Report for exercise 3 from group C

Tasks addressed:	4
Authors:	Daniel MAYAU (03724644) Amélie Chloé SCIBERRAS (03725378) Yalvac TOP (03720348)
Last compiled:	2019–12–11
Source code:	https://github.com/Cyberlilie/EX1_practikum_crowd_modelling

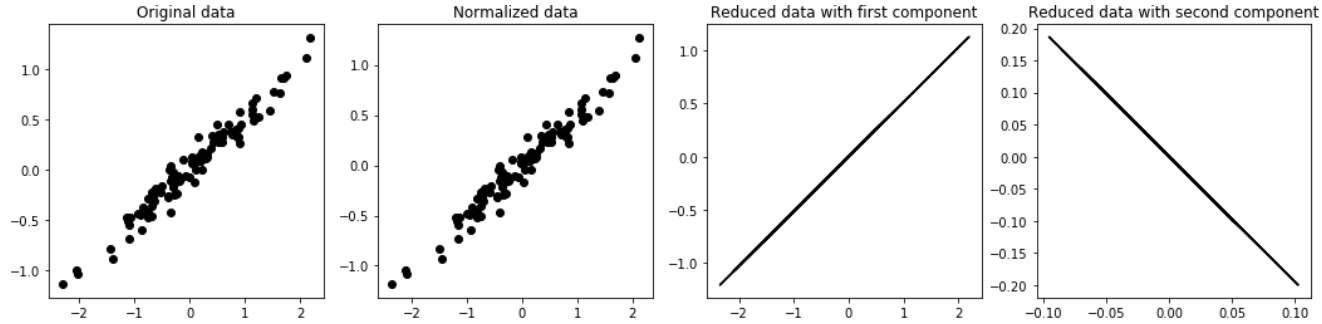
The work on tasks was divided in the following way:

Daniel MAYAU (03724644)	Task 1	34%
	Task 2	34%
	Task 3	33%
	Task 4	33%
Amélie Chloé SCIBERRAS (03725378)	Task 1	33%
	Task 2	33%
	Task 3	33%
	Task 4	33%
Yalvac TOP (03720348)	Task 1	33%
	Task 2	33%
	Task 3	34%
	Task 4	34%

The computing parts are coded in Python, they can be found in the folder "Exercise 3" of the following repository : https://github.com/Cyberlilie/EX1_practikum_crowd_modelling

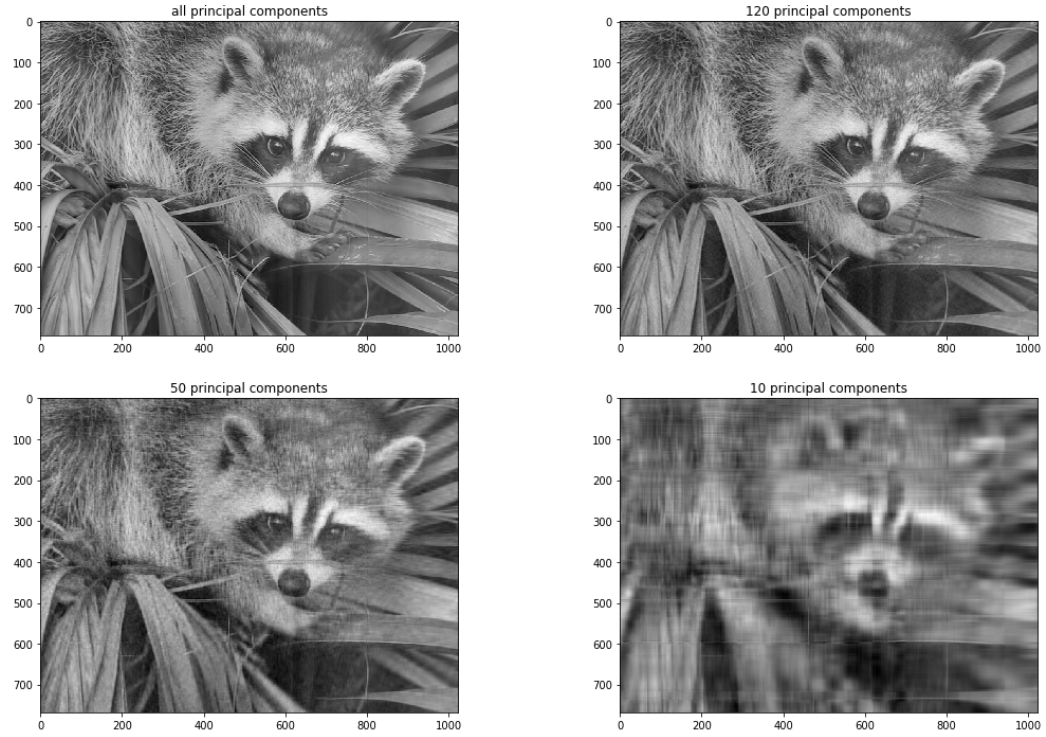
Report on task 1/5 Principal component analysis

1. After normalization, we apply the PCA on the data. The results were observed as follow:



The value of the energy for L = 1 is 92,3 %, which means that the energy contained in the second component is 7,7 %. We can observe this on the graph: The first component fits the data pretty well, because here the distribution of the data seems indeed to follow a line. The second component on the contrary deviates totally. In fact the information in this component has no value without the first one, it is additional to describe the data, adding the part that could be considered as noise.

