Report for exercise 3 from group F

Tasks addressed:	4
Authors:	Yuxuan Wang (03767260)
	Shihong Zhang (03764740)
	Mengshuo Li (03792428)
	Augustin Gaspard Camille Curinier (03784531)
	Qingyu Wang (03792094)
Last compiled:	2024–05–26

The work on tasks was divided in the following way:

Yuxuan Wang (03767260) **Project lead**	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%
Shihong Zhang (03764740)	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%
Mengshuo Li (03792428)	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%
Augustin Gaspard Camille Curinier (03784531)	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%
Qingyu Wang (03792094)	Task 1	20%
	Task 2	20%
	Task 3	20%
	Task 4	20%

Report on task 1, Principal component analysis

In this task, we implemented the Principal Component Analysis. According to the exercise sheet, we first form the data matrix and centered the matrix by removing the data mean from every row. Then, decompose the centered data matrix into U, V and S.

$$\overline{X} = USV^T \tag{1}$$

After that, we computed the energy of the principal components. Combine with PCA and energy, we reconstructed pictures and trajectories and performed analysis on them. All the results can be found in the file Task1_PCA.ipynb.

Part 1.1: How much energy is contained in each of the two components? The compute_svd function in utils.py use the np.linalg.svd function given by numpy. It returns U, S and Vt. By calling compute_svd we can get the principle components. To compute energy, we implemented compute_energy method according to

$$\frac{1}{trace(S^2)} \sum_{i=1}^{L} \sigma_i^2 \tag{2}$$

From the calculation results we can see that, about 99.314% is contained in the first component and about 0.685% in the second one.

Part 1.2: Plot the data set as in the figure, add the direction of the two principal components. We can see the data and two principal components from the figure below.

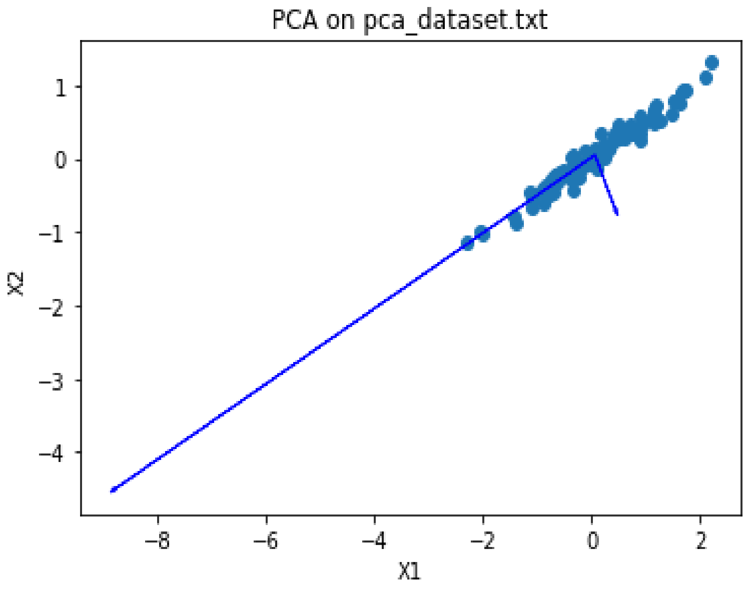

Figure 1: Data and the corresponding principal components

Part 2.1: Visualize reconstructions of the image (4 plots: all, 120, 50, and 10 components) In this part, we were asked to reconstruction a picture of racoon with (a) all, (b) 120, (c) 50 and (d) 10 principal components. We first load image from scipy.misc.face and resize it into (185, 249). After centered the data, we can perform a PCV using the same function of Part 1. With the U, S, Vt matrix, we can reconstruct the picture using the method provided in the exercise sheet. From the pictures, we can observe that the reconstruction result differ when using different number of principle components. Especially when using only 10 principal components, the picture becomes very blurry. The information loss become visible.

Part 2.2: At what number is the energy lost through truncation smaller than 1%? By computing the cumulate_energy we can notice that when the number is 76, the energy lost through truncation smaller than 1%.

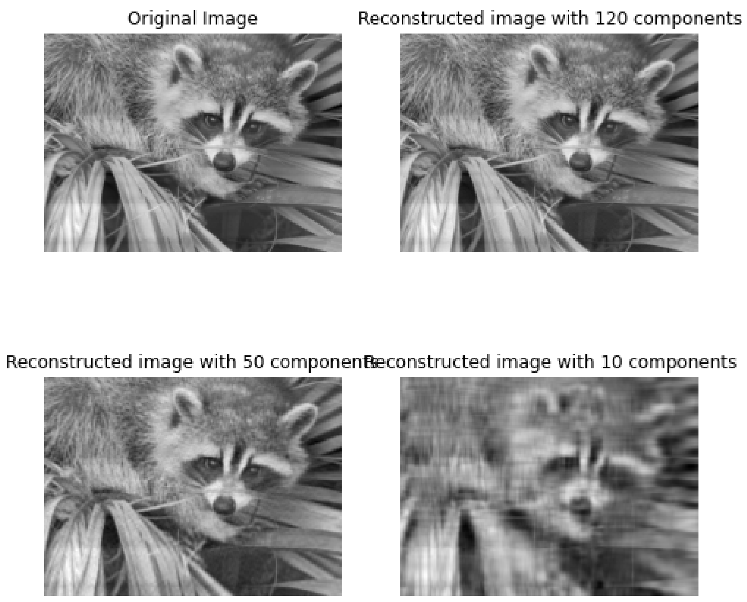
Original Image
Reconstructed image with 120 components
Reconstructed image with 50 components
Reconstructed image with 10 components

Figure 2: Visualize reconstructions of the image using all, 120, 50, and 10 components

Part 3.1: Visualize the path of the first two pedestrians in 2D space. In part 3, we used the trajectory data in the file data DMAP_PCA_vadere.txt and analyzed them. First, visualize the path of the first two pedestrians in 2D space. From the figure we can notice that, the path of both of them are in a particular

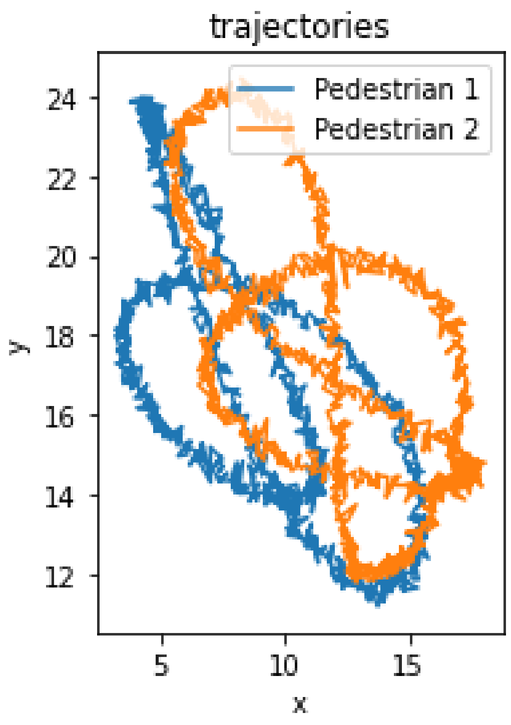

Figure 3: Visualize the path of the first two pedestrians in 2D space.

shape. Maybe they are walking along a certain path or they are having some events.

