Report for Exercise 4 from Group K

Tasks addressed:	4
Authors:	Yun Fei Hsu (03732414) Jan Watter (03665485) Yaling Shen (03734727)
Last compiled:	2021–01–29
Source code:	`https://github.com/ElsieSHEN/Representations-of-data-PCA-Diffusion-Maps-VAE`

The work on tasks was divided in the following way:
Equally for all group members and tasks.

Yun Fei Hsu (03732414)	Task 1	100%
	Task 2	100%
	Task 3	100%
	Task 4	100%
Jan Watter (03665485)	Task 1	100%
	Task 2	100%
	Task 3	100%
	Task 4	100%
Yaling Shen (03734727)	Task 1	100%
	Task 2	100%
	Task 3	100%
	Task 4	100%

Report on task 1, Principal component analysis

In the first task, we are supposed to implement Principal Component Analysis using a library method for Singular Value Decomposition. For this we use the *numpy.linalg.svd* function which gives back the matrices U, S and V. We use the S matrix which holds the singular values for the calculation of the explained variance. We see that the first Principal Component holds 99.31% while the second only holds 0.69%. The direction of the first Principal Component is encoded in the V matrix which gives *[-0.889, -0.457]*. We continue with plotting the two directions of the Principal Components as shown in Figure 1. The length of the arrows represents the explained variance in relation to each other. It is obvious that the first direction explains more of the energy compared to the second one. Due to the scaling of LaTex, it might look as though the directions are not orthogonal but in fact they are.

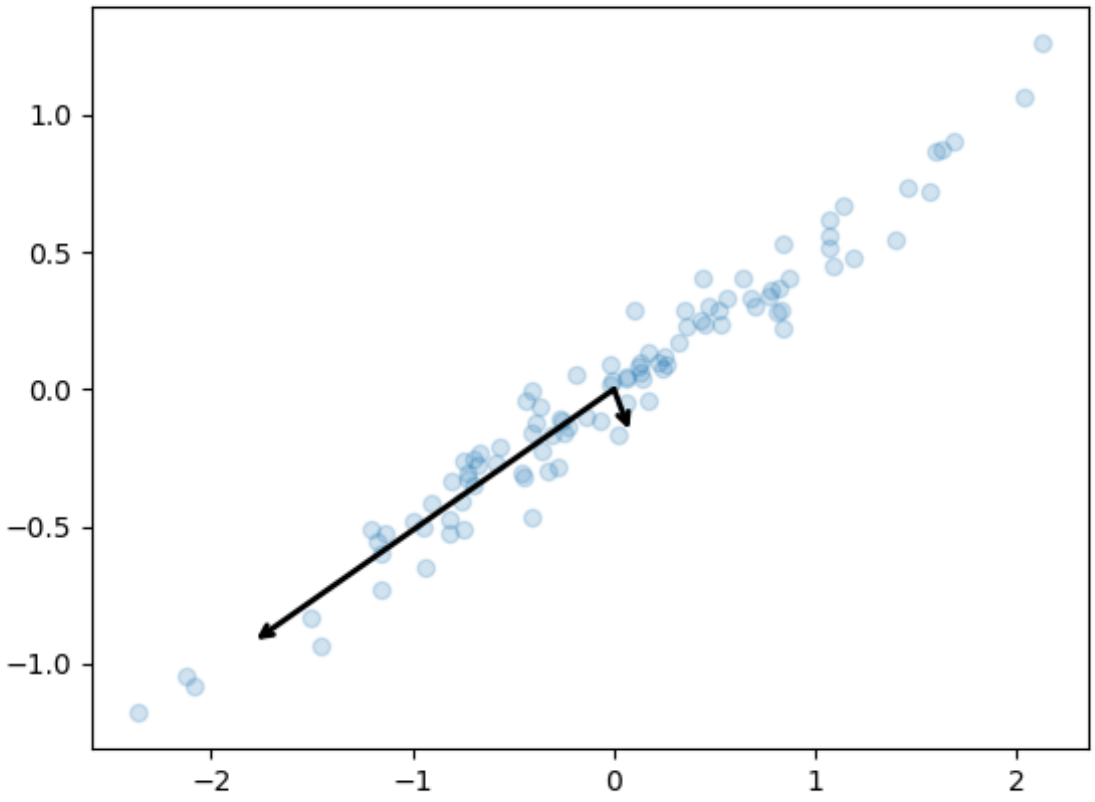

Figure 1: PCA dataset with directions of PCs

In the second part we are asked to apply PCA to an image of a raccoon. Initially we convert the image to gray-scale and re-scale the image to (249 x 185). As before we center our data and additionally set the standard deviation to one. This ensures our PCA will perform in the best way possible as mentioned in [1].
Analogously to the first part, we calculate the Singular Value Decomposition and the explained variance for each component. With L = 79 the energy lost through truncation is initially smaller than 1% with an explained variance of 99.0285.
We continue to reconstruct the image using all, 120, 50 and 10 Principal Components as shown in Figures 2 till 5. Using 50 PCs a visible information loss is visible. The image seems more blurry and the edges not as sharp as before. Again in the LaTex document this might not be visible since the images get scaled down and a difference is not visible. Using only 10 PCs a very clear information loss is visible even the re-scaled version in this document. To confirm the correctness of the explained variance, we use *sklearn.decomposition.PCA* which is a class from the scikit-learn library.

