1)\}$

Cluster 1 : $\{(1,2),(3,4)\}$ Cluster 2 : $\{(7,0),(10,2)\}$ → $\{M_1, M_2\} = \{(2,3), (8.5, 1)\}$ → Converge.

(b.)

Cluster 1 : $\{(1,2),(3,4)\}$ Cluster 2 = $\{(7,0),(10,2)\}$ → $\{M_1, M_2\} = \{(2,3), (8.5, 1)\}$

Cluster 1 : $\{(1,2),(3,4)\}$ Cluster 2 = $\{(7,0),(10,2)\}$ → $\{M_1, M_2\} = \{(2,3), (8.5, 1)\}$ → Converge. , No different.

(c.)

Consider initial centroids : $\{M_1, M_2\} = \{(1,2),(7,0)\}$

Cluster 1 : $\{(1,2),(3,4),(5,6)\}$ Cluster 2 : $\{(7,0),(10,2)\}$ → $\{M_1, M_2\} = \{(3,4), (8.5,1)\}$

Cluster1 : $\{(1,2),(3,4),(5,6)\}$ Cluster 2 : $\{(7,0),(10,2)\}$ → $\{M_1, M_2\} = \{(3,4), (8.5,1)\}$ → Converge.

And consider initial centroids : $\{M_1, M_2\} = \{(1,2),(5,6)\}$

Cluster 1 : $\{(1,2)\}$ Cluster 2 : $\{(3,4),(5,6),(7,0),(10,2)\}$ → $\{M_1, M_2\} = \{(1,2), (6.25, 3)\}$

Cluster 1 : $\{(1,2),(3,4)\}$ Cluster 2 : $\{(5,6),(7,0),(10,2)\}$ → $\{M_1, M_2\} = \{(2,3), (7.33, 2.66)\}$

Cluster 1 : $\{(1,2),(3,4)\}$ Cluster 2 : $\{(5,6),(7,0),(10,2)\}$ → $\{M_1, M_2\} = \{(2,3), (7.33, 2.66)\}$ → Converge

※ We know that these two has different answers, not matter who is optimal, it do show that k-means alg. could converge to a local min. with certain centroids ※

Programming Part

(a)

