Report for exercise 4 from group K

Tasks addressed:	4
Authors:	Oliver Beck (03685783)
	Junle Li (03748878)
	Chenqi Zhou (03734992)
Last compiled:	2023–01–07
Source code:	https://github.com/Linnore/MLCMS-EX4-GroupK

The work on tasks was divided in the following way:

Oliver Beck (03685783)	Task 1	33%
	Task 2	33%
	Task 3	33%
	Task 4	33%
Junle Li (03748878)	Task 1	33%
	Task 2	33%
	Task 3	33%
	Task 4	33%
Chenqi Zhou (03734992)	Task 1	33%
	Task 2	33%
	Task 3	33%
	Task 4	33%

Abstract

In this exercise, we learned how to represent high-dimensional data with lower-dimensional representations.

There is a large number of methods available, and we cannot cover all of them here. The algorithms we studied in this exercise are

1. Principal Component Analysis,
2. Diffusion Maps, and
3. Variational Auto-Encoders.

Python	3.8.8
Jupyter Notebook	6.3.0

Table 1: Software versions

The important note a) How long did it take to implement and test the methods: Task 1 took around 2 Working days of 2 people, so a total of around 16 hours. Task 2 took around 12 working hours, some processes worked more fluently. Task 3 took around 20 hours.

The accuracy of our implementation basically was limited by the functionality and implementation of libraries such as numpy. Obviously accuracy differs, with the different amounts of PCA which was part of the tasks.

We learnt that the data sets are complex and from a first sight it is hard to tell how much energy is contained within each dimension. It is especially hard to illustrate 3 dimensional data in a paper such as this one, since usual visualizations are limited to two dimensions. The methods used bring a totally new view towards our understanding of how Crowd Simulations might be achieved. By utilizing PCA multiple dimensions can be reduced to less complex data sets opening more options for more complex data.The methods used including the machine learning seem to bring a whole new level of various options and differ from what we have learnt so far.

Report on task 1, Principal component analysis

Task description

1.1 description In the first part of this task, you are supposed to implement principal component analysis using a library method for singular value decomposition. Then, approximate the one-dimensional, linear subspace of the twodimensional data set pca dataset.txt on Moodle that is optimal in the sense of variance reduction. How much energy is contained in each of the two components? Plot the data set as shown in the figure, and add the direction of the two principal components by drawingterse lines starting from the center of the data set.

1.1 solution The diagram 1 shows the reproduction of the figure 3 in the exercise sheet as desired. Most of the values are located around a diagonal line with basically no exceptions. The first component contains around 40% of the energy, the second component around 32% and the third compinent contains around 27% of the energy. In figure 2 we added the drawingterse lines starting from the center of the data set . These line represent the first two components and their directions. For better visibility they were scaled by the square root. The first component runs along the diagonal, along the "length" in which the points are arranged, the second component in cooperates the points in the "width" of the line.

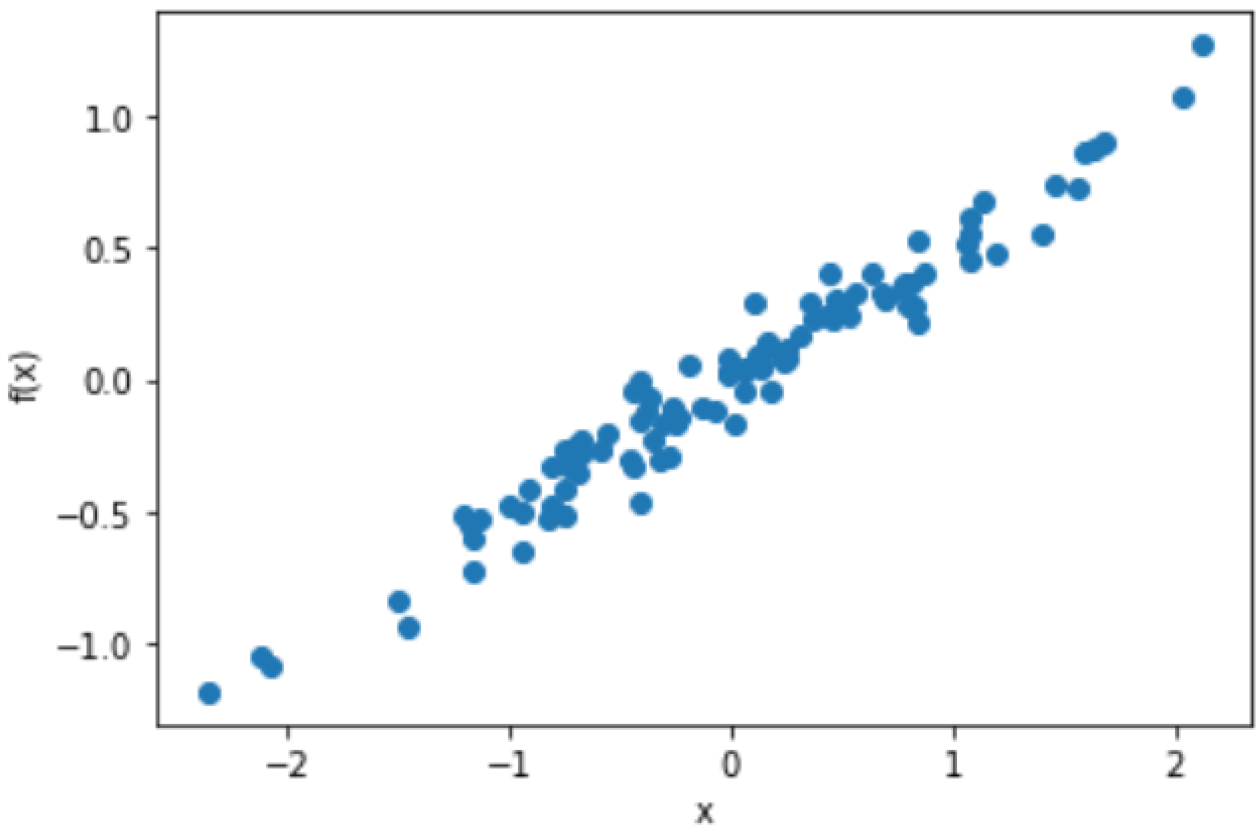


Figure 1: Reproduced Figure 3 in the exercise sheet.