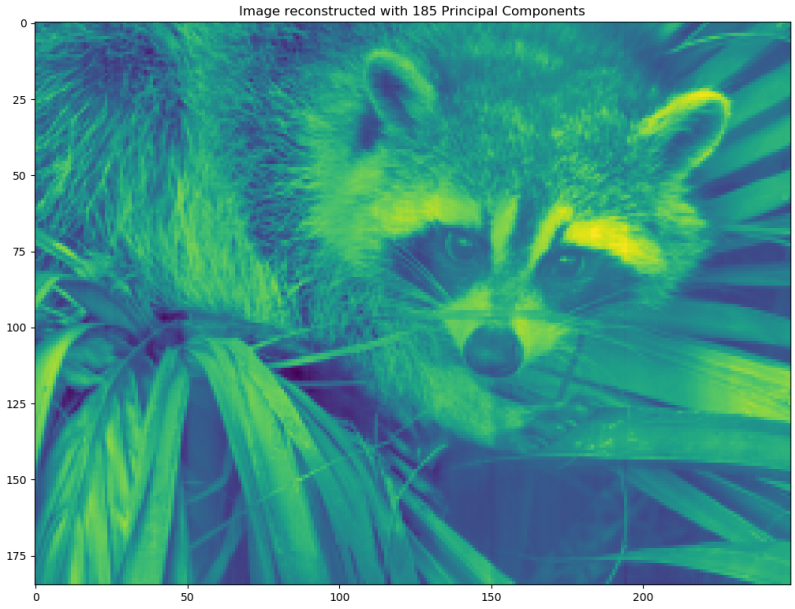

Figure 2: Image reconstructed using 185 PCs

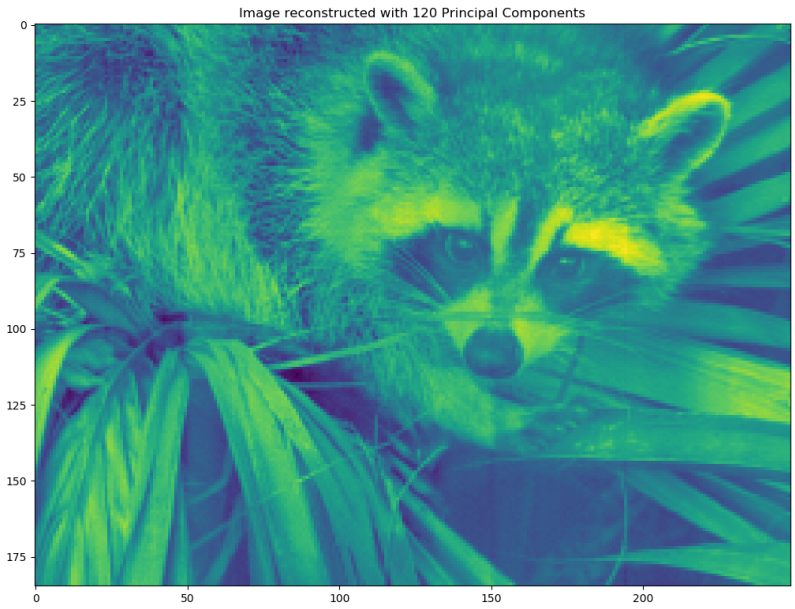

Figure 3: Image reconstructed using 120 PCs

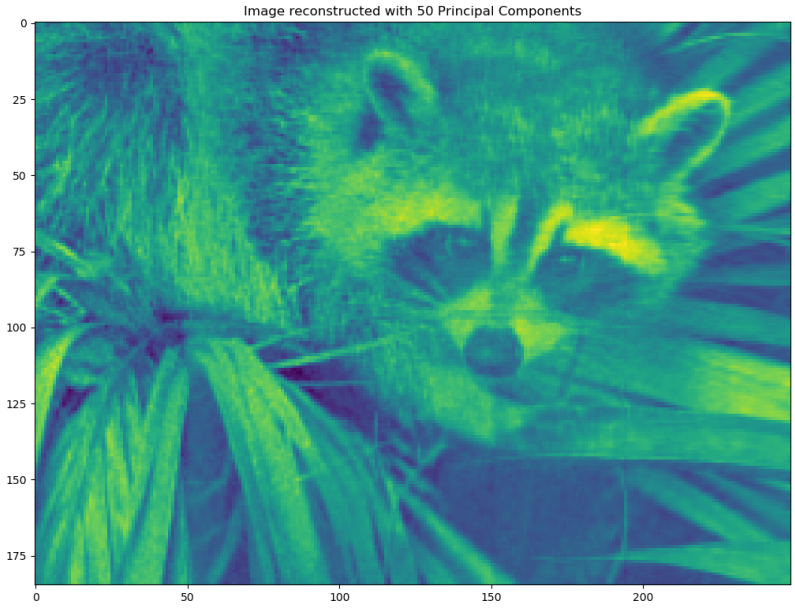

Figure 4: Image reconstructed using 50 PCs

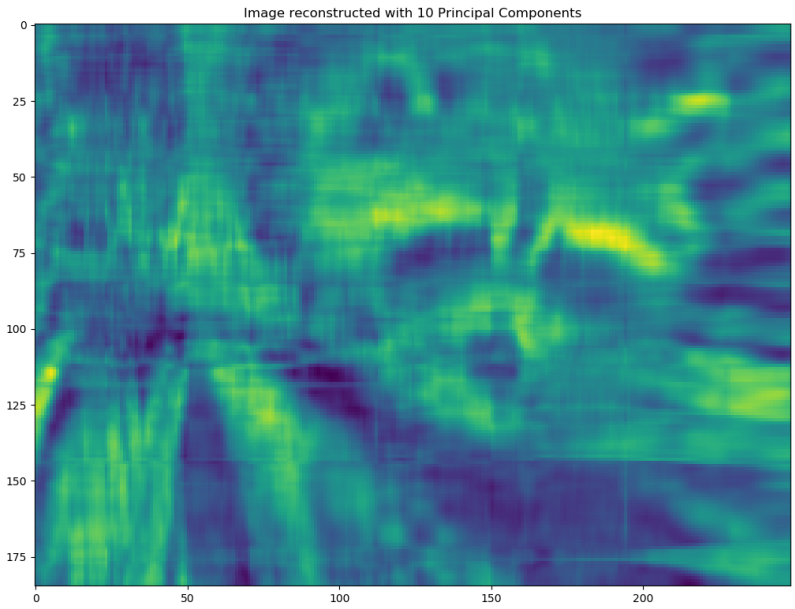

Figure 5: Image reconstructed using 10 PCs

In the third part we are asked to analyze a dataset containing the positional data of 15 pedestrians over 1000 time steps. Initially we visualize the path of the first two pedestrians. We create an animation using the *matplotlib.animation* module. For a proper visualization please run the code. The trajectory of pedestrian 1 is in black and pedestrian 2 in red. The final trajectory is shown in Figure 6. One can see that both of the pedestrian follow a "zig-zag" movement in contrast to a more smooth path. Furthermore both of the trajectories form circular motions while the turns the pedestrians take mirror themselves. Meaning when pedestrian 1 turns right to form a circular motion, pedestrian 2 will turn left. This holds except for the last turn where both of them turn left. Also they finish their walk almost at the exact point where they started. Maybe the two path-lines are topologically equivalent? Under the assumption that the ends are joined together, this could be studied further under knot theory or under the mathematical field of topology in general.