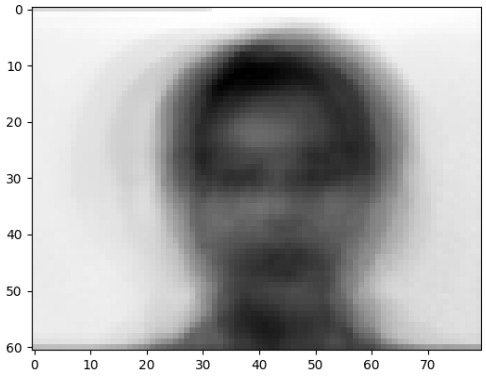


Fig1. Mean vector

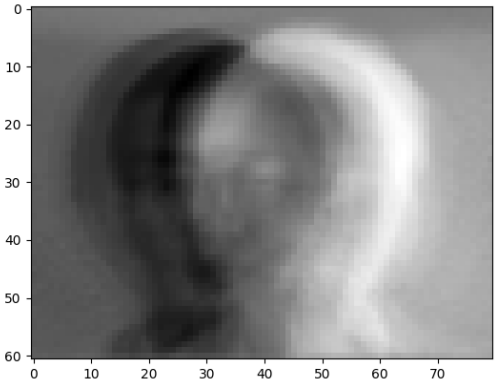


Fig.2 1st eigenvector

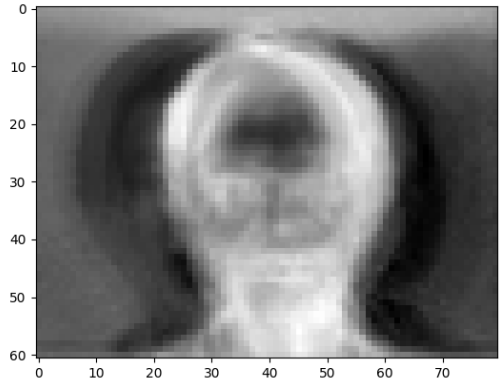


Fig3. 2nd eigenvector

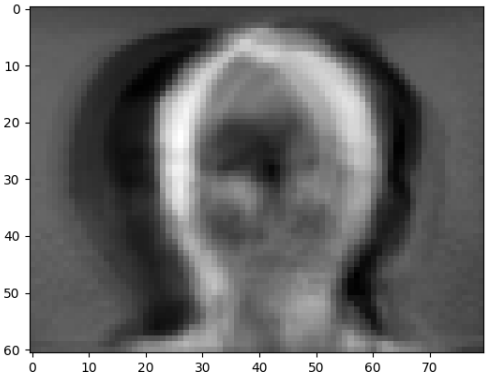


Fig.4 3rd eigenvector

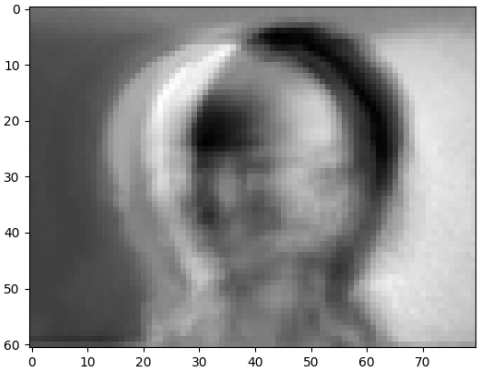


Fig5. 4th eigenvector

(b)

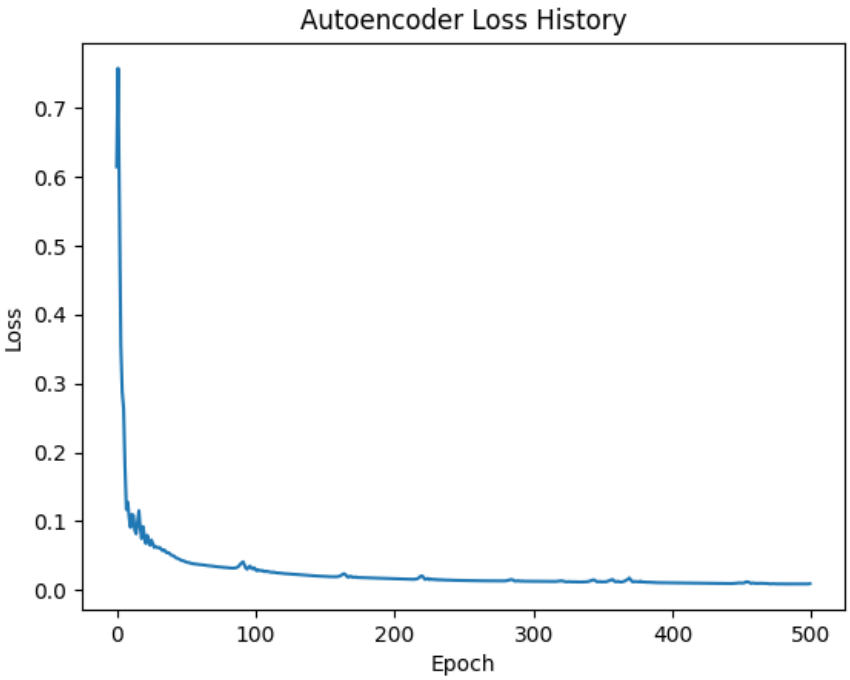


Fig6. Loss curve of Autoencoder

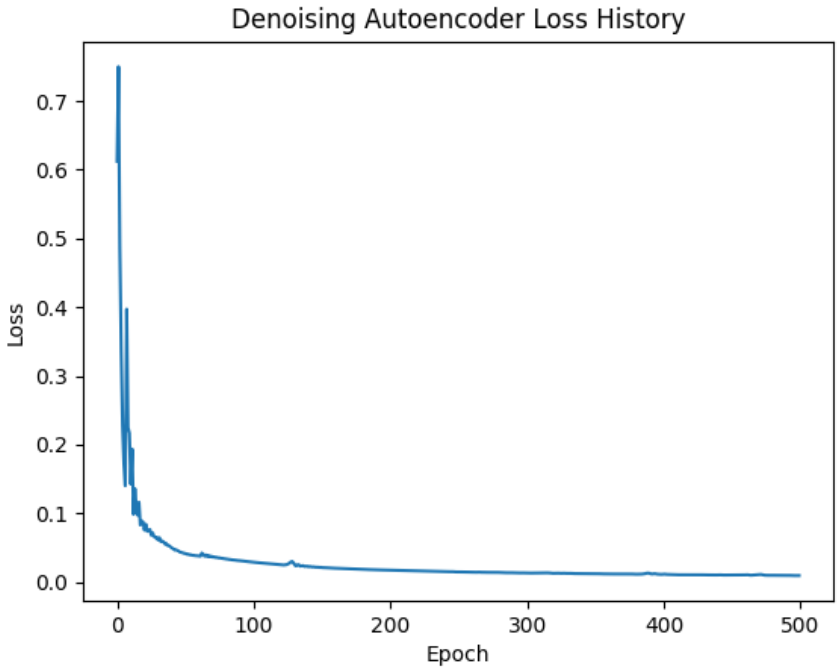


Fig7. Loss curve of DenoisingAutoencoder

(c)