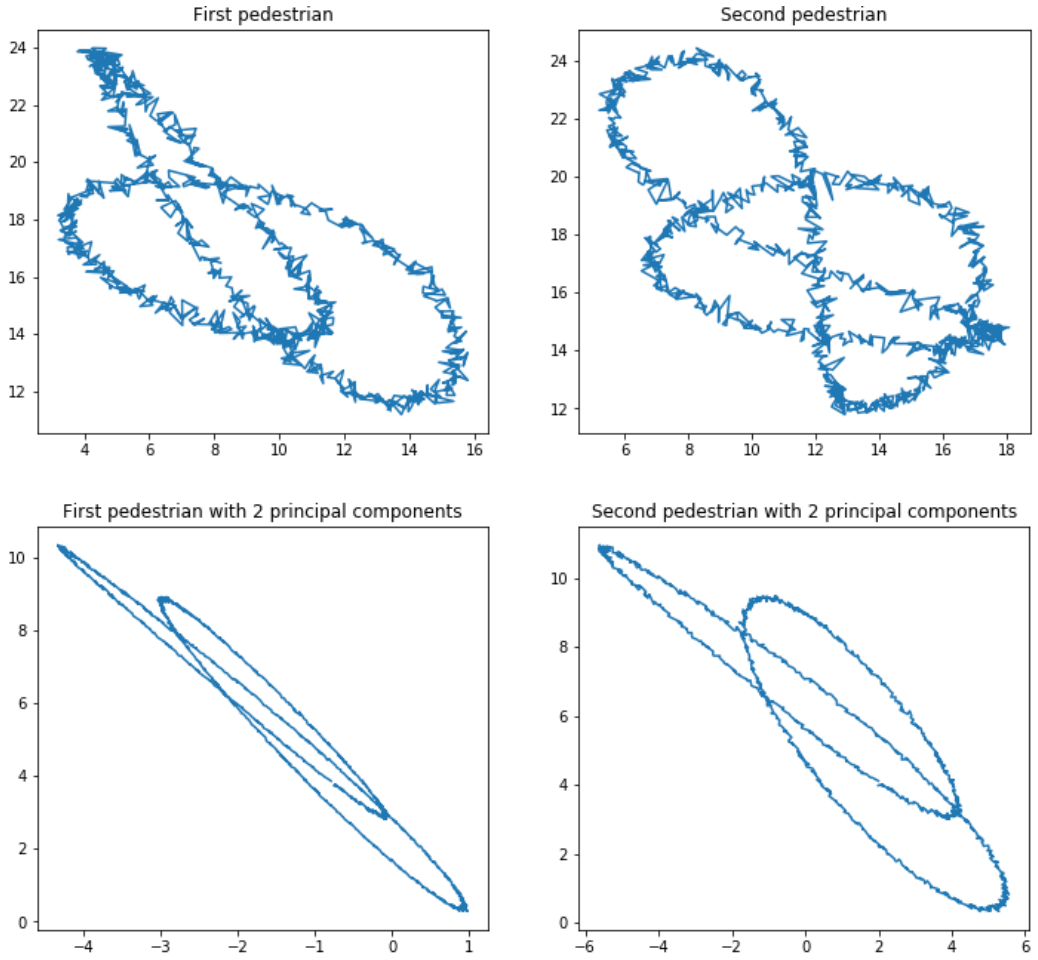
2. In this part, it is the rows which were considered as data, the columns were considered as the space dimension. This was simply easier, no need to transpose the matrix.