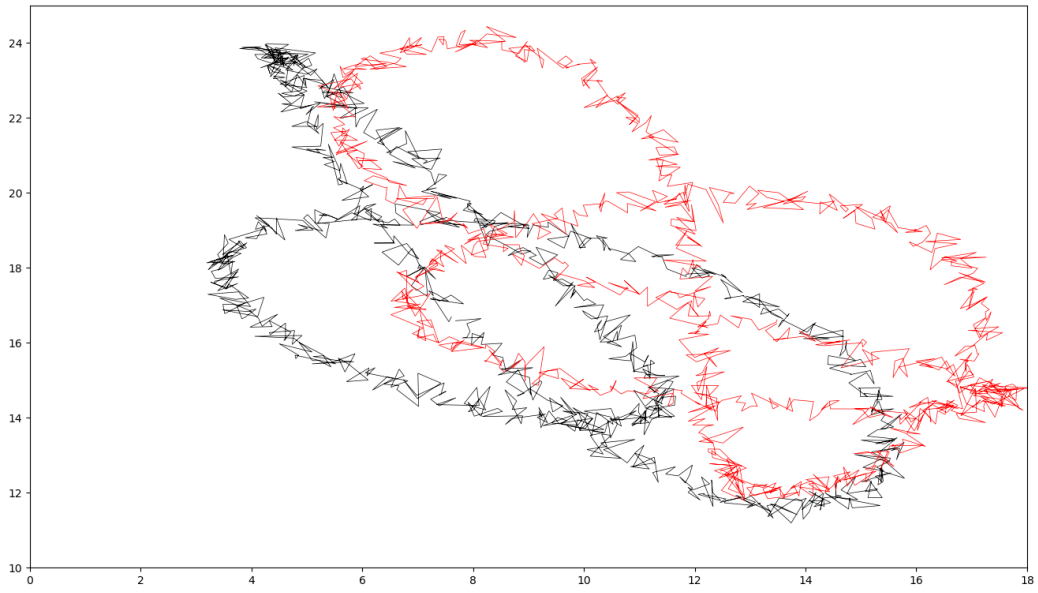

Figure 6: Trajectories of the first two pedestrians

In the second sub task of the third part, we shall apply PCA to our data and reduce the 30 dimensional dataset (15 pedestrians described by x- and y-coordinates) to two dimensions. Similar the tasks before we use SVD and manually compute the explained variance and the transformed dataset before confirming results using sklearn functions.
The first two components cover 84.35% of the energy, so by definition of the worksheet they do not cover most of the energy since this is $< 90\%$. Adding another PC the explained energy rises to 99.71% so most of the energy is explained. So the third PC adds another 15.4% variance to our 84.35% of the first two PCs. For more investigation: similar to the tasks before we created arrays with the absolute explained variance, the ratio of the explained variance and the cumulative explained variance in descending order.
The explained variance ration is calculated as follows:

$$S = \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_r \end{pmatrix}, \quad \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{j=1}^{r} \sigma_j^2} \tag{1}$$

The indices of the array belong to the corresponding PC starting at index 0, since we are using python.
PCA can be interpreted as finding new coordinate axis for our data that have the direction of the largest variance. Since these axis capture the "energy" in an optimal way, one is able to reconstruct the data using fewer dimensions compared to the original data. In this case this might be interpreted as all the possible paths that can be explained using the first 2 most important (as can explain most of the energy) PCs can explain

84.35% of all possible paths in the whole dataset. The dataset simply does not allow do capture more variance using two dimensions. Figure 7 shows the plot of the dataset projected to the first two PCs and Figure 8 shows the same projection calculated by sklearn. One can see that the two plots appear to be mirrored. Probably this is due to the sign ambiguity when normalizing the U vectors as explained in [2].

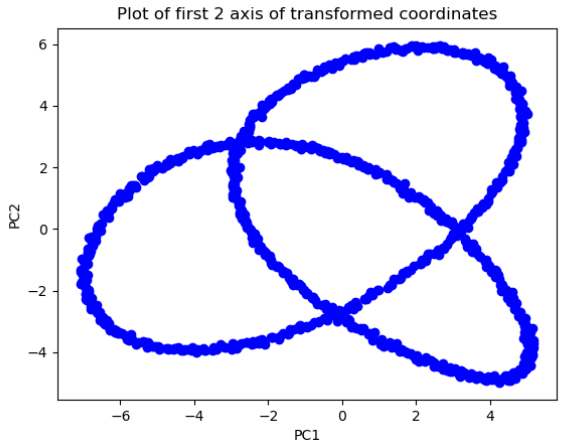

Figure 7: Dataset using 2 most imortant PCs

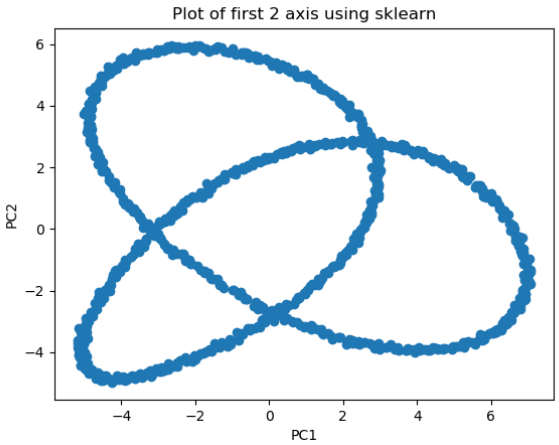

Figure 8: Dataset using 2 most imortant PCs using sklearn

Additional reporting:

(a) I did not clock my working time, so it is hard to tell. For the next worksheet I will use toggle to have a precise time measurement. I approximate one day for watching the videos from this worksheet and researching manifolds. Two days for a refresher of Linear Algebra and researching SVD and PCA. Then about 3 days of implementing and testing methods. And about one day writing report, clean up code and prepare repository. With one day, I mean an 8 hour workday. Retrospectively it does not seem that much work and the topic does not seem that complex anymore but it took a while to get there.

(b) I guess the question about how accurate the data could be represented in this task refers to the explained energy in this task. As mentioned in the report, an explained variance array is computed for each task and are described in the report of the tasks. Since we rely on some numpy functions there could be some rounding errors or truncation errors. Also I am also not familiar with the machine precision when using python and a numerical analysis of such errors was not conducted but I assume these kind of errors are negligible for the given tasks. I compared the explained variance that I calculated with the one calculated by sklearn. Both of them are calculated with six decimal places and they matched perfect in this precision.

(c) I got an introduction to manifolds and since I did not learn about topology in any lecture so far, it seems still complex but I developed an intuitive understanding about basic manifolds. Then I had to refresh my knowledge about Linear Algebra. Here I looked into: Eigenvalues and Eigenvectors (Singular values and Singular vectors, Pseudoinverse, Variance and Covarinace(-Matrices) and Orthgonality. With this SVD and PCA could be understood. I continued to watch a lot of videos about PCA and researched some implementations. About the datasets, I understood how one can classify them in features/variables and samples and how PCA can determine which features explains how much of the data and how it finds a new set of axis to describe the data. For the first dataset, I learned how the directions and the scaling of PCA worked. In the second part, I learned about reconstruction and information loss when doing this kind of dimensionality reduction. This is useful for Deep Learning with image data. The last task felt a bit counter intuitive but it was interesting to use PCA even for positional data and showed that one can get rid of many dimensions with very few information loss.

Report on task 2, Diffusion Maps

Part One We compute five eigenfunctions ϕ_l associated to the largest eigenvalues λ_l with Diffusion Maps, on a periodic data set with $N = 1000$ points given by:

$$X = \{x_k \in \mathbb{R}\}_{k=1}^{N},\ x_k = (\cos(t_k), \sin(t_k)),\ t_k = (2\pi k)/(N+1) \tag{2}$$

Then we plot the values of the eigenfunctions $\phi_l(x_k)$ against t_k in Figure 9.