Fig8. Original image

Fig9. Reconstructed with PCA (MSE= 0.010710469688056314)

Autoencoder Reconstructed Image

Fig10. Reconstructed with Autoencoder(MSE= 0.015646440532965353)

DenoisingAutoencoder Reconstructed Image

Fig11. Reconstructed with DenosingAutoencoder(MSE= 0.012897145575758865)

(d)
I've tried the deeper and shallower architecture with all the settings remains the same as the sample.
Original one: **Acc**: 0.9333333333333333 and **Reconstruction Loss**: 0.013306048900814367

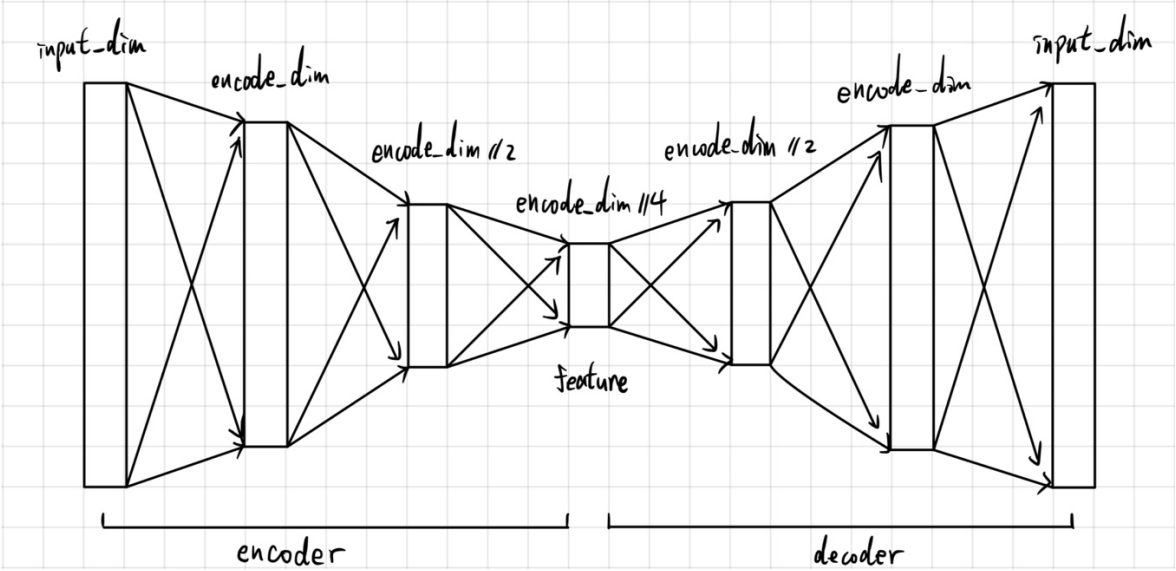


Fig12. The deeper architecture

The deeper one has the
Acc: 0.06666666666666667 and **Reconstruction Loss**: 0.042025545732162176
We can see that its performance is worse than the original one. I think it might because of the decoding would be harder for deeper model.

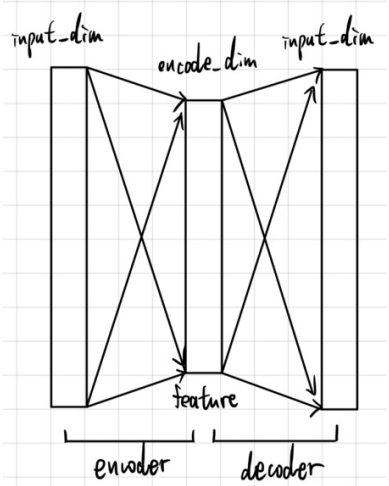
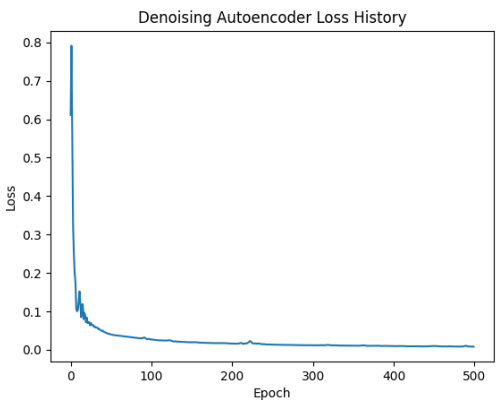