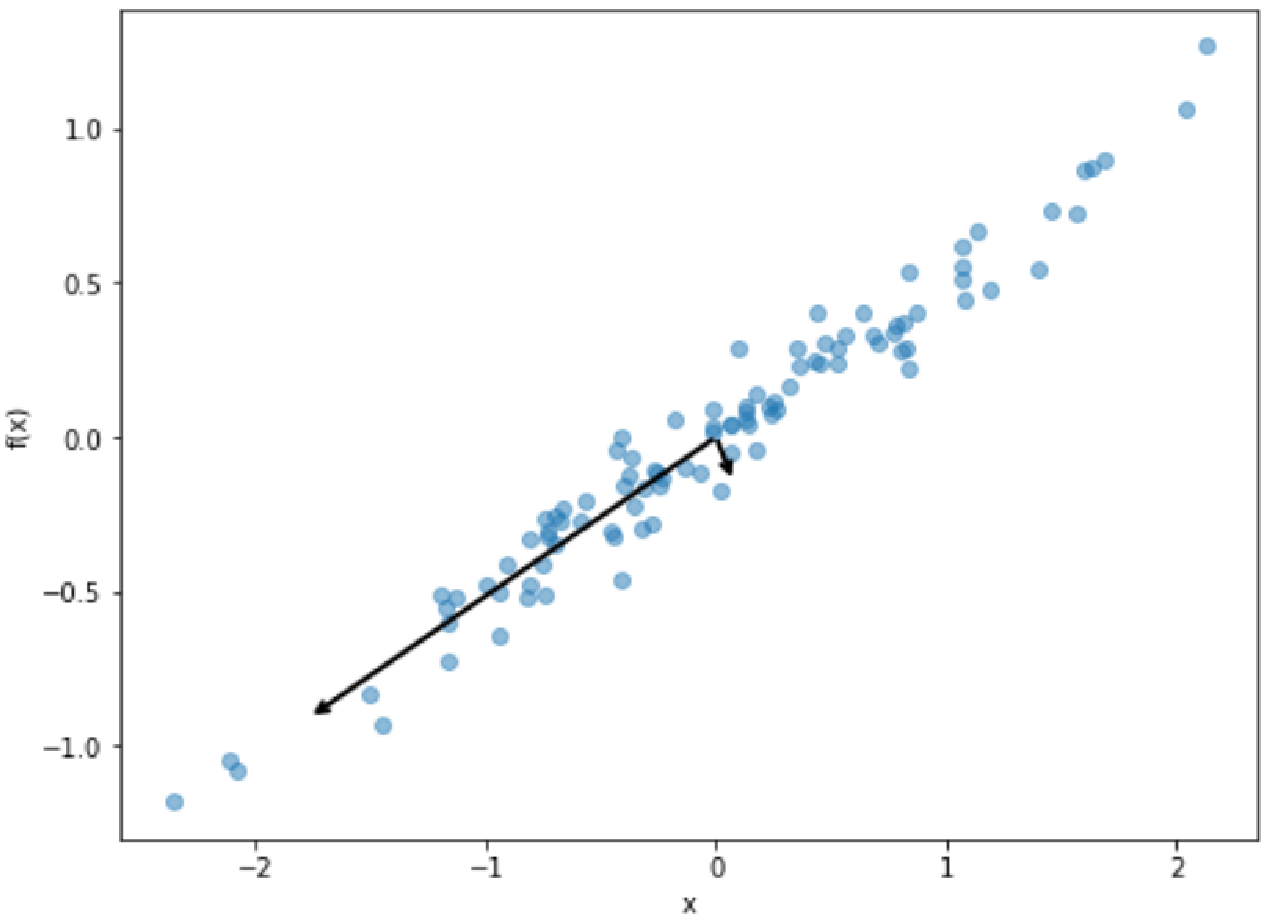


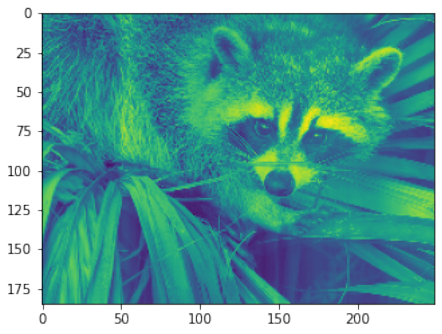
Figure 2: Linear subspace of the two-dimensional data set.

1.2 description For the second part, apply PCA to the image below. You can download the image on its original website, or generate it through the scipy.misc.face() command in Python. Note that this image is the spiritual successor to "Lena", which was removed due to copyright issues. The columns of the image should be considered as the data points in the analysis. You have to convert the image to gray-scale before starting the analysis, e.g. by using scipy.misc.face(gray=True). You also should rescale the image to have size (249 x 185).

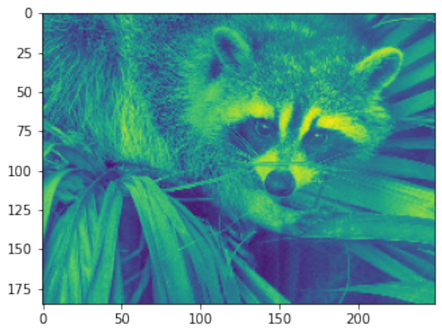
Visualize reconstructions of the image with (a) all, (b) 120, (c) 50 and (d) 10 principal components. At what number is the information loss visible? To obtain a reconstruction with a certain number L of principal components, just set the singular values to zero that are smaller than the L-th singular value and then reconstruct using $\widehat{X} = U\widehat{S}V^T$. At what number is the "energy" lost through truncation smaller than 1% ?