Part 3.2: Analyze the data set by projecting the 30-dimensional data points to the first two

principal components We can reconstruct the path figure using the method mentioned above. we can notice

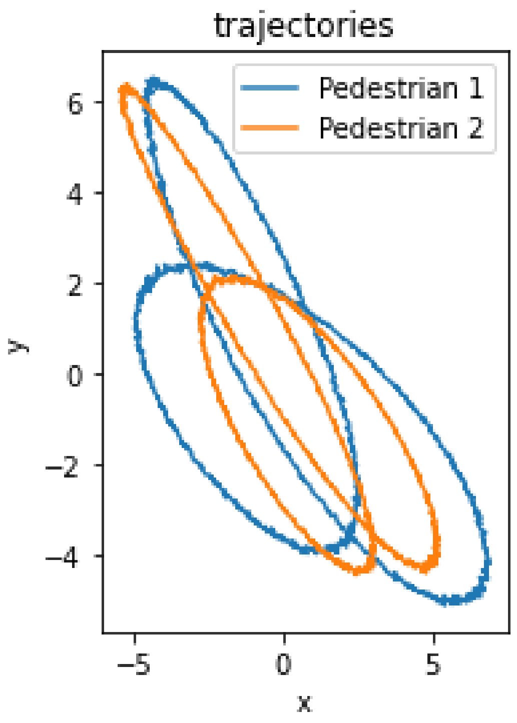

Figure 4: Reconstruction of the path of the first two pedestrians in 2D space.

that, the reconstructed image is roughly the same, but still lacks some details.

Part 3.3: Are two components enough to capture most of the energy of the data set? Why, or why not? By computing the cumulative_energy, we noticed that only 84.924% of the energy was captured. That is not enough to capture most of the energy of the data set. The L2 loss between original data and reconstructed data is about 1.24

Part 3.4: How many do you need to capture most of the energy? According to the computation, we need at least 3 components to capture most of the energy.

Answering the three questions from the first page of the exercise sheet
(a) an estimate on how long it took you to implement and test the method?
– About 5 hours.
(b) how accurate you could represent the data and what measure of accuracy you used?
–To compare original pictures and reconstructed pictures, we can use L2 loss to measure the accuracy. I used L2 loss and the loss value of task 3 is about 1.24.
(c)What you learned about both the dataset and the method?
–For task 1, the direction of first principle component is very similar to the distribution of data.
–For task 2, the information loss become visible when using 10 components but it is hard to notice that it already failed to capture most of the energy when using 50 of the components.
–For task 3, I noticed that the path of both of them are in a particular shape. Maybe they are walking along a certain path or they are having some events.

Report on task 2, Diffusion Maps

In this task, we were asked to implement the Diffusion Map[2, 1, 3, 5] only using the basic library. Following the 10 steps described in the exercise sheet, we implemented our method in task2_diffusion_maps/utils.py and all the steps can be found in the notebook Task2_diffusion_maps.ipynb.We used scipy.spatial.KDTree for a fast distance matrix implementation.

Part 1.1: Five eigenfunctions, plotted against t_k? In this part, we are going to demonstrate the similarity of Diffusion Maps and Fourier analysis. To do this, we computed the five eigenfunctions ϕ_ll= associated to the largest eigenvalues λ_l with Diffusion Maps, on a periodic data set with N = 1000 points. The data set is given by:

$$X = \{x_k \in \mathbb{R}^2\}_{k=1}^{N}, \tag{3}$$
$$x_k = (\cos(t_k), \sin(t_k)), \tag{4}$$
$$t_k = \frac{2\pi k}{N+1}. \tag{5}$$

The plot of the original data set is figure 5 and 6. The plot of the five eigenfunctions is in figure 7.

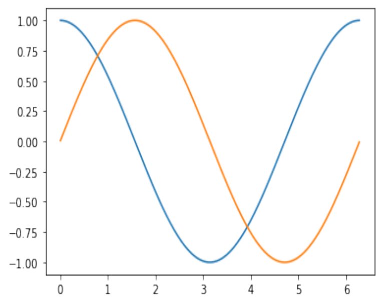

Figure 5: Fourier data set

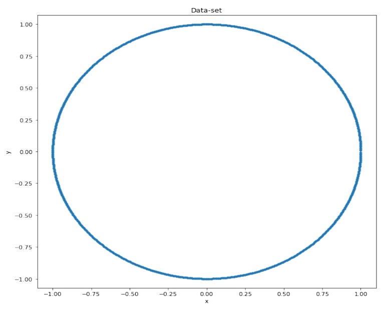

Figure 6: Plot x_1 against x_2

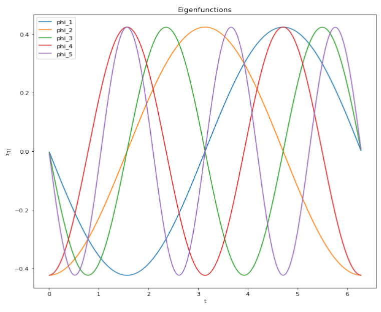

Figure 7: Five eigenfunctions

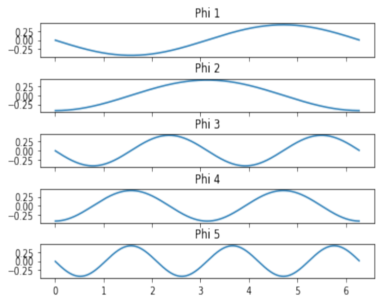

Figure 8: Separate figure of each eigenfunction

At the same time, we can check the separate figure of each eigenfunction in figure 8. From the separate figure we can notice that, ϕ_1 and ϕ_2 reconstruct the data and ϕ_3 and ϕ_4 contains 2 periods of the sin- and cos-function. ϕ_5 contains 3 periods.

Bonus: The relation of your results to Fourier analysis The dataset is a periodic signal on a unit circle. The eigenfunctions computed by the diffusion maps algorithm are similar to the sin- and cos- basis functions in Fourier analysis and these eigenfunctions reflect the components of the signal at different frequencies. So diffusion maps can be seen as a tool for capturing thevariation in data with periodic structures, which is similar to Fourier analysis.

Part 2.1: show a plot of the swiss roll data in 3D. In this part, we were required to use the algorithm to obtain the first ten eigenfunctions of the Laplace Beltrami operator on the“swiss roll” manifold. We used make_swiss_roll to obtain swiss roll from sklearn.dataset. The following figure is the swiss roll in 3D.

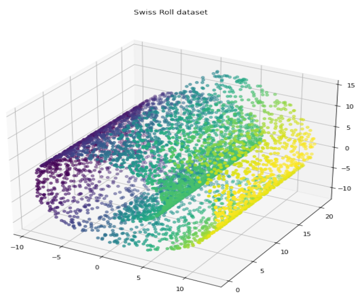

Figure 9: Swiss roll in 3D

Part 2.2: 10 eigenfunctions on swiss roll, plotted against ϕ_1. Except ϕ_1 plotted against itself will be a constant result, we plotted the figure of ϕ_1 against ϕ_2 to ϕ_{10}. From the figures 10 we can notice that when $l \leq 4$ the figures are similar to the degree 2, 3 and 4 polynomial functions. When applying the Diffusion map, the swiss roll were unrolled and the main features captured by the Diffusion map is very similar to the degree 2, 3 and 4 polynomial functions.