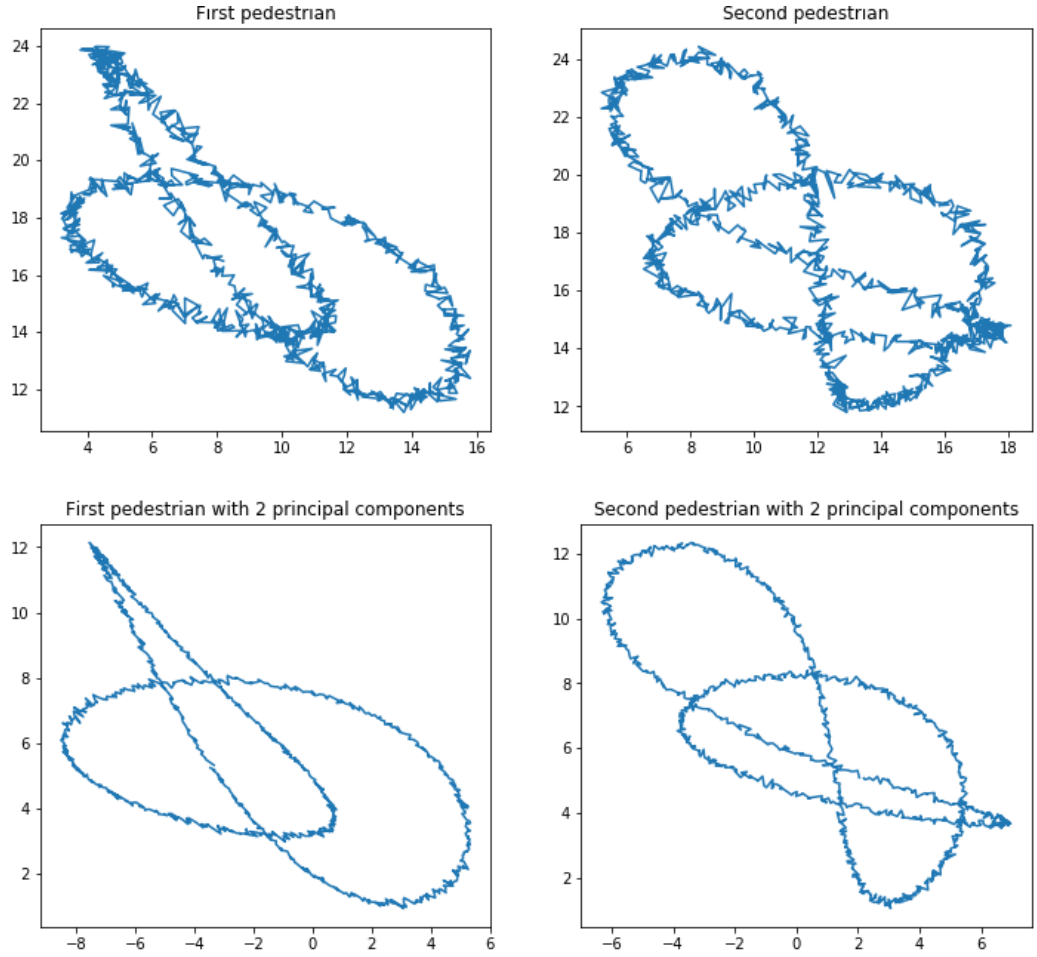
We can see here that 120 components are, to the human eye, enough to describe the picture. With 50 components, the picture starts to blur a little but is still of a correct quality, which is not the case anymore for 10 components. With a loop it was possible to get that to have a loss smaller than 1 %, at least 483 components

are required.

3. The data shows for both first pedestrian a clear trajectory running as a loop (it is not possible to determine a starting point and an ending point). Nevertheless, the trajectory is very noisy, as the pedestrians move around the trajectory by an indirect path (which makes sense for a pedestrian). To study the trajectory though, the noise is not necessary. Therefore we can apply PCA, up to only 2 components:



The result with 2 components offers an interesting perspective, the global shape is similar to the original in the way that the number of loops and their main direction are respected but the loops are not crossed in the same way and thus the reduced image is not sufficient to describe the original one (at least for some studies that would require the original shape). To do so it seems that 4 components are enough as follow:



To confirm this hypothesis, 4 components give an energy of 92,5 %, where 2 components only gave 65,0 %

(a) The implementation and test of this task, including the understanding of the concept, took approximately 5 hours
(b) As suggested in the exercise sheet, accuracy was measured either by simply observing the results and comparing the shape to the original data or (mainly) by computing the energy
(c) The fact that decomposing the data set such that some components could be removed without impacting the main shape was not totally new but the possibility to observe it that easily was offered a new tool in data analysis. While for the first data set it was pretty easy to understand that the data could be assimilated to a line, the other two parts thought us a lot about the dimension of objects we would not assume so much could be removed.

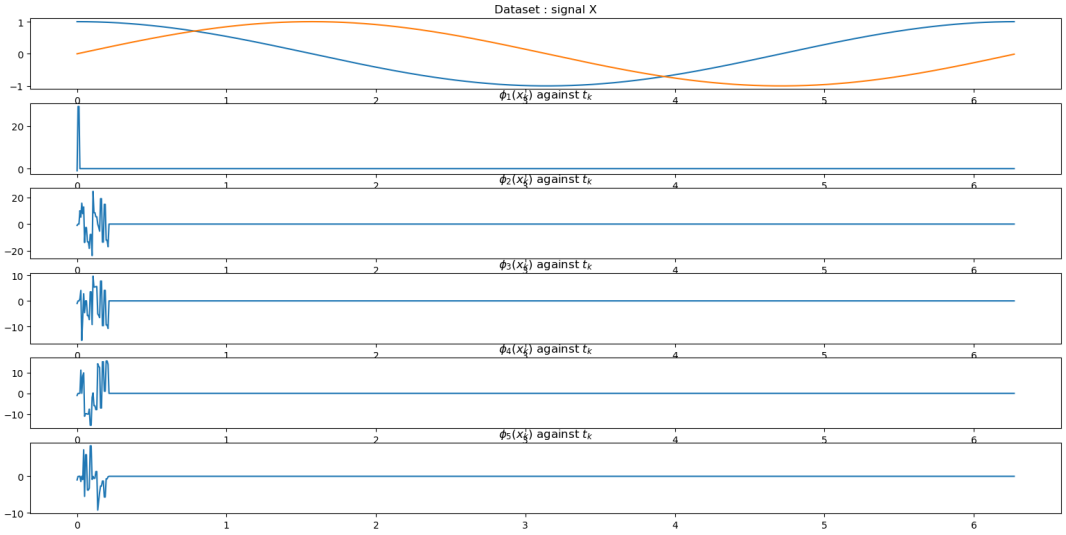
Report on task 2/5, Diffusion Maps

Part One

We computed the diffusion algorithm in Python, it is in the file Task2.py, in the function named diffusion_map. We followed the algorithm given in class.