1.2 solution The information loss from the PCA with 120 components 3(b) is only visible, when directly comparing it with the original photo 3(a) and focus on the brightest and darkest spots of the photo. For example the contrast isn't as extreme as in the original photo. This is more noticable when comparing the 50 PCA 3(c) with the original photo. Not only are the contrasts less developed, but also the single hairs of the misc are harder to divide from each other and don't seem as fine grained anymore. Still the quality loss isn't noticeable immediately. This changes when looking at the picture with 10 PCA 3(d). This picture seems blurry, very unclear, and overall unpleasant to look at due to the bad quality. Many details are lost. But overall a misc still can be identified and main features are still visible.

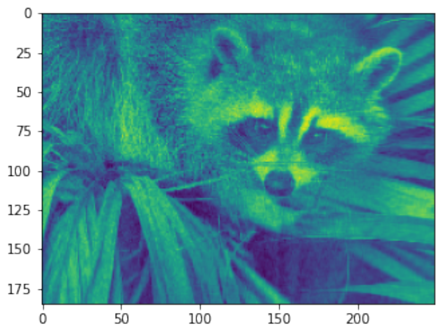
For L $\geq$ than 120 the energy loss through truncation is smaller than 1%. This fits to our observed results, that the pictures of 3(a) and 3(b), if at all, can only be differentiated by minimal differences.



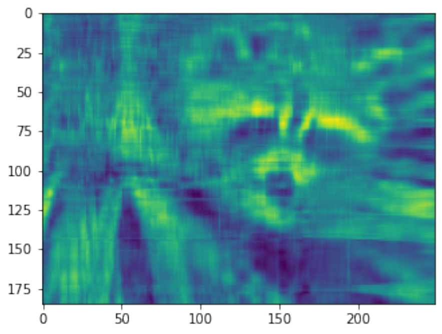
(a) all



(b) 120



(c) 50



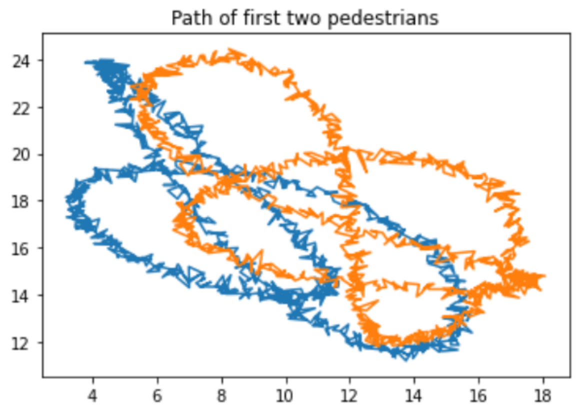
(d) 10

Figure 3: Reconstruction of the system with different Principal Components

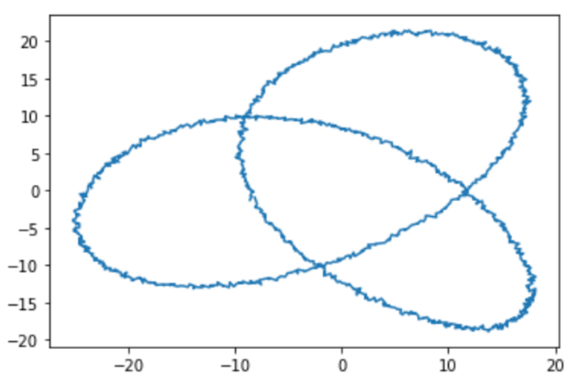
1.3 description The third and last part of this task concerns the file data_DMAP_PCA_vadere.txt containing the trajectory data on Moodle. The file contains position data of 15 pedestrians over 1000 time steps (in the form $x1, y1, x2, y2, ...$ for the x, y positions of all pedestrians).

1. Visualize the path of the first two pedestrians in the two-dimensional space. What do you observe?
2. Analyze the data set by projecting the 30-dimensional data points (one in each row of the file) to the first two principal components. Discuss your findings, in particular: are two components enough to capture most of the energy (¿ 90%) of the data set? Why, or why not? How many do you need to capture most of the energy?

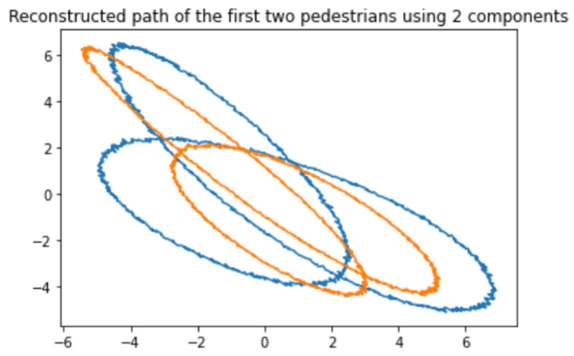
1.3 solution What we observe is that the trajectories of pedestrians have similar shape, similar y starting coordinates, but quite different x starting coordinates 4(a). In 4(b) shows a good perspective on the underlying form of the graphs. What we can tell is that the orientation of the graphs gets lost , when we only use two components. This is due to the nature of these plots being 3 dimensional, but the two components only capturing energy of 2 dimensions. The numbers we observe for the respective energy for the representations with 1 component is 40%, 2 components is 72% and 3 components is 100%. When comparing the values of the respective x coordinates and the respective y coordinates of the two diagrams, it is apparent that the x values differ majorly compaired to the y components. This means that the x components are the reason for the observedd differences,



(a) Visualizing the path of the first two pedestrians



(b) first two components as basis



(c) reconstructing the dataset with two components

Figure 4: Reconstruction of the system with different Principal Components

Report on task 2, Diffusion Maps

Task description I expect you to implement the Diffusion Map algorithm yourself, in the language of your choice, using only basic library support. You do not need to implement eigensolvers, matrix multiplication routines, etc., but you are also not supposed to use existing algorithms for Diffusion Maps (except for the bonus points with datafold).

2.1 description Part one: Use the algorithm to demonstrate the similarity of Diffusion Maps and Fourier analysis. To do this, compute five eigenfunctions ϕ_l associated to the largest eigenvalues ϕ_l with Diffusion Maps, on a periodic data set with N = 1000 points given by

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

. Plot the values of the eigenfunctions $\phi_l(x_k)$ against t_k.

2.1 solution The five eigenfunctions associated with the largest eigenvalues are plotted in **??**.

