Report for exercise 4 from group H

Tasks addressed:	4
Authors:	Ahmad Bin Qasim (03693345) Kaan Atukalp (03709123) Martin Meinel (03710370)
Last compiled:	2019–12–12
Source code:	https://gitlab.lrz.de/ga53rog/praktikum-ml-crowd

The work on tasks was divided in the following way:

Ahmad Bin Qasim (03693345)	Task 1	33%
	Task 2	33%
	Task 3	33%
	Task 4	33%
Kaan Atukalp (03709123)	Task 1	33%
	Task 2	33%
	Task 3	33%
	Task 4	33%
Martin Meinel (03710370)	Task 1	33%
	Task 2	33%
	Task 3	33%
	Task 4	33%

Report on task 1, Principal component analysis

First part: We used the data set from moodle and applied Principal Component Analysis. In Figure 1 you can see the original data set with the two principal components marked in red. The first principal component is the one pointing to the right upper corner. So the second one is the other principal component pointing towards the left upper corner and being orthogonal to the first principal component.

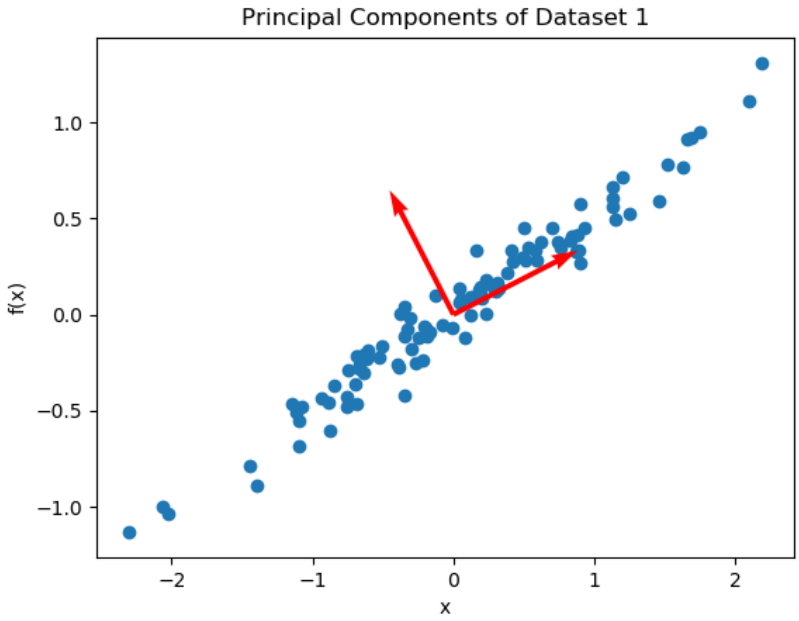


Figure 1: Plot of the data set with the two first principal components

The energy of a principal component describes how much variance of the data the principal component covers. In the following table you can see the energy of the drawn two principal components and the sum of them. It can be seen that the first principal component has a big energy value and therefore covers almost all the variance of the data. From the sum of both energy values it can be seen that both principal components cover the entire data set.

	Energy
Principal Component 1	0.993
Principal Component 2	0.07
First two components	1.00

Second Part: At first we used the rows of the images as single data points, but after seeing the reconstructed images, we considered the columns of the image as single data points and the resulting reconstructed pictures were reconstructed much better than before. Figure 2 shows all the four reconstructed images starting with using all principal components to only using the first ten principal components.

(a) Reconstructed image of a raccoon using all principal components

(b) Reconstructed image of a raccoon using the first 120 principal components

(c) Reconstructed image of a raccoon using the first 50 principal components

(d) Reconstructed image of a raccoon using the first 10 principal components

Figure 2: The reconstructed images of a raccoon using different numbers of principal components

It can be seen that the reconstructed images using the first 120 or first 50 are close to the original picture. The reconstructed image where only the first 10 principal components are used seems blurry but the image can still be recognized. By using the first 20 out of 1023 principal components, we can cover 99.01% of the variance.

Third Part: We have a data file containing the (x,y) positions for 15 pedestrians over 1000 time steps. Figure 3 visualizes the path from the first two pedestrians in space. It shows that the first two pedestrians are walking in loops.