Part 2.3: Number l of eigenfunctions where ϕ_l is no longer a function of ϕ_1? From the plot we can notice that from l=5 the subplot of ϕ_5 the swiss roll is unrolled as a plane and ϕ_5 is no longer a function of ϕ_1

Part 2.4: PCA on swiss roll

Please notice that, for this part and all the parts in this assignment 3, we only implemented the TODO list in the .py documents. Other code for report and analyse can be found in the related jupyter notebook.

In this part, we used the utils.py from task1 pca and renamed it as pca.py. When N=5000, we can notice in the figure 11 that 3 components can perfectly reconstruct the swiss roll but 2 components is unable to do so.

Part 2.5: Why do you need three principal components for the swissroll, not two? The swiss roll is a 3-dimensional object and it is necessary to capture all the variance in 3 dimensions. So we need 3 principal components.

Part 2.6: What happens with 1000 data points, for Diffusion Maps and PCA? From the figure 12 we can see the swiss roll with 1000 data points.There are some gaps and holes due to the less of data points.

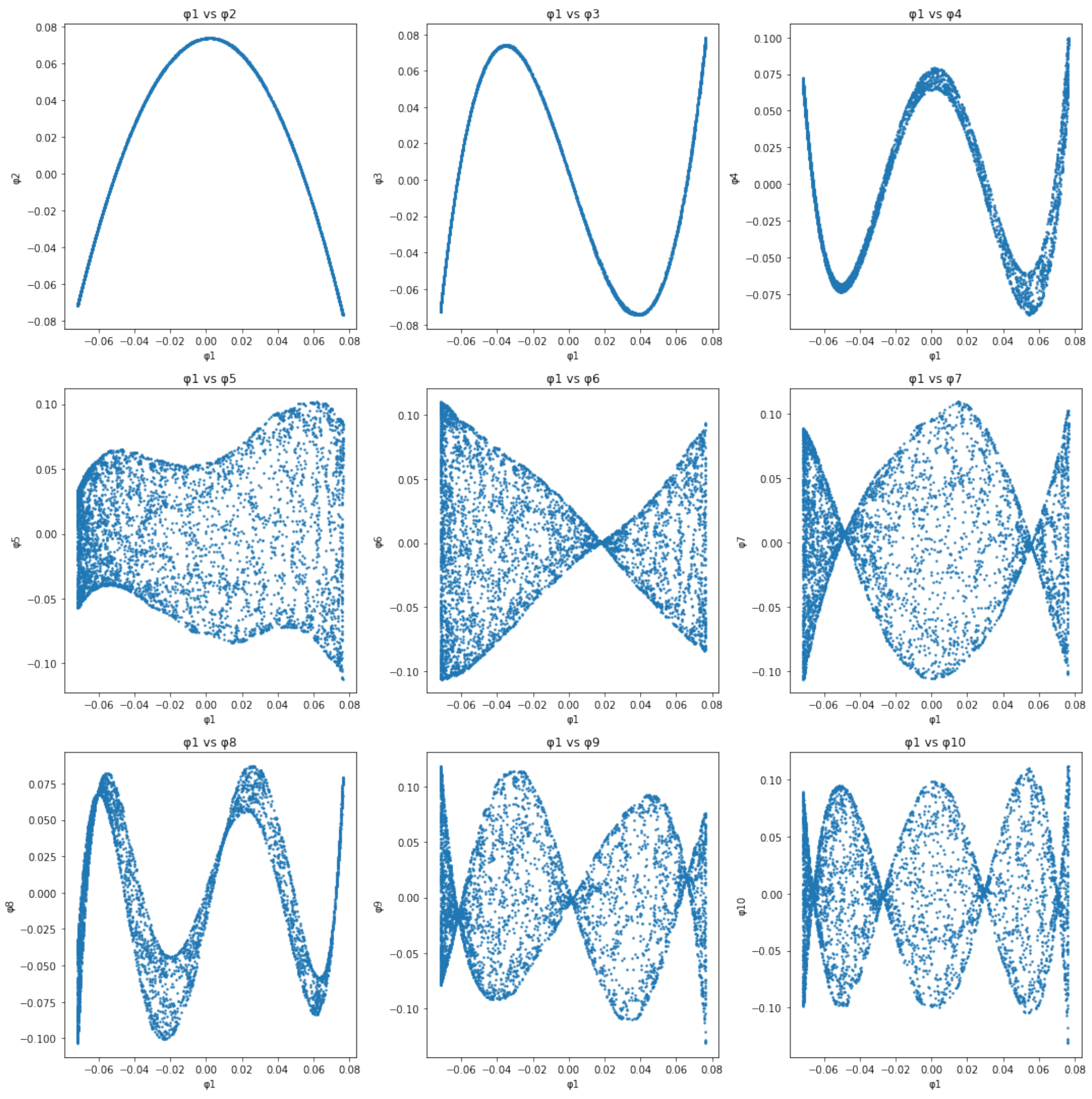

Figure 10: 10 eigenfunctions on swiss roll

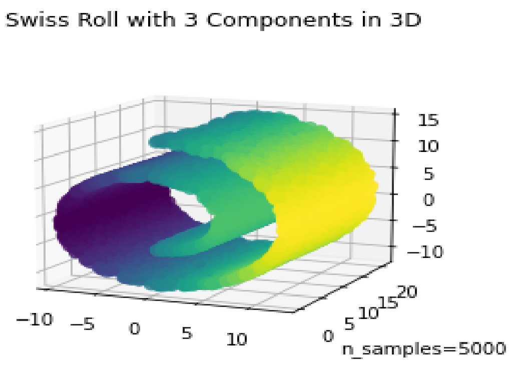

Figure 11: 3 and 2 Components to reconstruct swiss roll (N=5000)

First to repeat the experiments on Diffusion Map, we plotted the figures of ϕ_1 against ϕ_2 to ϕ_{10}. We can compare the figure 13 with the previous figures and notice that, with less sample points, the quality of diffusion

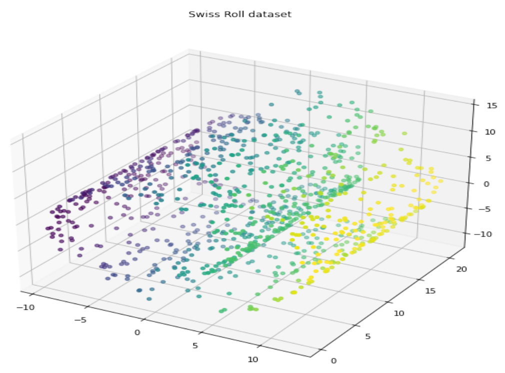

Figure 12: Swiss roll in 3D (N=1000)

map get worse. The unrolling which has shown in the previous ϕ_1 to ϕ_5 also can no longer be observed. Then