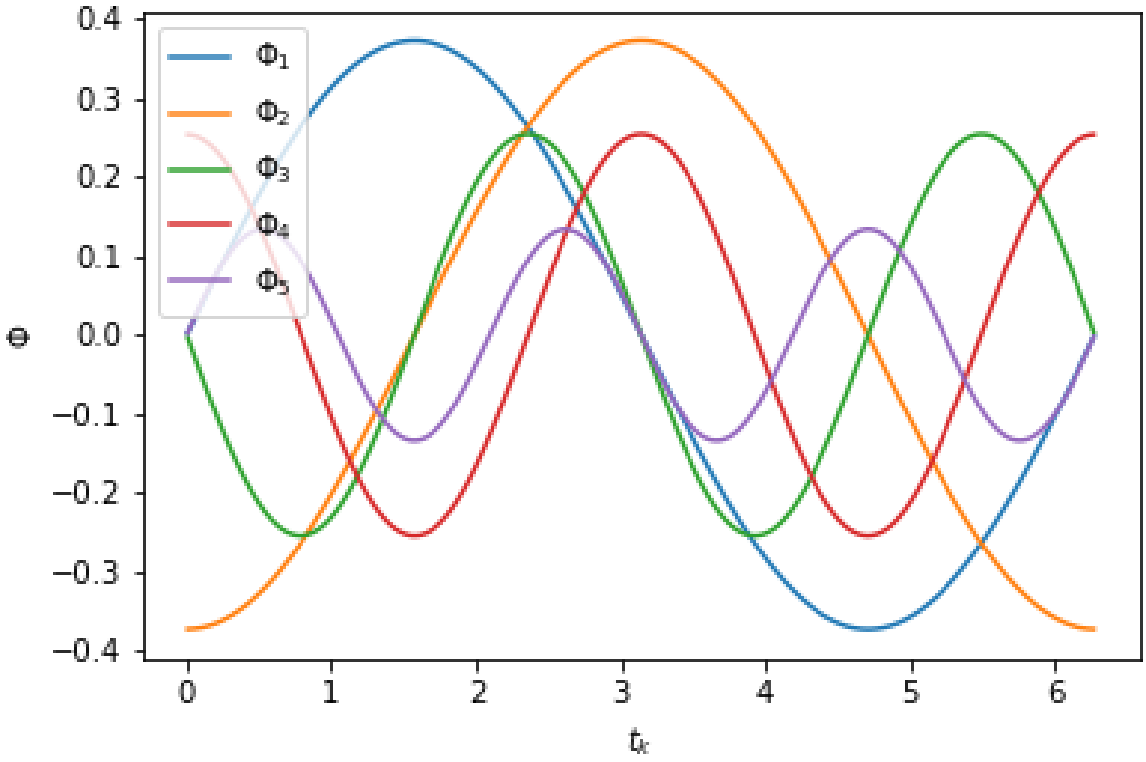

Figure 9: Values of $\phi_l(x_k)$ against t_k

The five eigenfunctions looks like trigonometric functions and it is possible that they can be represented by the summation of sine and cosine functions. Then if we apply Fourier analysis, we may able to represent the given function by a series of the form:

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}\cos(nx) + \sum_{n=1}^{\infty}\sin(nx) \tag{3}$$

Part Two In this part, study on the Laplace Beltrami operator on the "swiss roll" manifold, defined through:

$$X = \{x_k \in \mathbb{R}\}_{k=1}^{N},\ x_k = (u\cos(u), v, u\sin(u)) \tag{4}$$

where $(u, v) \in [0, 10]^2$.

We first use `sklearn` Python library to generate the swiss-roll data set with 5000 data points in three-dimensional space without noise, which is shown in Figure 10.

Then we use Diffusion Maps algorithm to obtain approximations of the eigenfunctions, and plot the first non-constant eigenfunction ϕ_1 against the other eigenfunction in 2D plots in Figure 11. When $l = 2, 3, 4$, the eigenfunctions ϕ_l are functions of ϕ_1. However, when $5 \leq 10$, ϕ_l are not functions of ϕ_1 anymore.

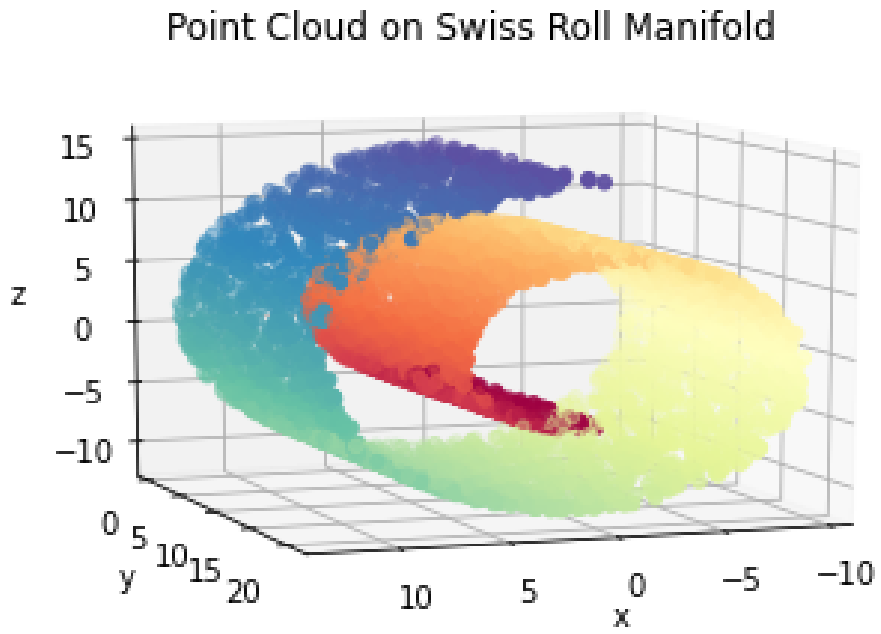

Figure 10: Swiss Roll Manifold

We then use PCA with three principal components to reduce the data set. We find that first three pincipal components take up $[0.40, 0.32, 0.28]$, respectively. First two principal components only take up 72% data information in total, which shows that we would lose 28% of the data feature if we only use two principal components to represent the data. Therefore, it is impossible to only use two principla components to represent the data.

We then change the data points from 5000 to 1000. Figure 12 shows the relation between the first non-constant eigenfunction ϕ_1 and other eigenfunction in 2D plots. With less data points, the data points are more discrete and when $l \geq 5$, there is even no clear line formed by those scatter points.

When applying PCA with three principal components, they take up $[0.38, 0.33, 0.29]$), respectively. Similar to the situation when there are 5000 data points. Only using first two principal components will lose a lot of information, which is impossible to represent the data.

Part Three We have already analyzed this data set with PCA in task 1. Setting the number of components to be 3, we find that first three principal components takes up $[0.47, 0.38, 0.15]$. First two components take up 85%, which is not large enough for us to think of ignoring the third component.