Then we computed the eighenfunction of the first dataset : $X = (cos(t_k); sin(t_k)); t_k = \frac{2k}{N+1}$

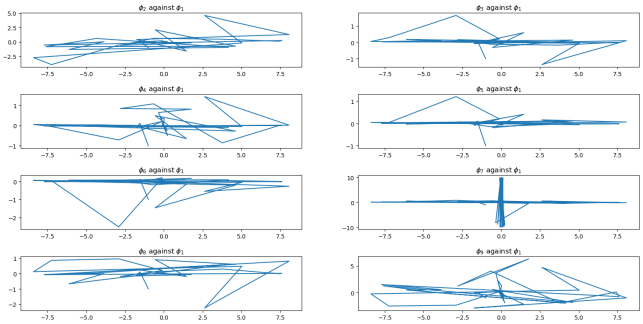
When plotting the eighenfunctions against time, we get a constant for the first eighenfunction, and then we have more complex signals. We see that each eighenfunction conveys a part of the information of the signal X, as the Fourier coefficient would.



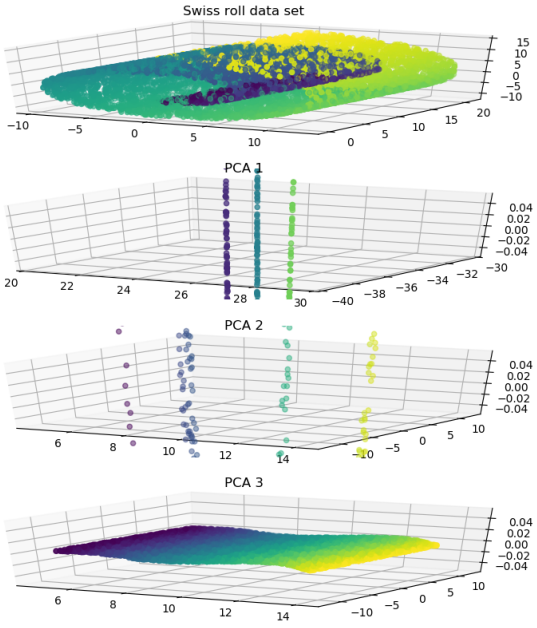
Part Two

We now use the swiss-roll dataset as a X.

First, we plot the eighenfunctions against each other. The functions never seem to be a function of phi_1 the first non_$constant eighen function. I don't really understand why$.

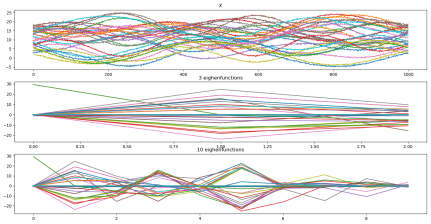


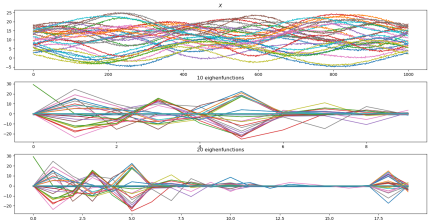
Second, we compute the 3 principal components of the swiss roll-data set. We choose to plot them in 3D because it is easier to understand why we cannot use less than 3 principal components to map the data : as you can see below, the data is 3 dimensional, and each direction has essential information to understand the data set. So plotting a reduction that would give less than 3 components would imply a loss of information that will make it impossible to map the dataset with only these 2 components.



Part Three

To get an idea of how many components we need for representing the data set, we plotted X and we plotted the eighenfunctions to have an idea if the data is well represented by the eighenfunctions. Looking at the plot, 10 eighenfunctions seems enough to get the whole energy, but this is a very qualitative analysis.





Report on task 3/5, Training a Variational Autoencoder on MNIST

In this task, variational autoencoder is implemented.It took for me 4 days to implement because I had difficulties in overcoming the error related with the loss function. It was also challenging to grasp the mathematical proof behind the reparametrization trick. The test of the algorithm took also considerably long especially for the cases which required 50 epochs while GPU was not used in my local hardware.
Loss function is used in order to measure the accuracy of the model during the tests. The results are also printed in order to manually check the results by eyes. All digits were present in the generated output. The decreasing loss function also assured that the algorithm was working properly.
I learned with the right configuration of hyperparameters and with large enough data, a successful model can be trained in order to generate digits in a chosen size of picture. It was impressive to observe the generation of a digit with a given noisy input to the decoder. I also learned the importance of the latent layer's dimensions because this is where the data is represented. So we should be able to store enough information in order to successfully represent the input data. The model learned the patterns in the input images in order to generate new images using those patterns. The high dimensionality in the input data is compressed and represented in the latent layer. It was also impressive to be able to represent these features as a distribution which made the generation of new images possible with the regularization effect unlike standard autoencoders.