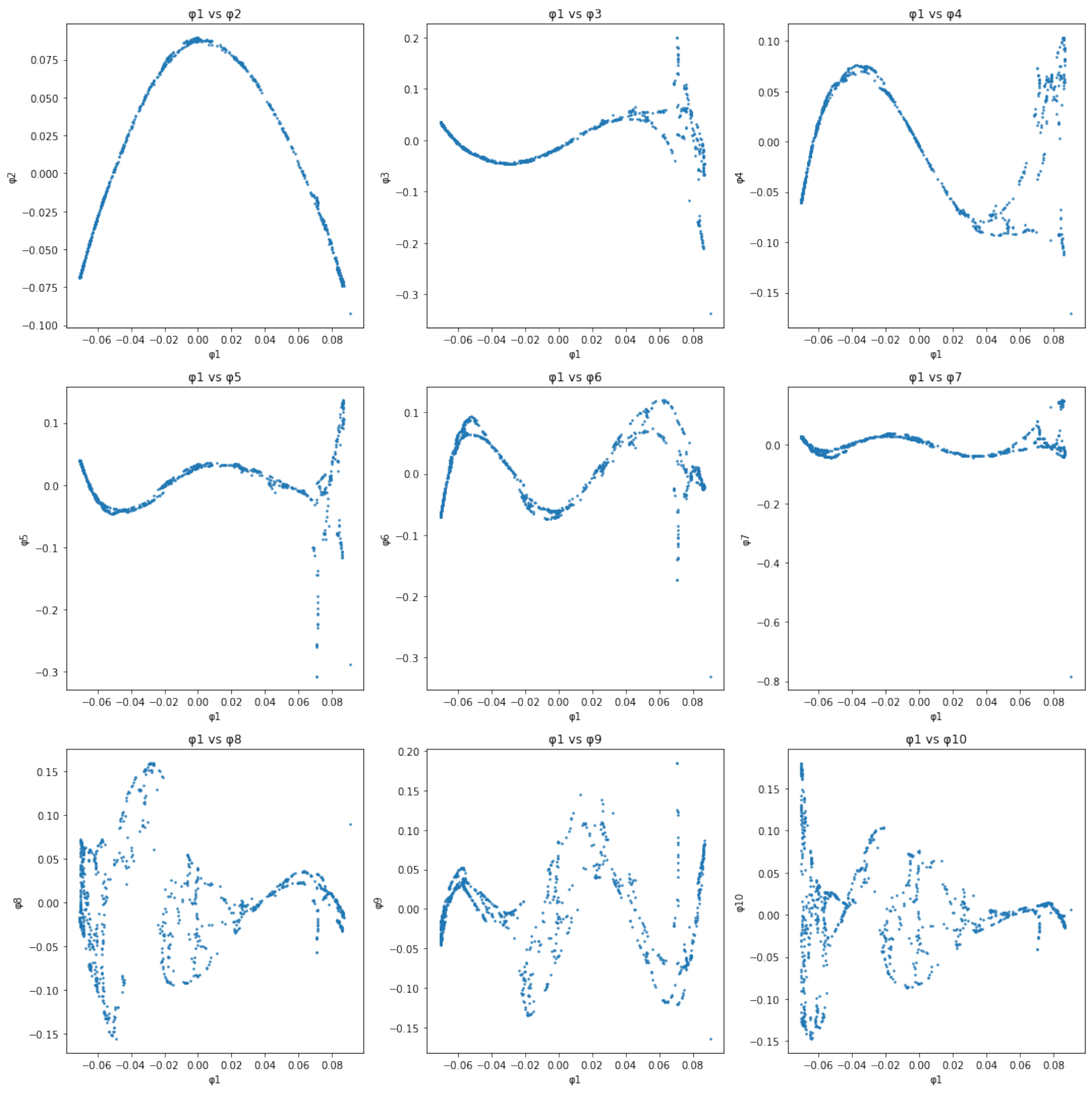

Figure 13: 10 eigenfunctions on swiss roll (N=1000)

we repeat the experiments on PCA. The quality is worse than the figure before and the data points are of high variance to the original image. With less data points, the confidence of positions of data points get lower.

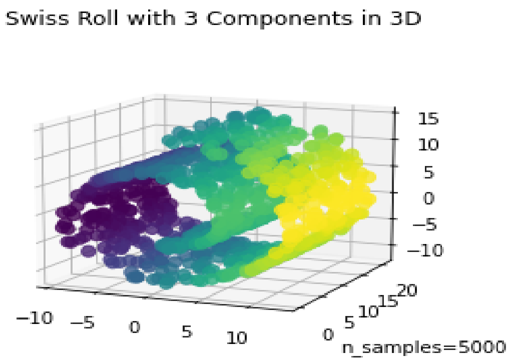

Figure 14: 3 and 2 Components to reconstruct swiss roll (N=1000)

Part 3: Same analysis as in PCA case for the file, but now with Diffusion Maps In this part, we performed the same experiments on the pedestrian trajectories. First we can rewiew the visualization of the first 2 pedestrians.

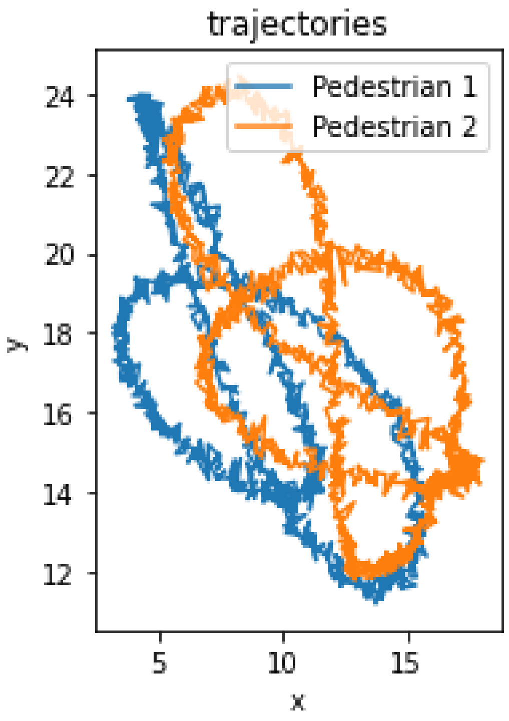

Figure 15: Visualize the path of the first two pedestrians in 2D space.

Then, we tried to plot the figures of ϕ_1 against ϕ_2 to ϕ_5. As shown in the figure 16, we can guess that the trajectories are circles. So the result of diffusion map should be similar. We can see from the figure ϕ_1 against ϕ_2 that, the first 2 eigenfuctions capture the circle or ellipse. So we can conclude that the first 2 eigenfunctions are sufficient to accurately represent the data set,

Answering the three questions from the first page of the exercise sheet

(a) an estimate on how long it took you to implement and test the method?

– About 8 hours.

(b) how accurate you could represent the data and what measure of accuracy you used?