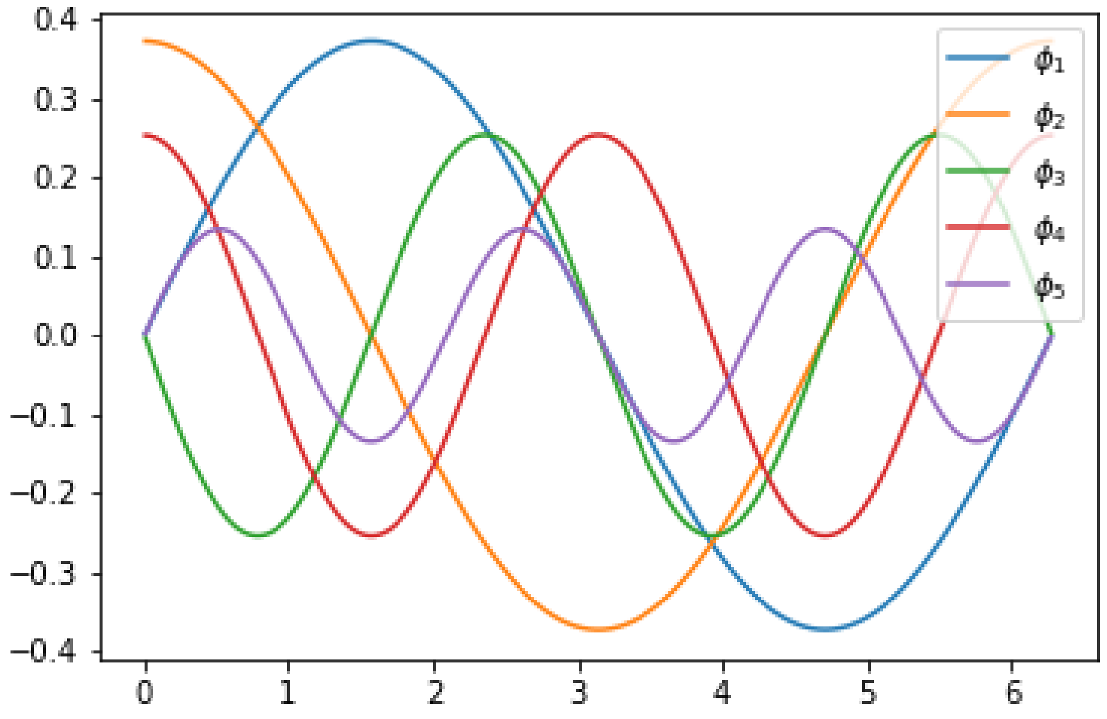


Figure 5: five eigenfunctions associated to the largest eigenvalues with Diffusion Maps

2.1 Bonus task description Briefly explain the relation of your results to Fourier analysis of the "signal" X (two or three sentences).

2.1 Bonus solution In the Fourier analysis the main spectrum channels of the diagram are split into it's main components. The decomposition into the main eigenfunctions basically does the same thing. Therefore the frequencies of the above plots should correspond to the strongest 5 frequencies identified in the Fourier Analysis.

2.2 description Use the algorithm to obtain the first ten eigenfunctions of the Laplace Beltrami operator on the "swiss roll" manifold, defined through

$$X = \left\{X_k \in \mathbb{R}^3\right\}_{k=1}^{N}, x_k = (u\cos(u), v, u\sin(u)),$$

where $(u, v) \in [0, 10]^2$ are chosen uniformly at random. The sklearn Python library has a simple routine to generate the swiss-roll data set6. Use this method to create 5000 data points in three-dimensional space, with no additional noise. Use the Diffusion Map algorithm to obtain approximations of the eigenfunctions, and plot the first non-constant eigenfunction ϕ_1 against the other eigenfunctions in 2D plots (the function ϕ_1 on the horizontal axis). At what value of l is $\phi_l, l > 1$ no longer a function of ϕ_1? Compute the three principal components of the swiss-roll dataset. Why is it impossible to only use two principal components to represent the data? What happens if only 1000 data points are used?

2.2 solution At the value $\phi_l, l > 4$ no longer is a function of ϕ_1. 7(a) and 7(b) mostly have values near the "main" line of the graph, e.g. for each ϕ_1, there is only a small range of ϕ_2 values. At 7(c) the values are in a small corridor around the "main" plot, which is larger than in the previous two plots. In 7(d), 7(e) , 7(f) , 7(h) and 7(i) we can see big areas of the graph covered in points. If you imagine a straw and twist it, one more time for every plot that is how one could describe the results visually. The only exception from this is 7(g), which looks fairly similar to 6(c).