We did not use any activation functions for the mean and standard deviation of the approximate posterior and the likelihood because we were satisfied with the results. We also were not able to find any documents or examples which use activation function for these layers. In case we had to use an activation function, we would use sigmoid in order to normalize the input for the decoder layer such as in the range from -1 to 1. When we want to generate new data, we tell the model a data point which did not exist before. So we expect to use the model to predict what the output should be given the previous training data. Obtaining good reconstructed but bad generated digits can be caused by bad regularization. Overfitting happens when your model represents the input data perfectly but does not know what to do when an input out of the training data is given. A well generalized model's loss would increase but at the same time it would better generate a non-existing data. We can ensure underfitting and overfitting at the rate we want by plotting the validation loss and training loss. We expect that the validation loss and training loss follow a parallel trend with each other. Training loss curve is expected to be under validation loss curve. If we generalize too much we can cause the training loss to diverge. In this case, techniques like early stopping and hyperparameter tuning are very useful.
As we can see from figures 1, 2 and 3 the representation of 2-Dimensional Latent space, the dimensionality of the images are reduced and can be represented. The digits are grouped as expected for epoch 5, 25 and 50.
We can observe that the model showed a good performance with 2-dimensional latent space even with only 5 epochs. However we are able to observe the change in reconstructed digits as the model was able to store and represent more information in the latent layer. We would expect to get the generated digits also better as the latent layer dimensions increase, however it was not the case. As we observe from figures 11 and 12, the model with 2-Dimensional latent space performed better in generating digits than the model with 32-Dimensional latent space. This might be the result of overfitting. Both models were trained for 50 epochs and as a result of storing too much information in the latent layer, the validation loss increased even though the reconstructed digits got better. We can also observe the same affect from figures 15 and 16. The model with 32-Dimensional latent space, which is trained for 50 epochs, has much less loss than the model with 2-dimensional latent space. However the model with 2-dimensional latent space perform better with generation of digits as the model with 32-dimensional latent space overfits the input data and fails to be successful in regulation to generate new data. Finally, we can say that we observed the most accurate results with the model with 32 dimensional latent layer which is trained for 5 epochs because of the early stopping's generalizing effect.