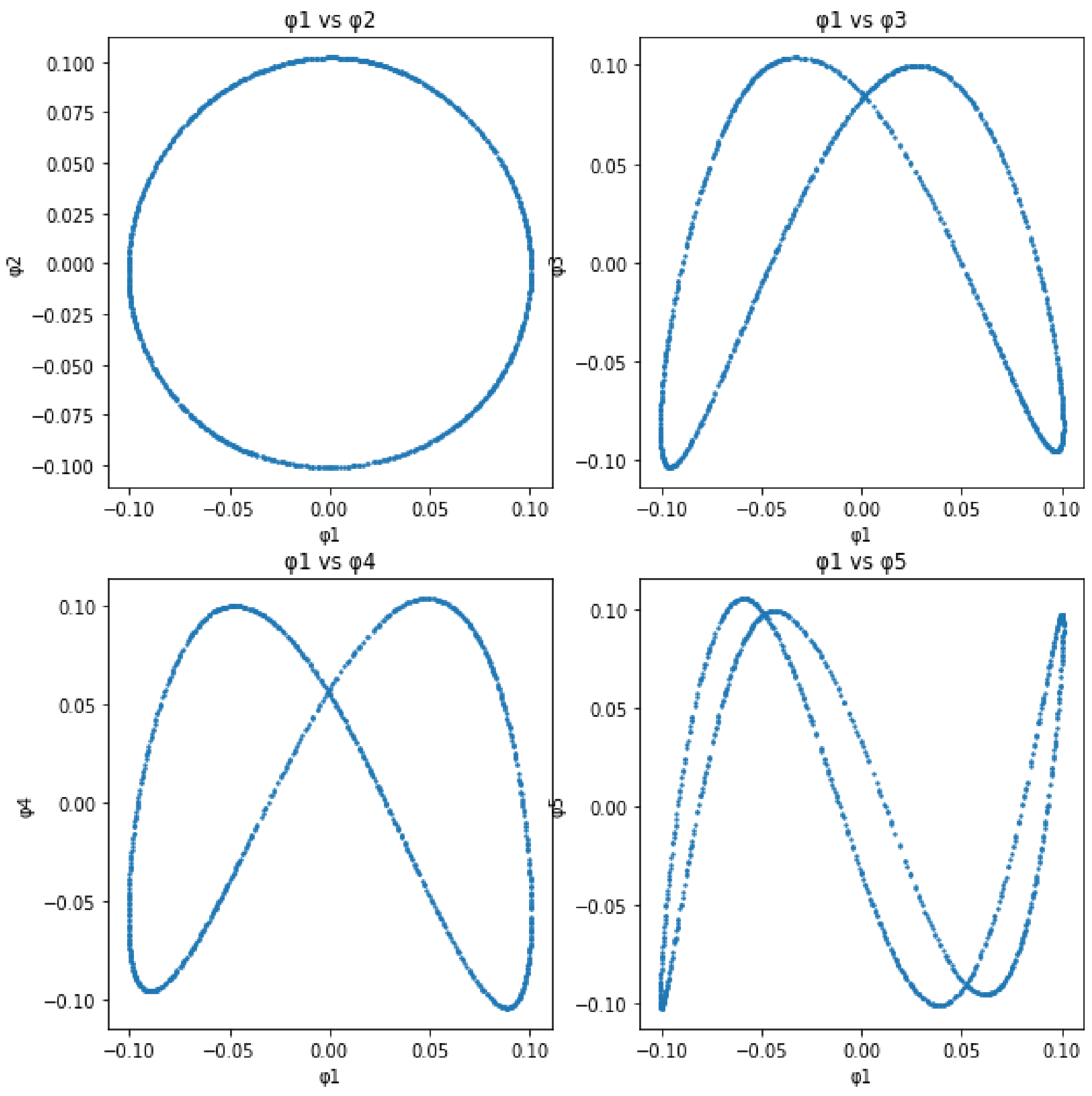

Figure 16: Diffusion map results

–To measure the accuracy, we can compare the diffusion distance against the euclidean distance of the diffusion map. From the following table we can see the MSE.

Acc	Fourier (ϕ_1, ϕ_2)	Swiss roll 5k (ϕ_1, ϕ_5)	Swiss roll (ϕ_1, ϕ_5)	Pedestrian (ϕ_1, ϕ_2)
MSE	0.290	6.409	5.307	13.697

Table 1:

(c)What you learned about both the dataset and the method?
–For task 3, As we noticed before, the path of both of them are in a particular shape. The diffusion map can easily discovering the underlying manifold[4].

Report on task 3, Training a Variational Autoencoder on MNIST

For this task we implemented our model in model.py and our training steps in utils.py. Also some code for report and analyse will be provided in the jupyter notebook. Following the instructions, we built a VAE model with an Encoder and a decoder, together with a linear layer for computing the mean and a linear layer for computing the variance.

Part 1: What activation functions should be used for the mean and standard deviation of the approximate posterior and the likelihood—and why? For the encoder, we used ReLU as activation function because ReLU can avoid problems like gradient vanish. We used ReLU together with Sigmoid as activation function for the decoder because the output should between 0 and 1.

Part 2: What might be the reason if we obtain good reconstructed but bad generated digits? To analyze this problem, we need first to check the loss function of the model,

$$\mathcal{L}_{\text{ELBO}}(\theta, \phi) = \mathbb{E}_{p_D(x)} \left[\mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z) \right] - KL(q_\phi(z|x)) || p(z)) \right] \tag{6}$$

The first term is relative to how similar the distribution of x is to the sample of approximate $q_\phi(z|x)$. This can help the VAE to learn how to generate a picture from the latent space accurately. The second term is about how close the distribution $q_\phi(z|x)$ is to the prior distrubution $p(z)$. A VAE with the second term applys a constriant to the latent distribution.

For a normal VAE model it is good at reconstructing, but have some problem for generating since the latent distribution are discontinuous. For our model, if we have a bad generating it means the KL divergence is very high, the approximate is very different from the prior. It happens when we have complex input and need to compress too many information into the latent space. The distribution of latent will be complex and thus VAE learn how to reconstruct but unable of working with randomly generated samples of latent vectors.

Part 3: Plot the latent representation, i.e., encode the test set and mark the different classes. From the figure 17 we can see the latent representation at epoch 0, 4, 24 and 49. We can see from the figures that, at fiest all of them distributed without special pattern. But because of the KL regulariser, the latent representation of different digits separated. Which will lead to a better reconstructions and digit generations.

Part 4: Plot 15 reconstructed digits and the corresponding original ones. We used our VAE model to reconstructed the digits. From figure 18 and 19 we can notice that, the result of reconstructions gets better with the training going on. However, there are still some mistaks in the last epoch. Our model cannot always distinguish between 4 and 9, and sometimes cannot distinguish 5 and 8 especially when the handwritting is not good.

Maybe we can find some clue for the poor performance of distinguishing certain number from the latent representation. According to figure 17, the encoding of number 0, 1 and 2 are the most separated and distributed from the rest. So it is possible that the model will not confused by these numbers. On the contrary, we can notice that the distribution of digit 4, 5, 8 and 9 are mixed together and overlapping. This poor latent representation means that the model will probably failed to distinguish these numbers, as we observed in figure 18 and 19.

Part 5: Plot 15 generated digits, i.e., decode 15 samples from the prior. From figure 20, we can check our result of generating digits. We can see that the results are blurry and some of them are even error digits. As we mentioned in **Part 2: What might be the reason if we obtain good reconstructed but bad generated digits?**, when we try to compress too much information into the latent space, the model may not able to capture enough information for a good generating. In this task, our latent dimension is 2, which is very small for the model to capture important information. In the following task we will try to increase the latent dimension to 32. We will compare the results later.

Part 6: Plot the loss curve (for the test set), i.e., epoch vs. ELBO loss. We plotted the loss curve in figure 21. From the loss we can notice that, the training loss and test loss both decrease with the training goes on. The final test loss is a little higher than training loss.

Part 7: 32-dimensional latent space: Compare 15 generated digits with the results in 3.2. For this part, we add our latent space dimension from 2 to 32. And then we perform the same step on the new model. The compare of the generate digits of 2 model are shown in figure 22. From the result we can see that, the digits are more clear than the 2-dimensonal latent space model, and there are less mistake than before. With a larger latent space, the model can capture more information and those information can help with the reconstruction.