We then apply Diffusion Maps with three eigenfunctions on this data set and plot them in Figure 13. We also plot these three eigenfunctions in 3D as shown in 14. In first two eigenfunctions, there is little color coverage (one specific data point has the same color in different plot) while in first and third eigenfunctions, there is much color coverage. From this perspective of view, only using two eigenfunctions is enough.

We use the `datafold` to compute the eigenvectors of the swiss roll data set and plot them against Ψ_0 in Figure 15. Similar to the scenario in part two where there are 5000 data points, when $l = 1, 2, 3, 4$ (especially when $l \leq 3$), the eigenvector Ψ_l can be seen as a function of Ψ_0. However, when $l \geq 5$, Ψ_l is no longer a function of Ψ_0.

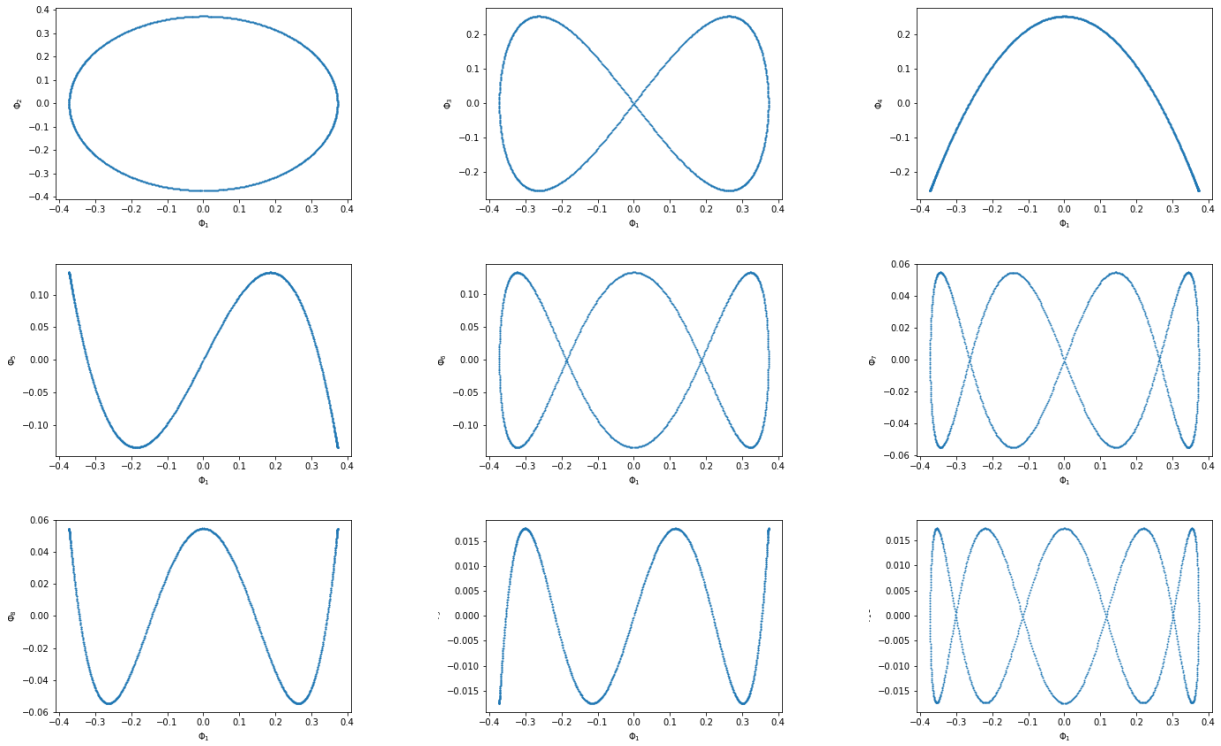

Figure 11: First Non-constant Eigenfunction ϕ_1 Against Other eigenfunctions with 5000 Data Points

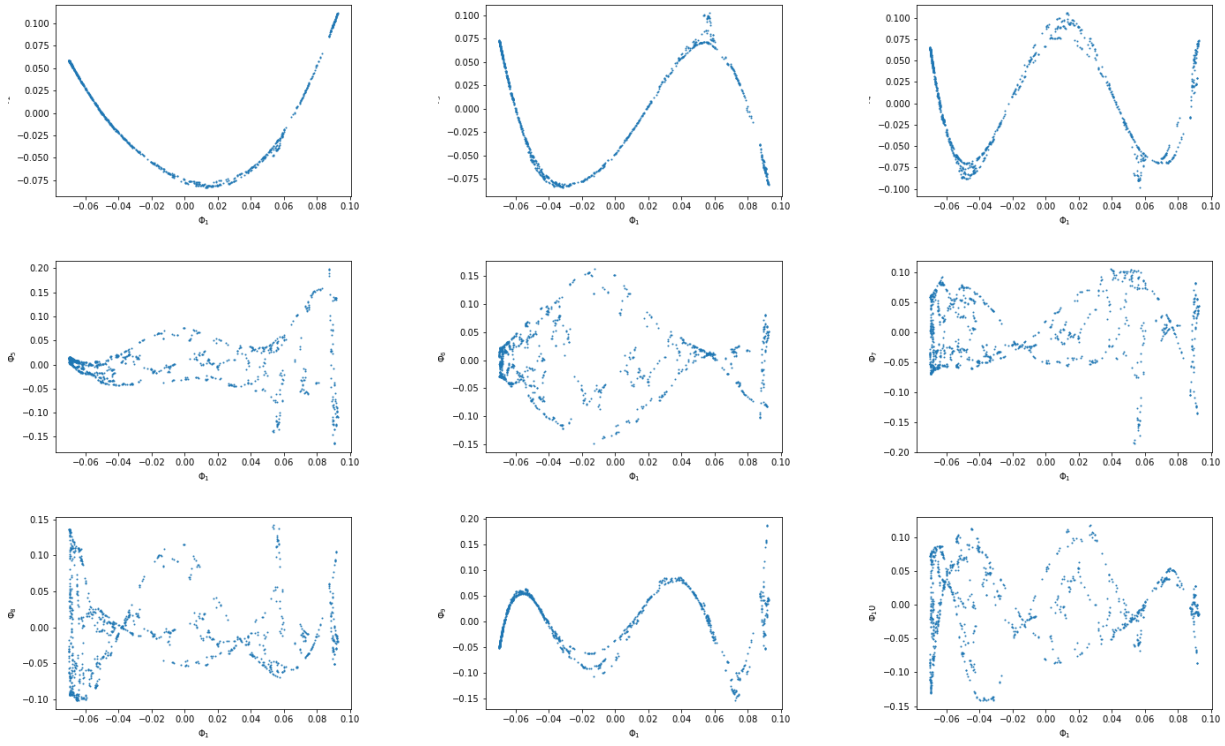

Figure 12: First Non-constant Eigenfunction ϕ_1 Against Other eigenfunctions with 1000 Data Points