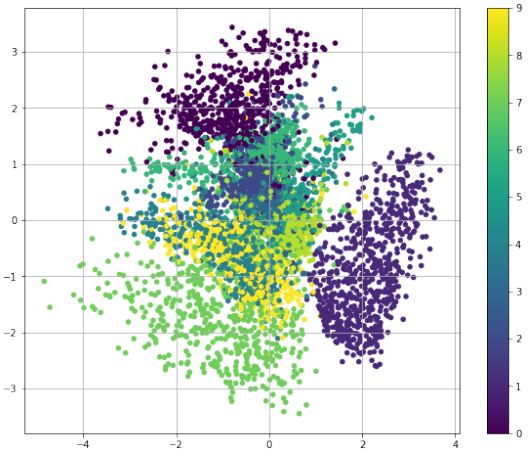


Figure 1: 2-Dimensional Latent Space, Epoch = 5

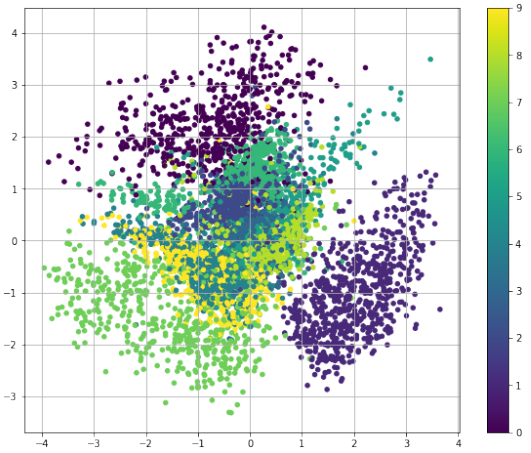


Figure 2: 2-Dimensional Latent Space, Epoch = 25

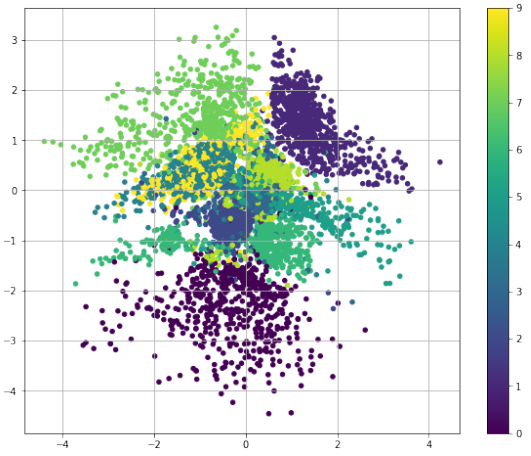


Figure 3: 2-Dimensional Latent Space, Epoch = 50

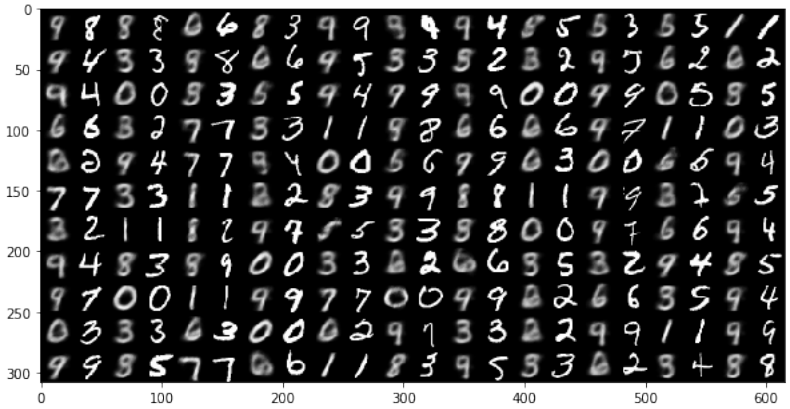


Figure 4: Reconstructed Digits With Corresponding Original Ones, Epoch = 5

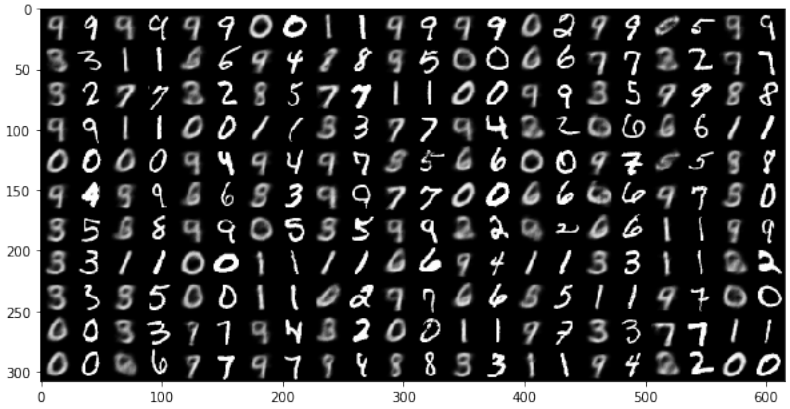


Figure 5: Reconstructed Digits With Corresponding Original Ones, Epoch = 25

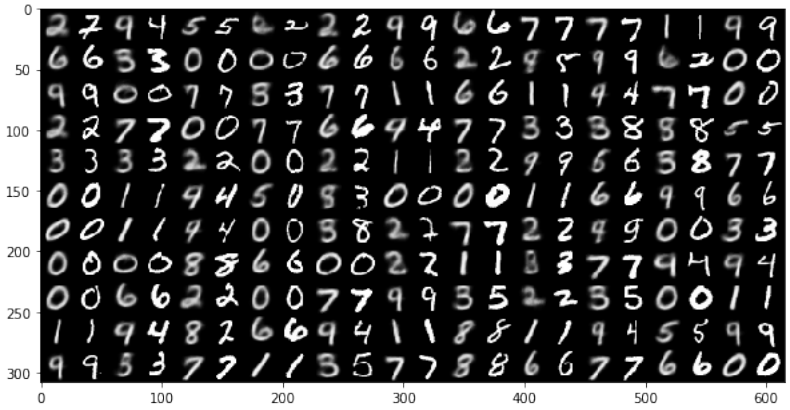


Figure 6: Reconstructed Digits With Corresponding Original Ones, Epoch = 50

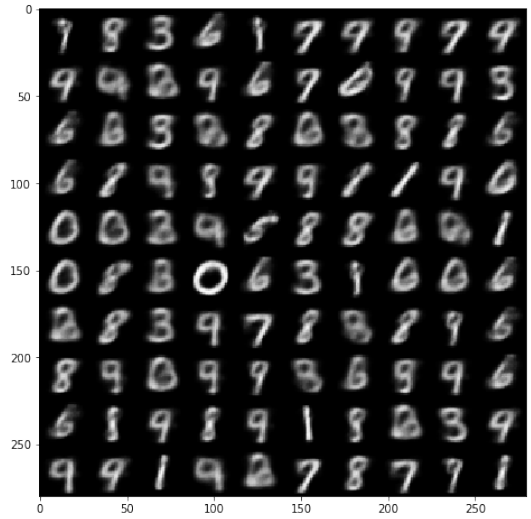


Figure 7: Generated Digits Based On The Trained Model, 2-Dimensional Latent Space, Epoch = 5

Figure 8: Generated Digits Based On The Trained Model, 32-Dimensional Latent Space, Epoch = 5

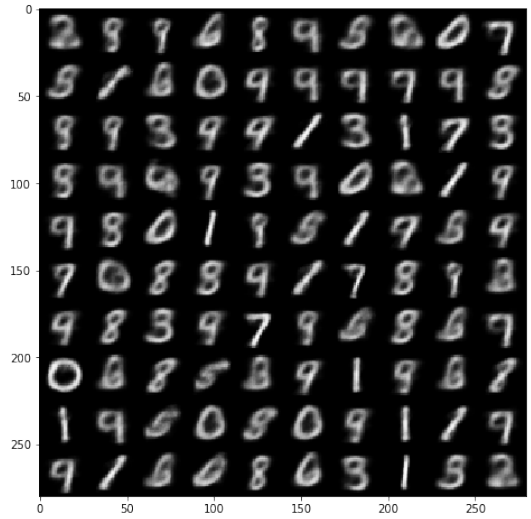


Figure 9: Generated Digits Based On The Trained Model, 2-Dimensional Latent Space, Epoch = 25