Part 8: 32-dimensional latent space: Compare the loss curve with the one in 4. Figure 23 is the loss curve of the model with 32-dimensional latent space. Compare the former one, we can see that the test loss gets lower and gets closer to the training loss. Which means this model performs better on the test compared to the former one. This is also because of the increasing of latent space.

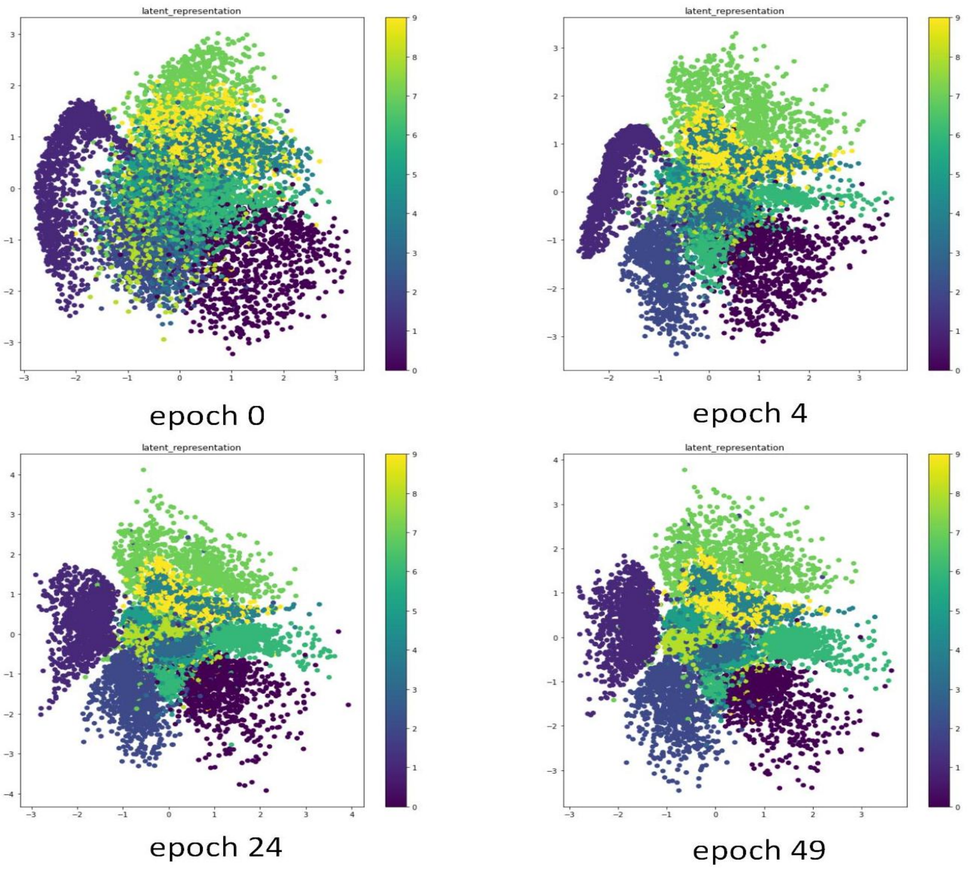

Figure 17: Latent representation

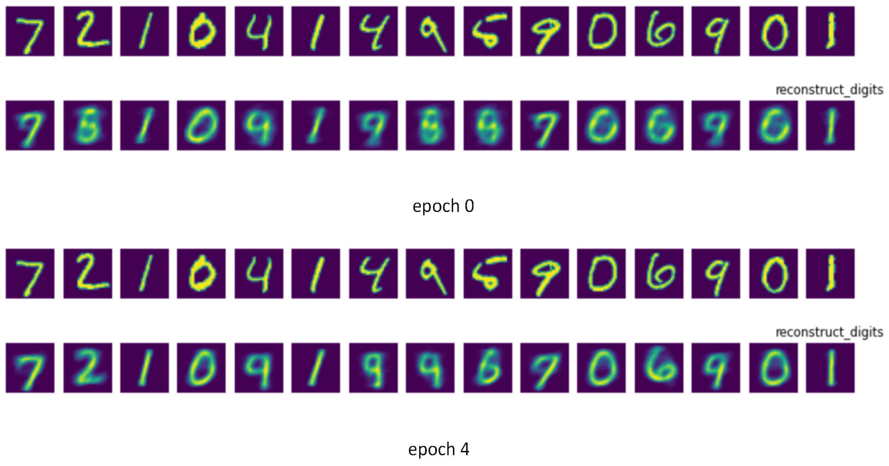

Figure 18: Reconstructed digits, epoch 0 and 4

Answering the three questions from the first page of the exercise sheet
(a) an estimate on how long it took you to implement and test the method?