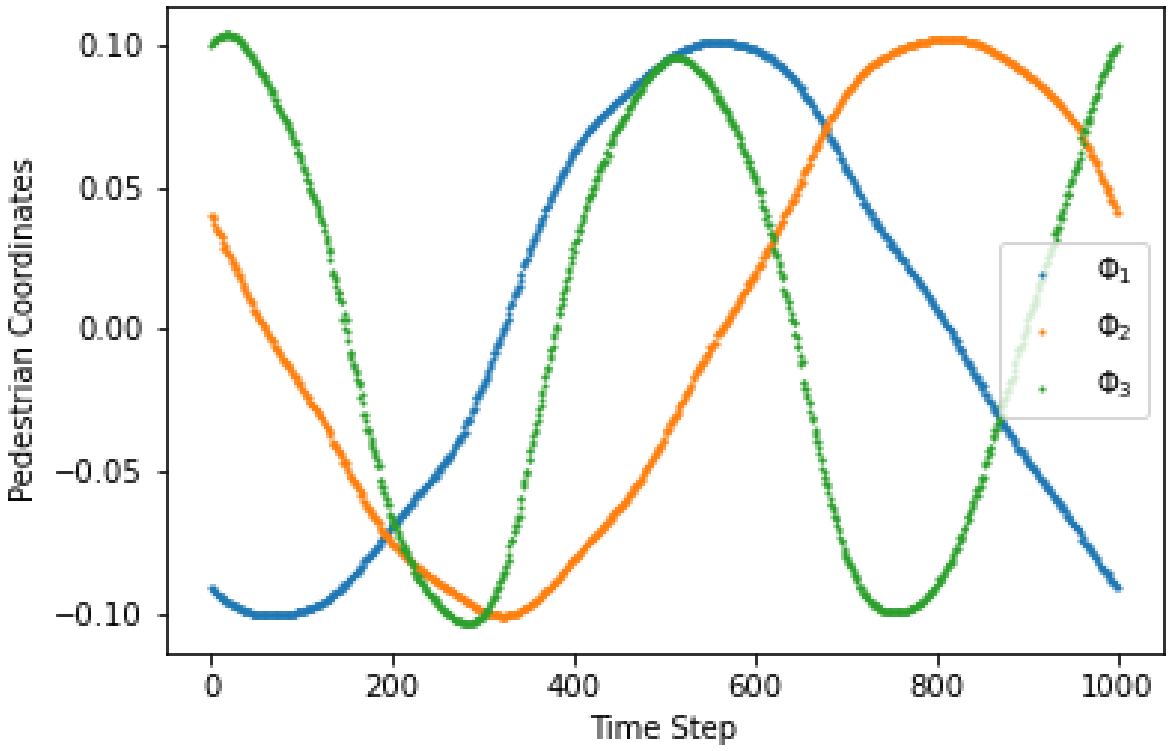

Figure 13: Pedestrian Coordinates

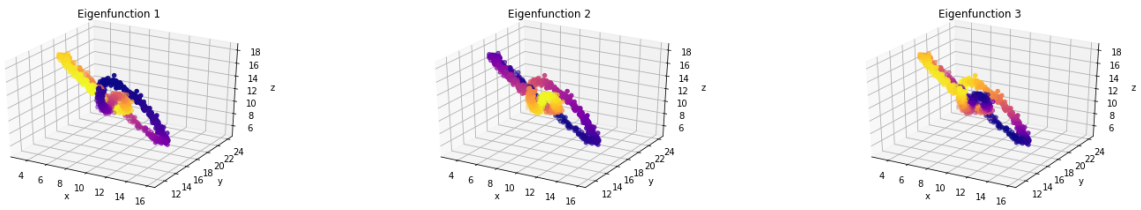

Figure 14: Eigenfunctions 3D Plot

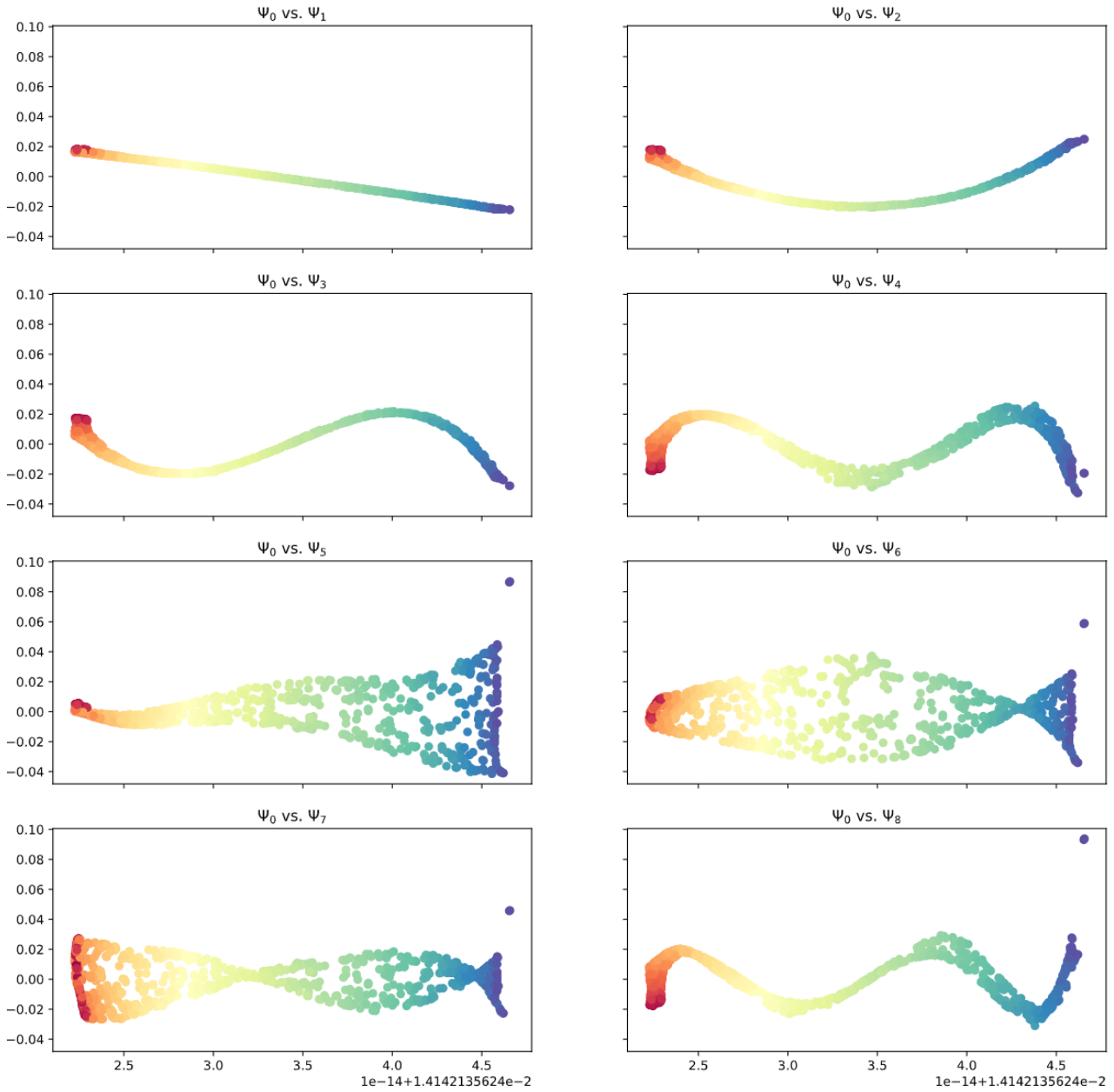

Figure 15: Pairwise Eigenvectors Against Ψ_0

Additional reporting:

(a) It took me about one day to watch the video, read related materials, and realize the Diffusion Maps method, which is the `diffusion_maps.py`. Then another day to implement the method to finish tasks stated in exercise sheet. And finally about one day to clear up the source codes and images, and write the report.