Figure 10: Generated Digits Based On The Trained Model, 32-Dimensional Latent Space, Epoch = 25

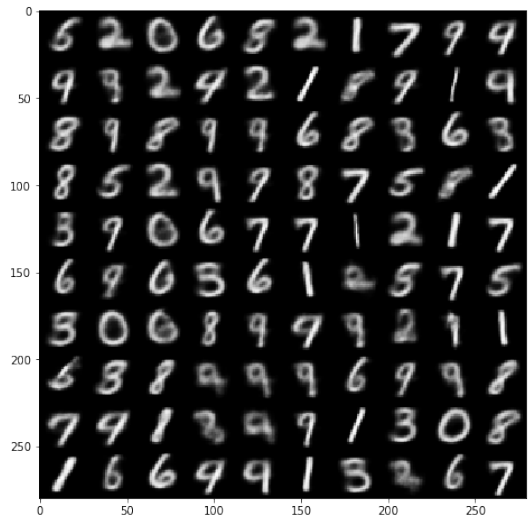


Figure 11: Generated Digits Based On The Trained Model, 2-Dimensional Latent Space, Epoch = 50

Figure 12: Generated Digits Based On The Trained Model, 32-Dimensional Latent Space, Epoch = 50

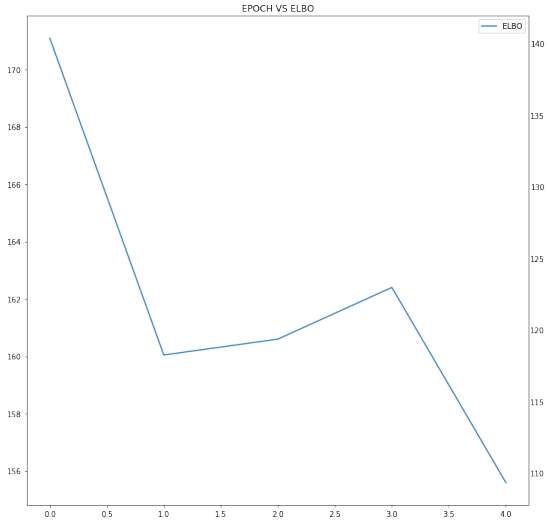


Figure 13: Loss Curve, 2-Dimensional Latent Space, Epoch = 5

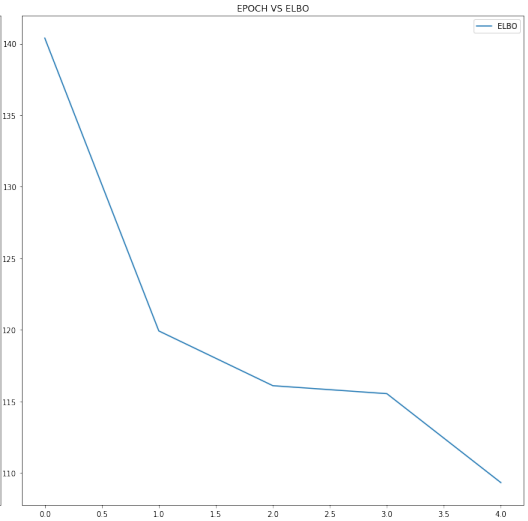


Figure 14: Loss Curve, 32-Dimensional Latent Space, Epoch = 5

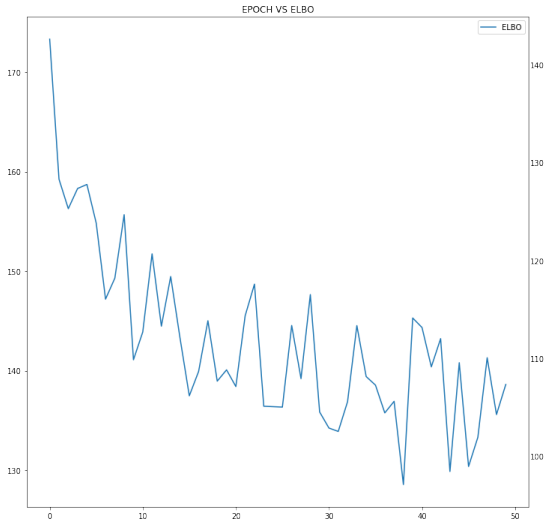


Figure 15: Loss Curve, 2-Dimensional Latent Space, Epoch = 50

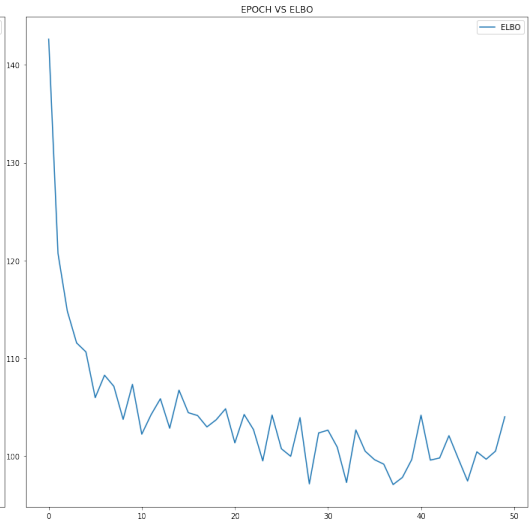


Figure 16: Loss Curve, 32-Dimensional Latent Space, Epoch = 50

Report on task 4/5, Fire Evacuation Planning for the MI Building

The algorithm from the previous task was used in this task with the FireEvac dataset. It is observed that we obtain better results if we increase the batch size. It is observed from the latent loss that gradients vanished