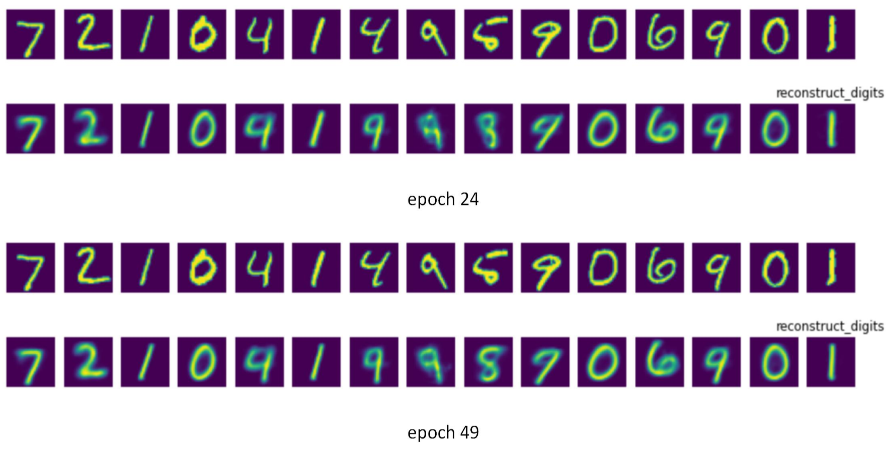

Figure 19: Reconstructed digits, epoch 24 and 49

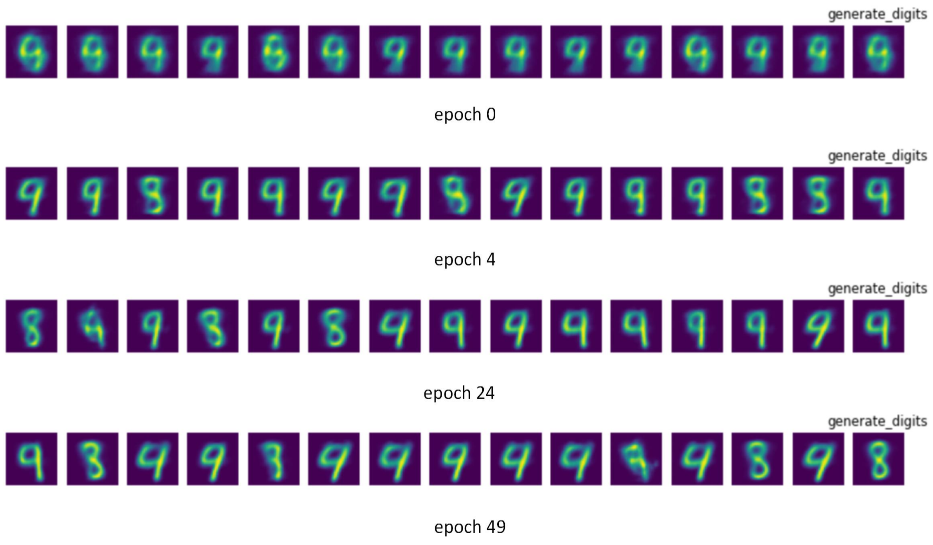

Figure 20: Reconstructed digits, epoch 0 and 4

– About 15 hours.
(b) how accurate you could represent the data and what measure of accuracy you used?
–To measure the accuracy, we used the ELBO loss, which is the sum of similarity and KL divergence.
(c)What you learned about both the dataset and the method?
–For the task, we can notice that, for hand writting digits, a lot of information are included so we cannot use a 2-dimensional latent space to deal with the digits. The information in the pictures are so much that they cannot be reduced to 2 dimensions.

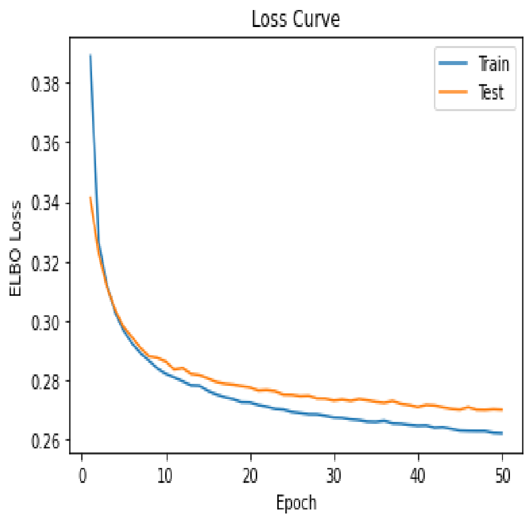

Figure 21: Loss

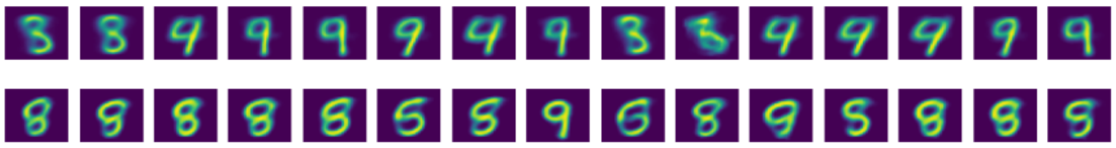

Figure 22: Generated digits of 32-dimensional latent space VAE model

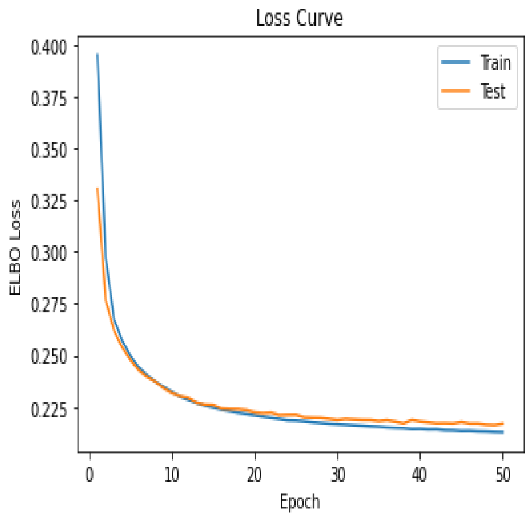

Figure 23: Lossl

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

With more and more student entering at the Technical University of Munich, the fire evacuation plan for the MI building needs to be reconsidered. In this task, we will Train a VAE on the FireEvac data and hope to find some useful information for the fire evacuation plan. The code can be found in the fire_evac and the jupyter notebook.

Download the FireEvac dataset scatter plot to visualise it. We visualise the training and test in the figure 24.

Train a VAE on the FireEvac data to learn p(x). Following the requirements in the Exercise sheet,

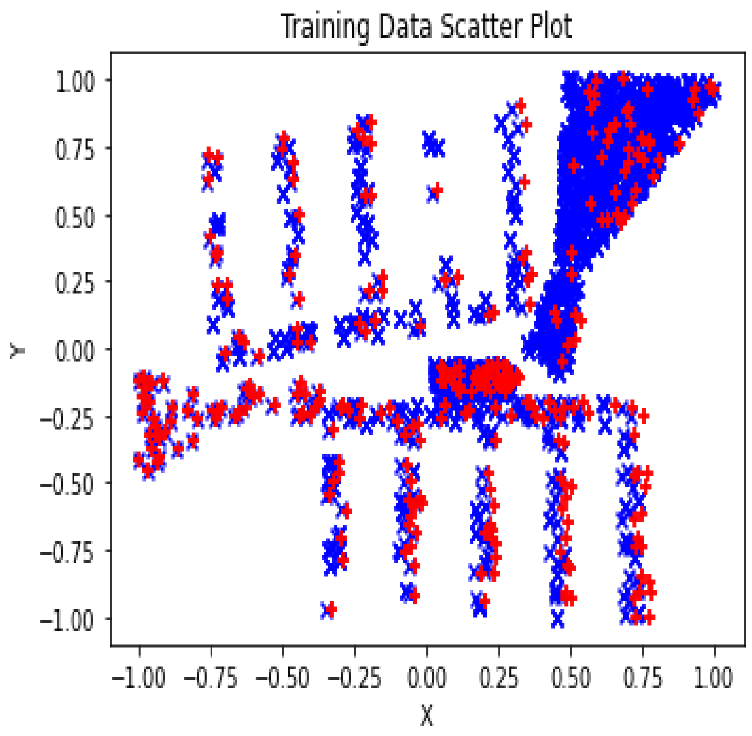

Figure 24: Visualise dataset

we use the VAE model from task 3. However, the input data in this task is in the range of [-1, 1], which includes negative part. To avoid dead ReLU, we changed our activation function from ReLU to LeakyReLU(0.2). To reconstruction the data properly, we replace the last activation function of the decoder to Tanh(), which cover the range from -1 to 1. But since we changed the model for the current task, the result in the jupyter notebook of former tasks are the model with ReLU, but not the one with LeakyReLU(0.2) and Tanh(). **(In order to pass the Artemis test, we uploaded the ReLU and Sigmpoid version. This may cause some problem when running this task. If you want to run the code for this task, please change the activation function to LeakyReLU(0.2) and Tanh()**

Also, the make the training steps and plot suitable for the dataset, we write a new training step and plot step.

By performing the former steps, we trained a model of FireEvac data. The training steps can be found in the jupyter notebook. During training, we found that if all the data are scaled to [-1, 1], the loss will also be very small. Small losses have little effect on gradient changes, and the function is prone to fall into the local optimum. The result of this is that all the reconstructed points stay together in a global munimum. We tried different way to solve this. We found that if we times the x and y value by 100, which can increase the loss and increase the penalty for mistakes, this can help the function avoid the local minimum.

Make a scatter plot of the reconstructed test set. We can check the reconstruction data in figure 25: We can see from the figure that, the position of reconstructed data are reasonable and some of them close to the test data.