(b) I did not exactly measure the accuracy of the data representation. Maybe when applying PCA to see the differences between PCA and Diffusion Maps is one method to manually check the accuracy. Then plotting figures is another to see what happened to the original data matrix when applying Diffusion Maps.

(c) These methods are used to reduce the dimension of the data set. For PCA, it reduces the dimension linearly, and Diffusion Maps is an extension of the PCA, which reduce the dimension non-linearly. However, they all realize the target by matrix transformations. We can also take the matrix transformation as changing the basis of the coordinates of the data. Swiss-roll data set just looks like its name in 3D plot.

Report on task 3, Training a Variational Autoencoder on MNIST

In this task, we are supposed to train a Variational Autoencoder on the MNIST dataset. We use TensorFlow as the machine learning framework since we don't have experience on PyTorch or TensorFlow. We obtain the dataset in the same way as the tutorial provided in the exercise sheet. The dataset was normalized between 0 and 1, and split into a training set with 55,000 data, a validation set with 5,000 data and a test set with 10,000 data.

1 For posterior, we do not use any activation functions for the mean and standard deviation because the parameters we trained to infer latent vectors can be any float number. However, for the likelihood, we add a softplus activation function to avoid the standard deviation, which generated from the low-dimension latent vectors, to be too close to zero.

2 One reason might be the lack of regularisation. If the KL divergence is too small, the decoder might not generalize enough to decode new samples generated from prior, and the model might collapsing to the deterministic autoencoder. So we obtain good reconstructed but bad generated digits because of the bad performance of generalization.

3 (a) We obtain latent vector z and test label data to generate latent vector scatter plot. See Figure 16.

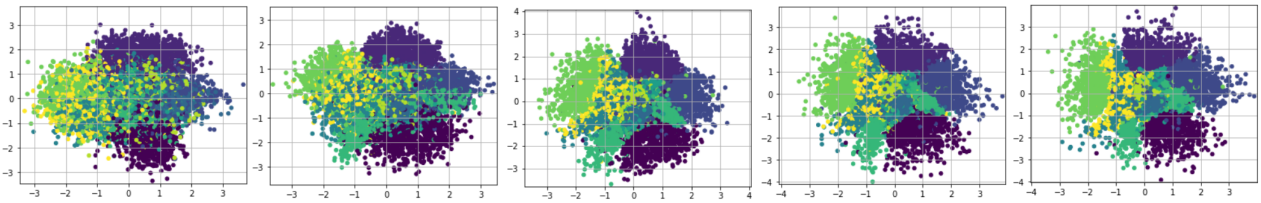

Figure 16: Latent Representation After Epoch 1, 5, 25, 50 and 104 (from left to right)

(b) For this task, we plot 15 reconstructed digits with their corresponding original image in the same row. The corresponding image is on the right side of each reconstructed digits. See Figure 17.

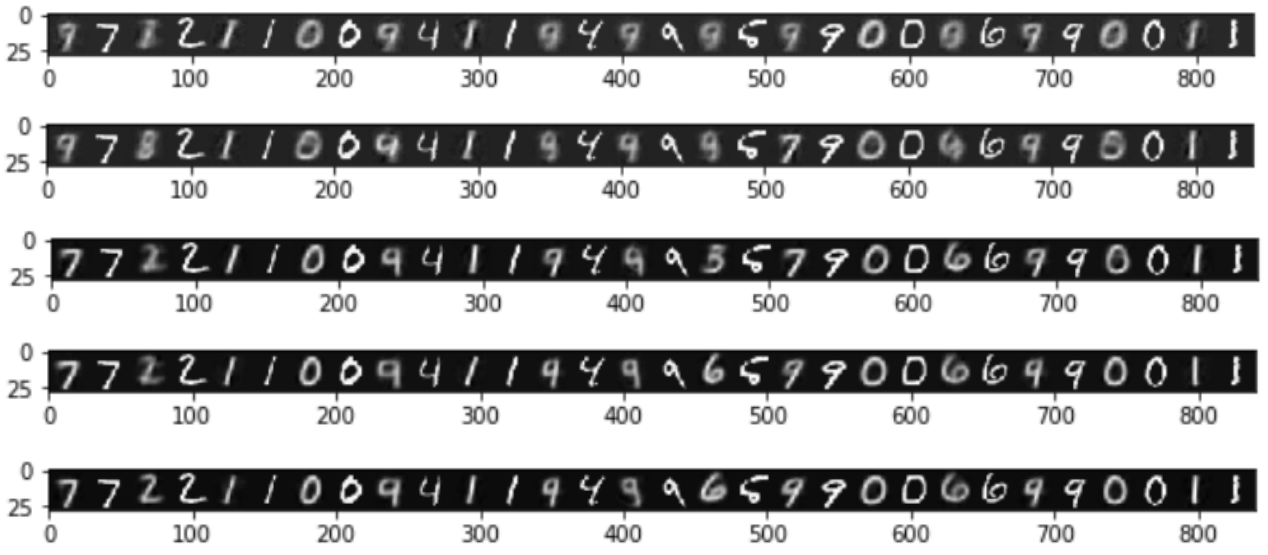

Figure 17: Reconstructed Digits After Epoch 1, 5, 25, 50 and 104

(c) We obtain samples from prior distribution and generate digits through the decoder. See Figure 18.

4 The loss curve is shown in Figure 19. It shows the right angle at 1st epoch means that the loss drops dramatically after the first training, but does not improve too much in the rest of the training.