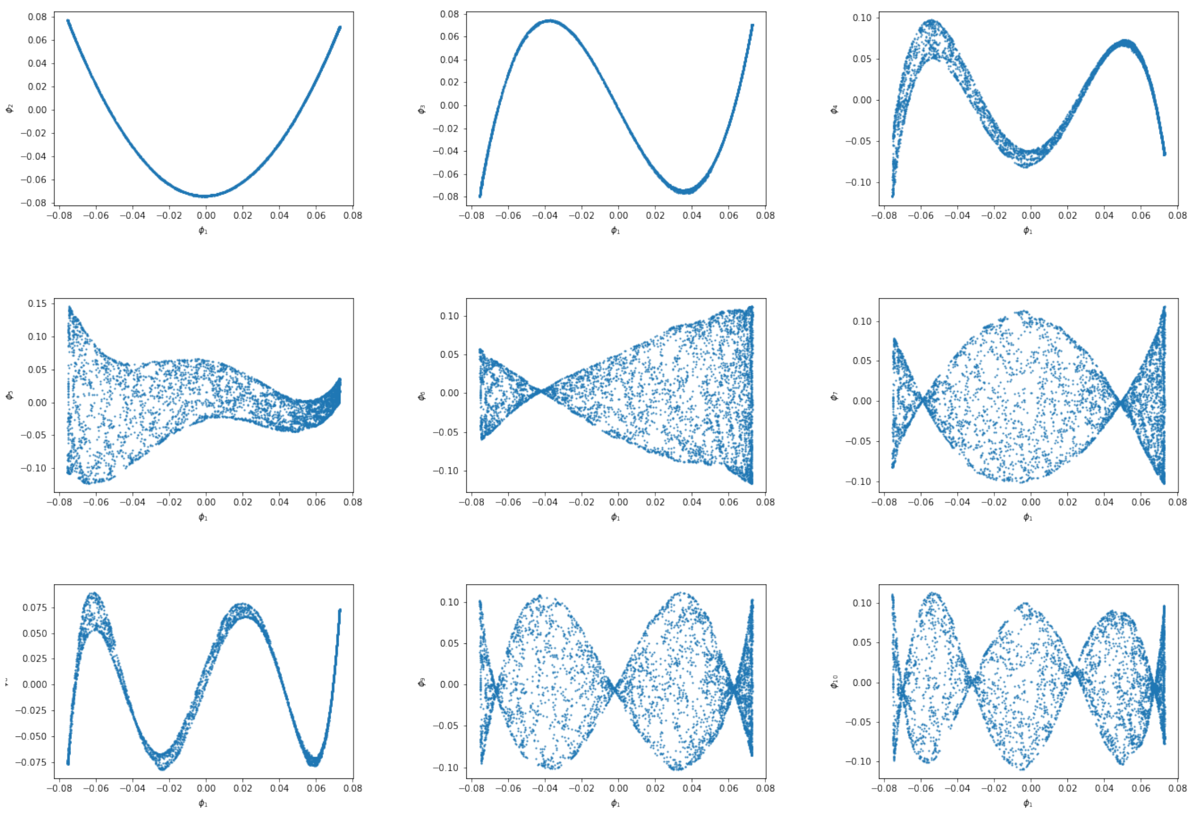


Figure 6: 5000 datapoints

Compared to the images with 5000 components 6 the images with **??** obviously have a lot less points and in comparison therefore look sparse. There are vague similarities between some of the graphs, based on the very basic layout (e.g. points form curves in low dimensionality and within a corridor, there are now random points away from this corridor. The graphs **??** and **??** etc. don't seem to directly, visually correspond.

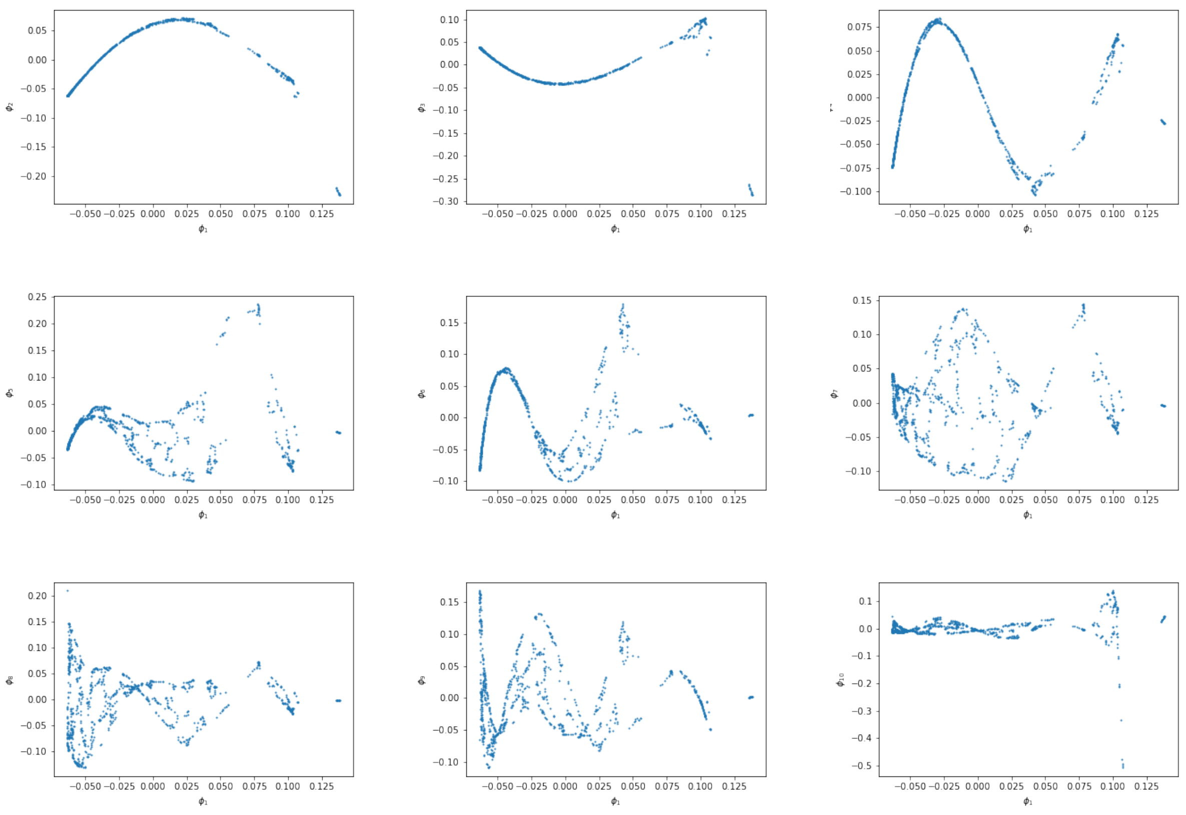


Figure 7: 1000 datapoints

As can be seen the 8(a) and 8(b) that only have 2 components are flat and not 3 dimensional. If you compare this with the 3 components of 8(c) and 8(d) it becomes clear that 2 components can't possibly recreate the nature nature of the "swiss role". Therefore in our case atleast 3 components are needed.