even tough we did not have too many layers or epochs. In order to prevent vanishing gradients, we assigned coefficient to the reconstruction and latent loss. We could not improve our total loss although we tried many different combinations in the range from the changes of dimensionality of layers to the quantity of layers or from number of epochs to different activation functions. As we did not receive any response to our e-mails from Alexej whatsoever, we had difficulty in improving our code. At the end we found that if there are 2227 people in the building, the main entrance would exceed with 1006 people.

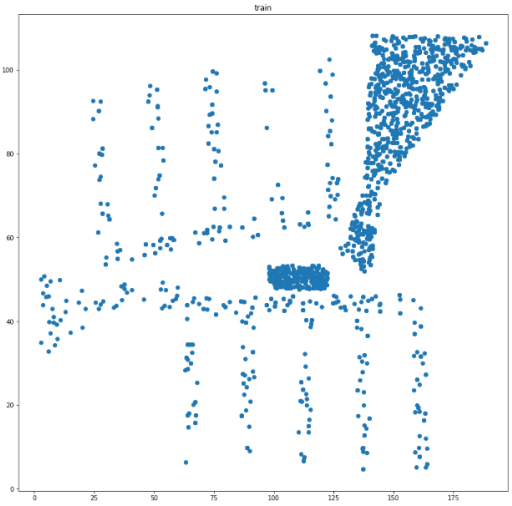


Figure 17: Scatter Plot of Train Dataset

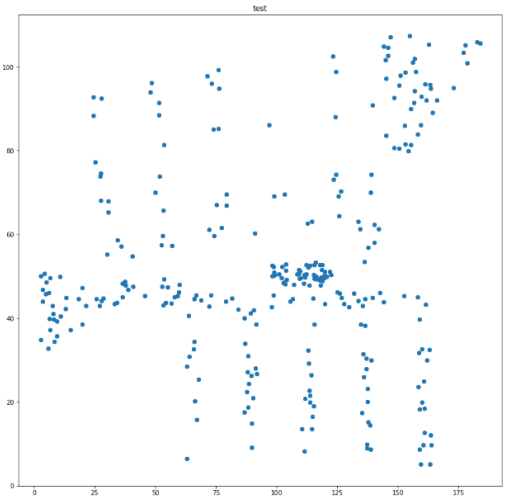


Figure 18: Scatter Plot of Test Dataset

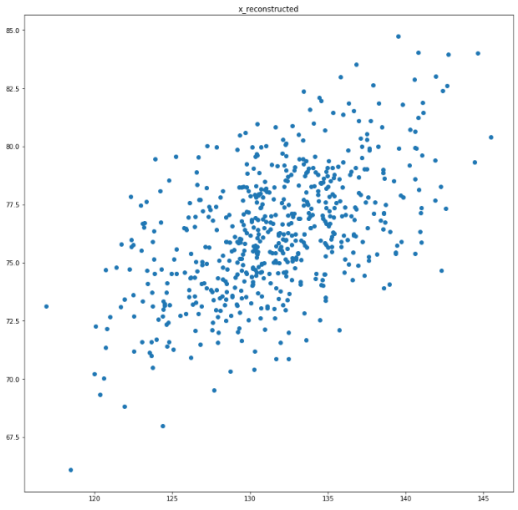


Figure 19: Scatter Plot of Reconstructed Test Set

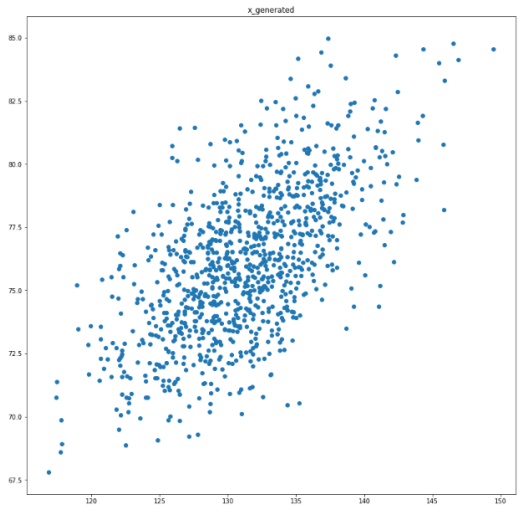


Figure 20: Scatter Plot of 1000 Generated Samples

https://wiseodd.github.io/techblog/2016/12/10/variational-autoencoder/