Make a scatter plot of 1000 generated samples. We can check the generate data in figure 26: By using the former model, we generated the data. But the generated data are quiet different from the test data. A majority of generated data located at the top of the location, which is not even inside the building.

Generate data to estimate the critical number of people for the MI building. and **How many samples (people) are needed to exceed the critical number at the main entrance?** Since the generated data are really unreasonable, the estimate of critical number are also incorrect. According to our model, only when 500000 in the building the critical number will exceed 1000. Figure 27 is the plot of 10000 people.

Answering the three questions from the first page of the exercise sheet

(a) an estimate on how long it took you to implement and test the method?

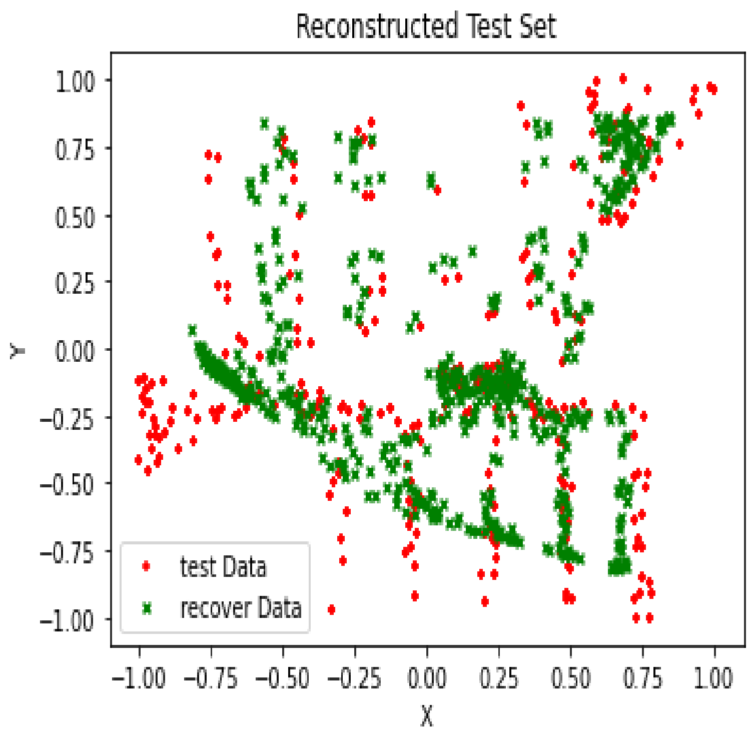

Figure 25: Reconstruction data

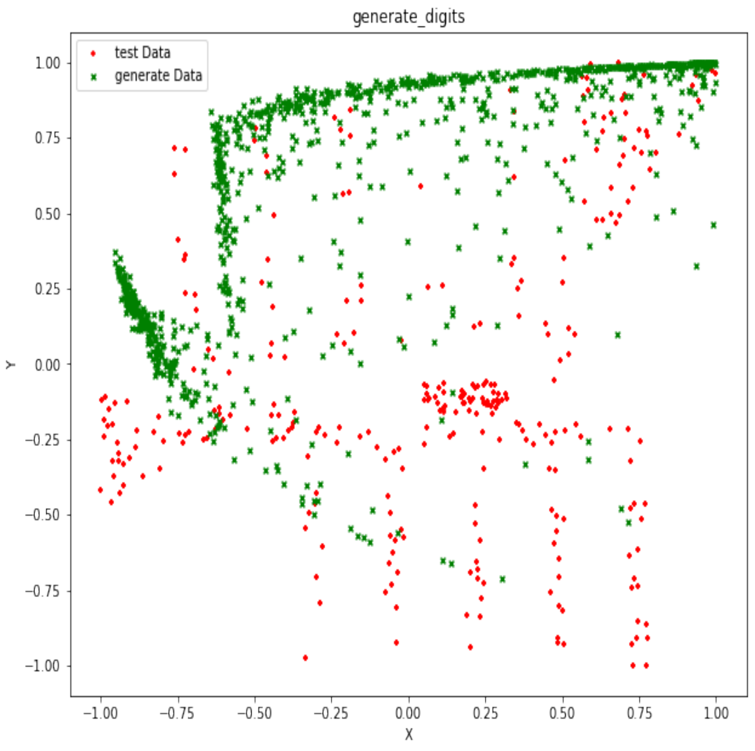

Figure 26: Generate data

– About 5 hours.
(b) how accurate you could represent the data and what measure of accuracy you used?

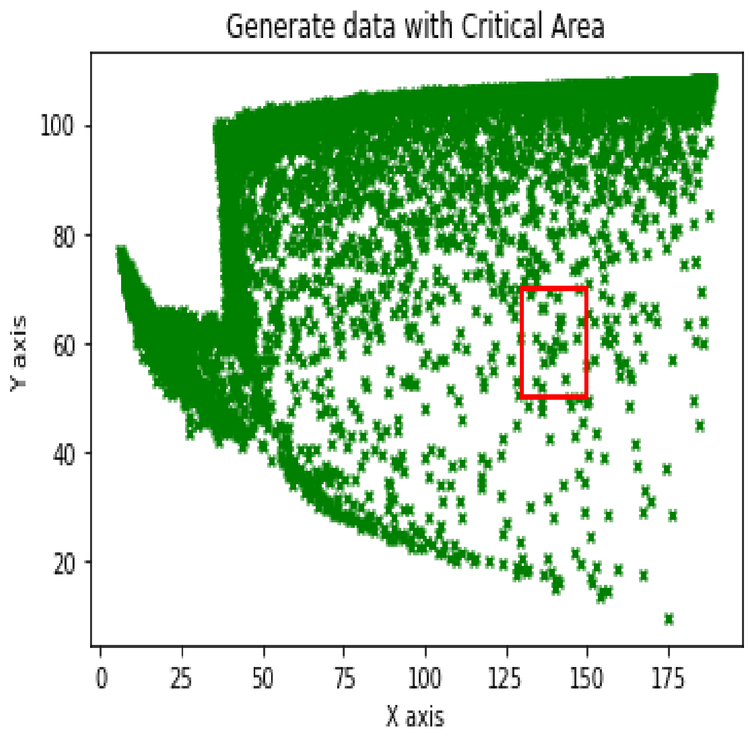

Figure 27: Generate data

–To measure the accuracy, we used the ELBO loss, which is the same as the previous task.
(c)What you learned about both the dataset and the method?
–For the task, we can notice that, some times the generate data are not reasonable, for example, some people outside the building. So use these data to calculate the critical number may not correct. The critical number is important for safety and it needs double checks from human.

References

[1] Ronald R. Coifman and Stéphane Lafon. Geometric harmonics: A novel tool for multiscale out-of-sample extension of empirical functions. *Applied and Computational Harmonic Analysis*, 21(1):31–52, July 2006.

[2] Ronald R. Coifman, Stcphane Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences of the United States of America*, 102(21):7426–7431, 2005.

[3] Stéphane S. Lafon. *Diffusion Maps and Geometric Harmonics.* PhD thesis, Yale University, 2004.

[4] John M. Lee. *Introduction to Smooth Manifolds.* Springer New York, 2012.

[5] Steven Rosenberg. *The Laplacian on a Riemannian Manifold.* Cambridge University Press, 1997.