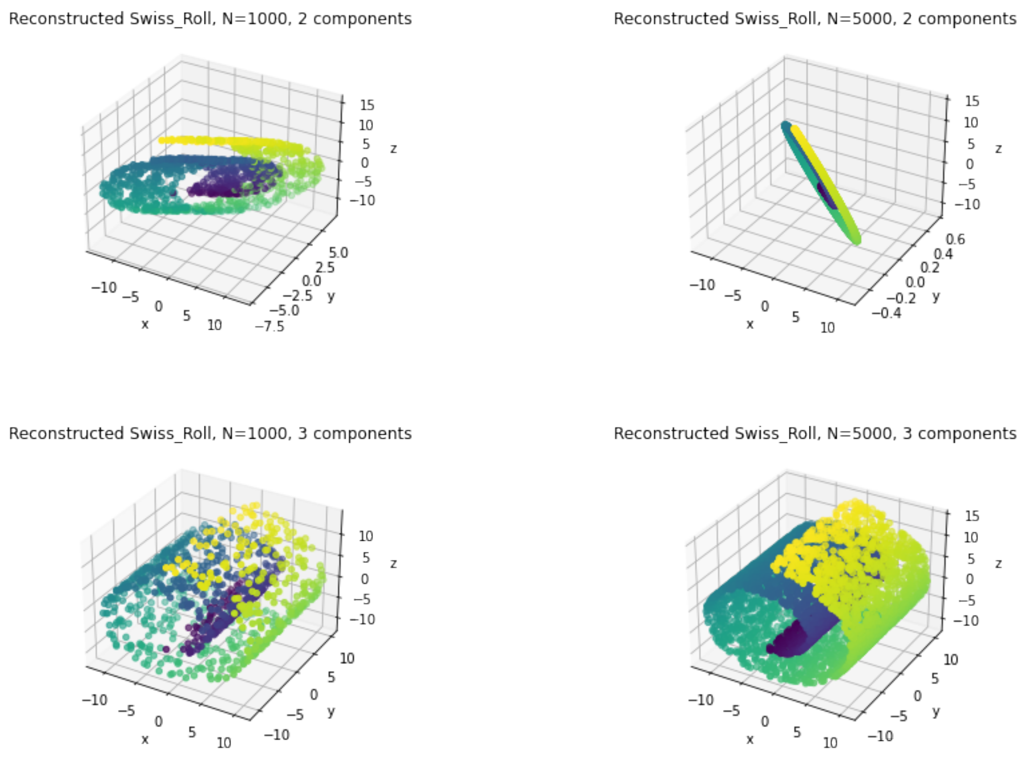


Figure 8: Reconstructed Swiss Roles with different configurations

---

Report on task 3, Training a Variational Autoencoder on MNIST

Task description In this exercise, we are supposed to implement and to train a VAE.

Dataset: Use the MNIST dataset. Note: do not binarise MNIST but normalise between 0 and 1, and split it into a training and a test set.

Model: Use a two-dimensional latent space. Use a multivariate diagonal Gaussian distribution as approximate posterior $q_\phi(z|x)$. The encoder neural network, which outputs the mean and standard deviation of $q_\phi(z|x)$, should consist of 2 hidden layers (dense/fully connected) with 256 units each and ReLU activation functions. Use also a multivariate diagonal Gaussian distribution as likelihood $p_0(x|z)$. The decoder neural network, which outputs only the mean of $p_0(x|z)$, should consist of 2 hidden layers (dense/fully connected) with 256 units each and ReLU activation functions. Implement the standard deviation as a trainable global variable. Use a multivariate diagonal standard normal distribution as prior p(z). Note: using a Bernoulli likelihood, as it is the case in most tutorials, is only possible for binarised versions of MNIST.

Optimisation: Use the Adam optimiser with a learning rate of 0.001. Use a batch size of 128 for training. Note: when training VAEs, there are—in addition to the loss—typically three sources of information, which indicate whether the model trains properly: the latent representation, reconstructed data, and generated data.

3.1 description What activation functions should be used for the mean and standard deviation of the approximate posterior and the likelihood—and why?

3.1 solution In order to apply to the binarised versions of MNIST, a Bernoulli likelihood is used. In the MNIST dataset we used, the pixel values have been normalized to the interval [0, 1]. For the last linear layer of the decoder, we use the Sigmoid activation function to make the output concentrated between [0, 1], similar to the input data.

Therefore, for binary data, we can activate the decoder with the Sigmoid activation function.

3.2 description What might be the reason if we obtain good reconstructed but bad generated digits?

3.2 solution Very good performance on the training set, but poor performance on the validation set is probably due to overfitting.

Overfitting usually means that the model learned by training has poor performance on the test set and good performance on the training set, that is, overfitting on the training set and has poor generalization (generalization ability). Especially when the training set is very small, the pattern learned during training may be very different from the target pattern. Generally, a model that is too complex will show good performance in the training set (even 0 error) but perform poorly in the test set.

Overly complex models usually lead to overfitting, and regularization and increasing the training dataset can alleviate overfitting.

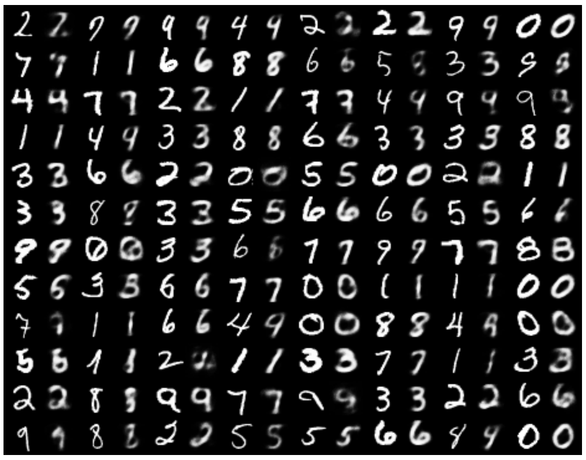
3.3 description Train the VAE (training set) and do the following experiments (test set) after the 1st epoch, the 5th epoch, the 25th epoch, the 50th epoch, and after the optimisation converged:

(a) Plot the latent representation, i.e., encode the test set and mark the different classes.

(b) Plot 15 reconstructed digits and the corresponding original ones.

(c) Plot 15 generated digits, i.e., decode 15 samples from the prior.

3.3 solution We train the VAE (training set). And plot 15 reconstructed digits and decode 15 samples from the prior after the 1st epoch, after the 5th epoch.