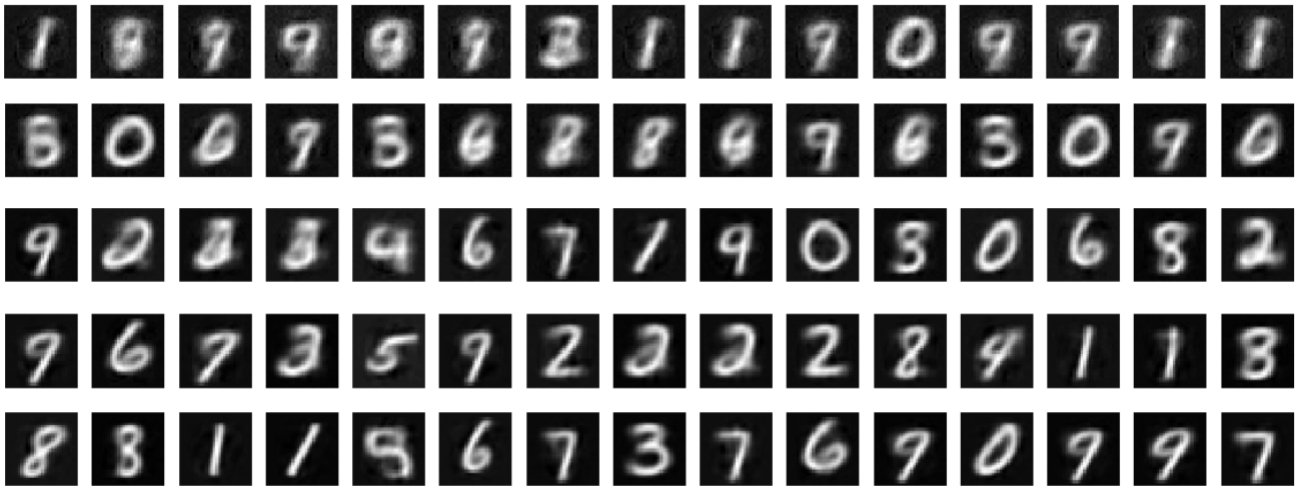

Figure 18: Generated Digits After Epoch 1, 5, 25, 50 and 104

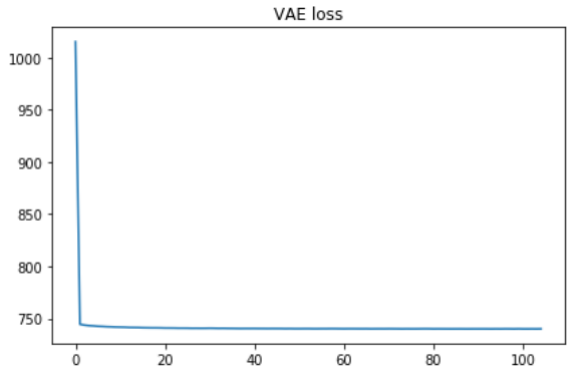

Figure 19: Loss Curve of 2-Dimensional Latent Space

5 (a) After training the VAE using a 32-dimensional latent space, the generated digits are shown in Figure 20. Compared with the generated digits under 2-dimensional latent space, there is no distinct difference between the two images. For our VAE model, simply increasing the dimension of latent space is not enough to improve the performance.

Figure 20: Generated Digits of 32-Dimensional Latent Space

(b) The loss curve is shown in Figure 21. As the previous curve, the loss does not improve too much after the second epoch. One possible reason is that the learning rate is set to be too large. The loss of the model will then oscillate over epochs and never reach the local (or global) minimum.

Additional reporting:

(a) I spend about two days to watch the lecture video, do the research on Variational Auto-Encoder and TensorFlow, and read the related materials. Since I have no experience in deep learning, clarifying all the steps is quite a challenge for me. It took me one full day to implement the method, but another two full days to test and debug. Then another one day for plotting and writing reports. I worked about 13 hours for each full day, and about 8 hours for the rest of the days.

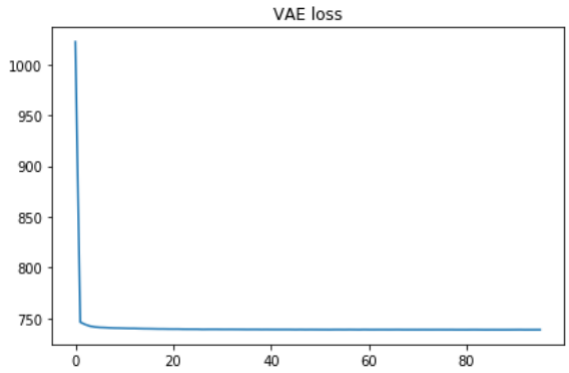

Figure 21: Loss Curve of 32-Dimensional Latent Space

(b) Since we use Variational Autoencoder as unsupervised learning, the data is said to be accurate means the generated digits is similar to the original data. We use the ELBO loss of test set to measure the accuracy, the lower loss represents the better performance of the model.

(c) When researching the material on deep learning, I have a brief understanding of neural networks, different types of activation functions, and what those activation functions are used for. For Autoencoder, it extract the most valuable features of the input by reducing the dimension. Then we can generate the output image based on these latent features. Then I got introduce to Variational Autoencoder which is a practical way to implement Autoencoder. I have to refresh my knowledge about multivariate diagonal normal distribution, prior and posterior. I also learn the mathematical analysis of the KL divergence between two multivariate normal distribution. For the dataset, what latent vector means for the digits image might be some obvious features such as circle and line. Theoretically, adding more dimension of the latent space will extract more detail features of the digit images, then the model can generate more accurate data.

Report on task 4, Fire Evacuation Planning for the MI Building

1. The scatter plot of the dataset is shown in Figure 22.

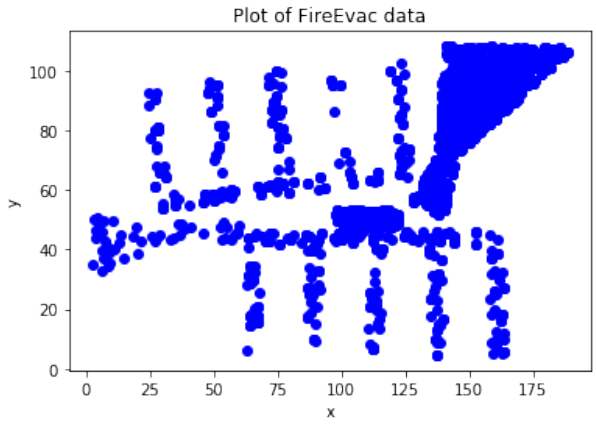

Figure 22: FireEvac scatter plot

2. We tried to train a VAE with the code from task 3 but due to limited time could not finish it.

References

[1] Dr. Mike (https://stats.stackexchange.com/users/5764/dr-mike). *Why do we need to normalize data before principal component analysis (PCA)?* Cross Validated. URL:https://stats.stackexchange.com/q/69159 (version: 2019-08-12). eprint: `https://stats.stackexchange.com/q/69159`. URL: `https://stats.stackexchange.com/q/69159`.

[2] Rodrigo de Azevedo (https://math.stackexchange.com/users/339790/rodrigo-de-azevedo). *Calculating SVD by hand: resolving sign ambiguities in the range vectors.* Mathematics Stack Exchange. URL:https://math.stackexchange.
(version: 2017-12-04). eprint: `https://math.stackexchange.com/q/1805239`. URL: `https://math.stackexchange.com/q/1805239`.