Fig13. The shallower architecture

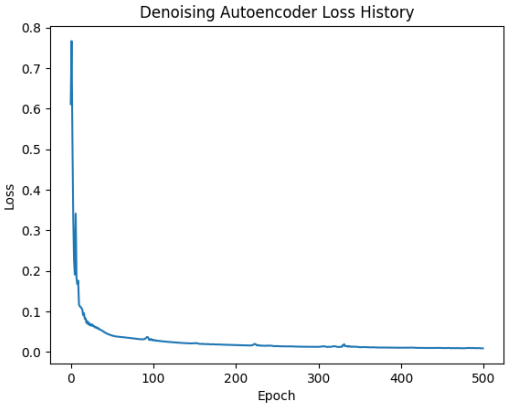
The shallower one has the
Acc: 0.8666666666666667 and **Reconstruction Loss**: 0.019420283759560463
We can see that the shallow one performs worse than the original one too. But has a slightly difference, this might suggest that the task is not that complex so the easy one could handle but still need the suitable settings to get the best performance. Which means that the architecture is truly important for the task.

(e)

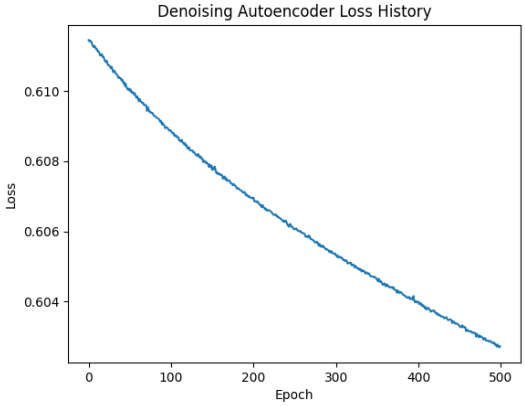
I've tested 5 different optimizers: Adam, AdamW, Adadelta, PMSpropk, and SGD.



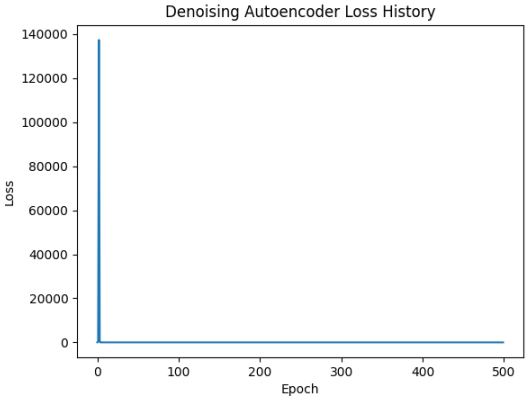
Adam



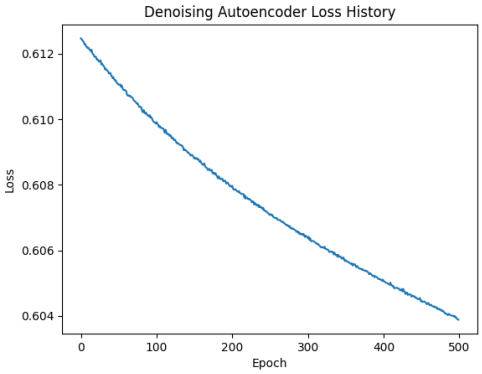
AdamW



Adadelta



RMSprop



SGD

Optimizer	Accuracy	Reconstruct Loss
Adam	0.9333	0.0133
AdamW	0.9333	0.0132
Adadelta	0.8667	0.6919
RMSprop	0.0667	0.1657
SGD	0.9000	0.6934

We can see that the Adam and AdamW have the similar results, which is pretty reasonable. And Adadelta's curve is converge so slow and has quite good accuract but bloddy terrible loss. In contrast, the RMSprop has bloody terrible accuracy but converge so fast with quite a large loss. And lastly, SGD has pretty good accuracy bt loss terrible as bad as Adadelta, which would oberved by the loss history of these 2 models, I think the trend indeed influence the loss.