(a) reconstructed digits epoch1

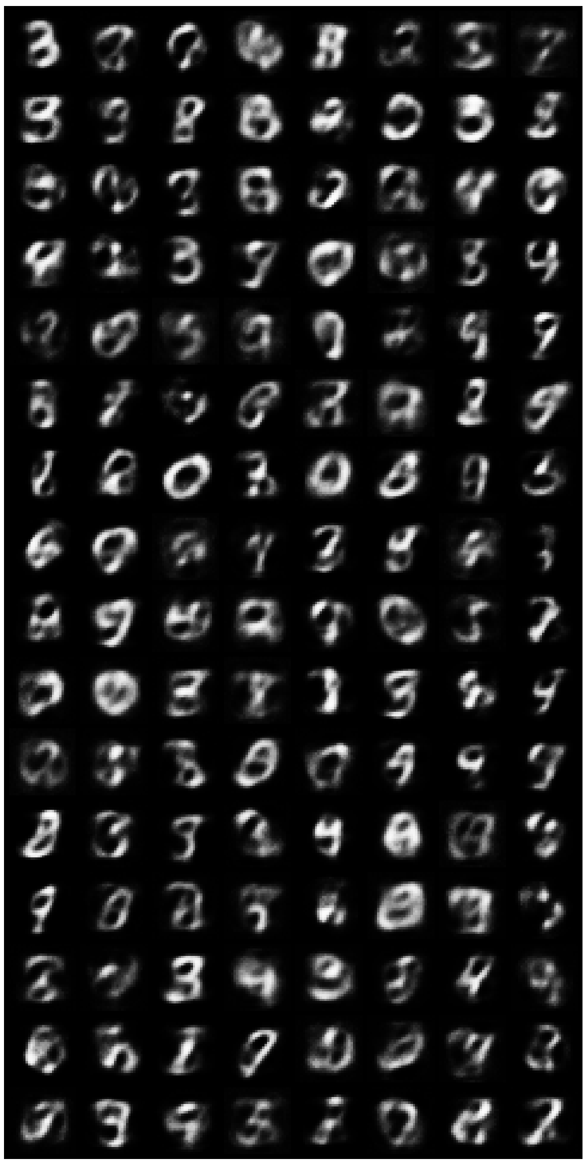
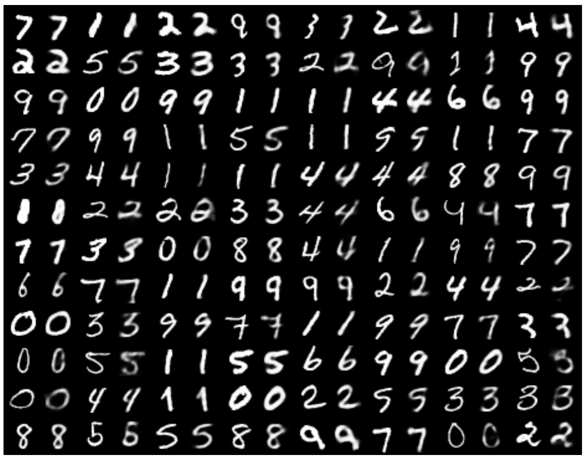

(b) generated digits epoch1

Figure 9: After the 1st epoch


(a) reconstructed digits epoch5

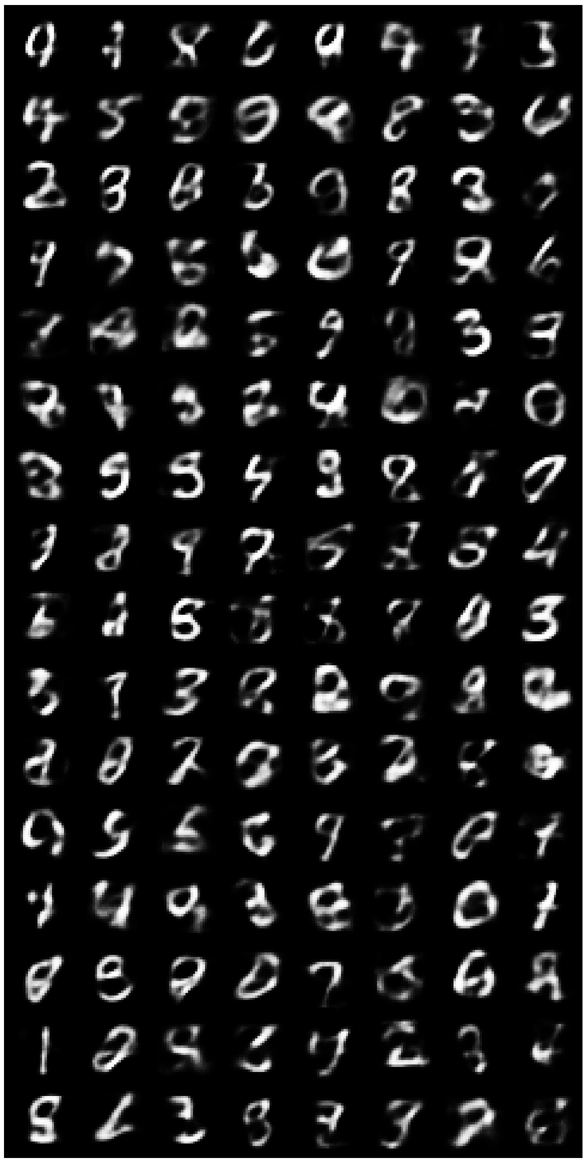

(b) generated digits epoch5

Figure 10: After the 5th epoch

3.4 description Plot the loss curve (test set), i.e., epoch vs. $-L_{ELBO}$
Here we define the reconstruction loss function (binary cross entropy) and relative entropy (KL Divergence Penalty). KL divergence measures the difference between two probability distributions.

3.5 description Train the VAE using a 32-dimensional latent space and do the following experiments after the optimisation converged:
(a) Compare 15 generated digits with the results in 3.2
(b) Compare the loss curve with the one in 4.

Summary We did not use the tensorflow suggested in the tutorial, but used PyTorch. We have not all completed task3.
In order to measure the accuracy of the data, we define the reconstruction loss function and KL DivergencePenalty. KL divergence measures the difference between two probability distributions.
The MNIST dataset is a database of handwritten digits. Each piece of data is a one-dimensional matrix with a length of 784, which is because the data has been standardized. Each piece of data is 784 pixels of a

handwritten picture. In the MNIST dataset we used, the pixel values have been normalized to the interval [0, 1].

In fact, VAE constructs an exclusive normal distribution for each sample, and then samples to reconstruct it. As far as I know, VAE is now widely used to generate images.

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

Task description Due to the increasing amount of students at the Technical University of Munich, the fire evacuation plan for the MI building needs to be reconsidered. An important information is the distribution of people within the MI building $p(x)$.

In a hypothetical scenario, the fire department decided to track 100 random students and employees during the busiest hour on different days. The idea is to use this data for learning $p(x)$. As a first experiment, the fire department wants to estimate the number of people that is critical for the building. To simplify the task, it defined a sensitive area in front of the main entrance—marked by the orange rectangle (130/70 & 150/50)-where the number of people should not exceed 100.

4.1 description Download the FireEvac dataset (training: 3000 x-y-positions; testing: 600 x-y-positions) and make a scatter plot to visualise it.

4.1 solution We download the FireEvac dataset and make a scatter plot to visualise it.

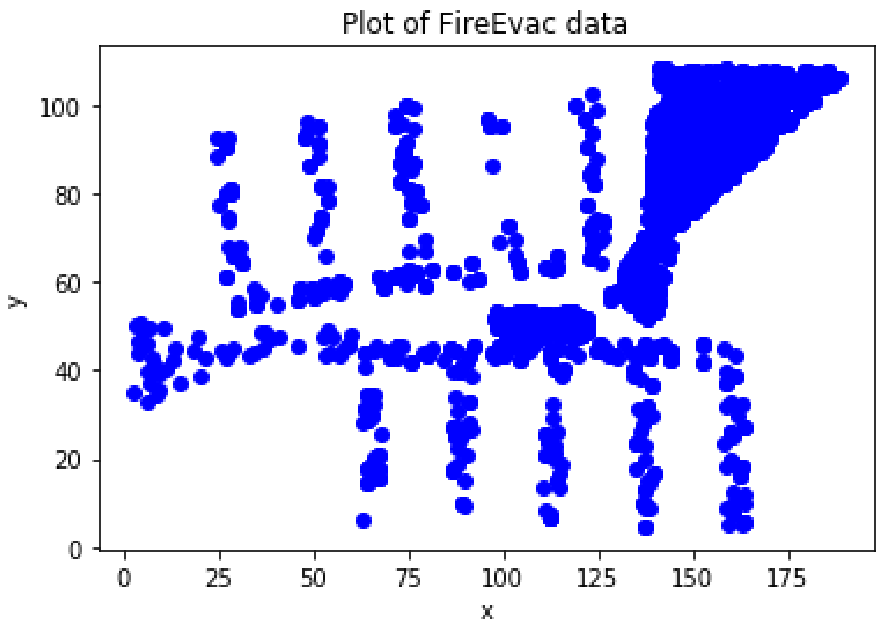


Figure 11: Visualise the FireEvac dataset

4.2 description Train a VAE on the FireEvac data to learn $p(x)$. You can reuse your VAE implementation from the previous task. Note: the observation space for this dataset is two-dimensional (instead 784 in MNIST). Although it is a low-dimensional dataset, it is not a simple one. You may need to train the VAE for quite a few epochs, or to tune hyperparameters. Another good idea is to rescale the input data x to a range of $[-1, 1]$ before you train the model.

4.3 description Make a scatter plot of the reconstructed test set.

4.4 description Make a scatter plot of 1000 generated samples.

4.5 description Generate data to estimate the critical number of people for the MI building: how many samples (people) are approximately needed to exceed the critical number at the main entrance?

4.6 Bonus Task description Later the fire department wants to use p(x) as the initial distribution for a precise crowd simulation.

4.6.1 description 1. Create a new Vadere scenario for the MI building with the appropriate dimensions, some of the walls (similar to the figure) and a target in the top right corner.

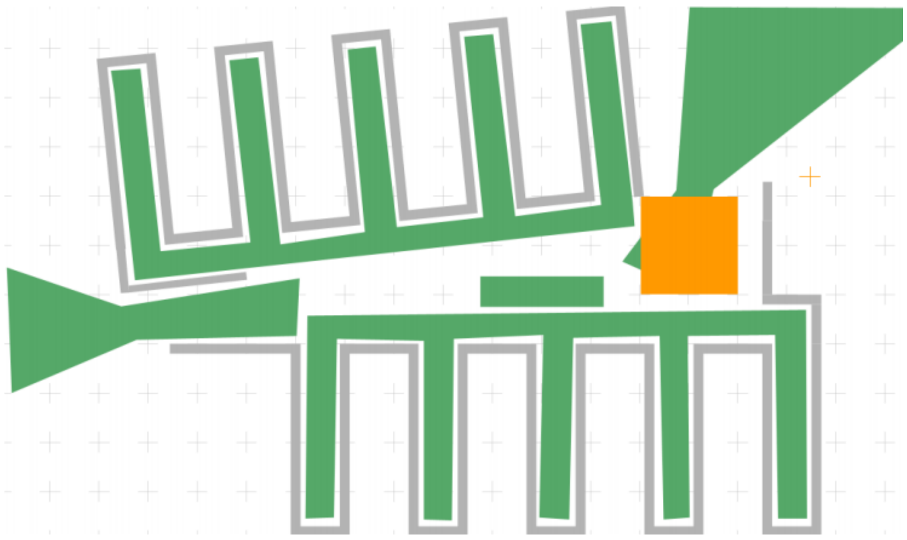


Figure 12: Schematic of the MI building in Garching, including the areas where pedestrians can be located (green) and are measured (orange).

4.6.2 description 2. Use $p(x)$ to sample the positions of 100 people and simulate their trajectories with Vadere as they move toward the target.

4.6.3 description 3. Report what you did and discuss the results.