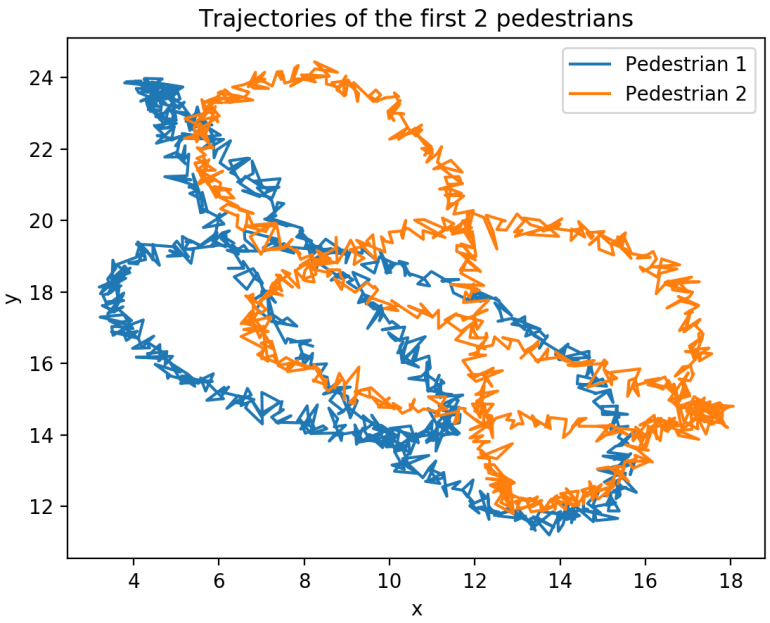


Figure 3: Visualized paths for the first two pedestrians over 1000 time steps

Now we use Principal Component Analysis and use the first two principal components of it to project the 30-dimensional data points to the first two principal components. The following table shows the energy values for the first two principal components and shows that the first two principal components cover a variance of almost 85% of the data.

	energy
Principal Component 1	0.473
Principal Component 2	0.376
First two principal components together	0.849

We tried to reconstruct the original trajectories from Figure 3 with the first two principal components. The following Figure 4 shows the reconstructed trajectories for the first two pedestrians using the first two principal components.

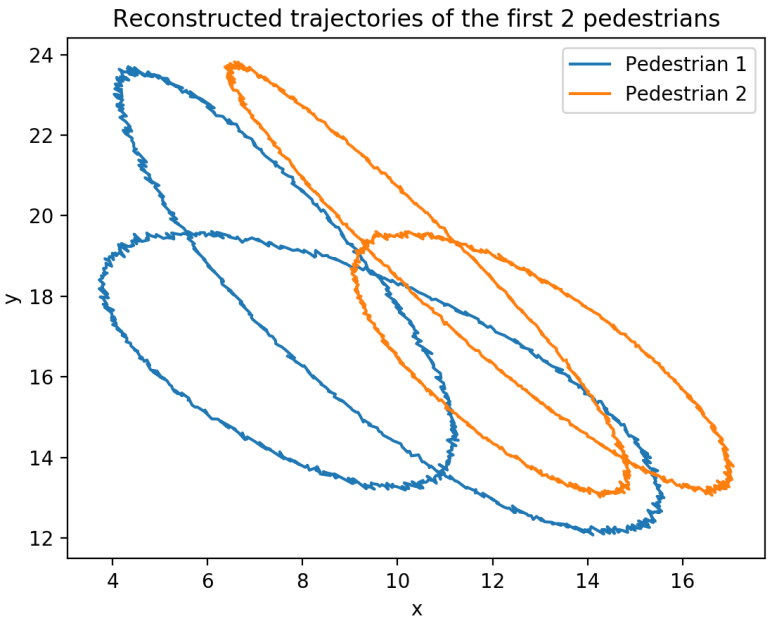


Figure 4: Reconstructed trajectories from using the first two principal components

From comparing the trajectories of the first two pedestrians from Figure 3 to Figure 4 it can be seen that the trajectories are similar. Besides of that, the total amount of energy is almost 85% and this is why in our opinion the first two principal components are enough to capture most of the energy.

It took us two days to implement and test the implementation of task1.
We could represent the data very accurately. Especially for part 2 of task 1, it can be seen that even the first 50 principal component are sufficient to visualize the picture of the raccoon. We measured the accuracy by using the energy, which describes how much variance of the original data can be described by using the corresponding principal component. Besides of that, we generated the reconstructed images and compared them to the original image.
Principal Component Analysis is used for image compression. It was interesting to see how well it works and we learned that the result is different if you consider the rows or the columns of an image as single data points. So we learned that it makes sense to think about how to apply Principal Component Analysis to the data in advance. Furthermore we learned that often the first few principal components are sufficient to cover the data.

Report on task 2, Diffusion Maps

First Part: Five eigenfunctions of the given periodic data set were computed using Diffusion Maps. As seen in Figure 5, the eigenfunctions have mapped the data on scaled sine/cosine waves with varying periods.

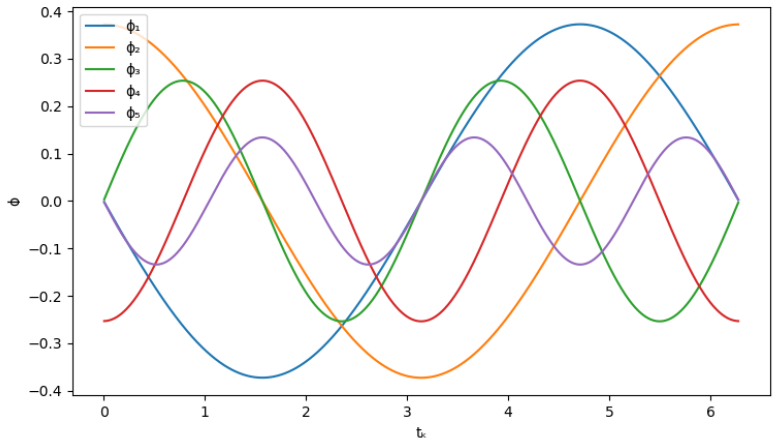


Figure 5: Values of the eigenfunctions with respect to the t value.

Second Part: Starting from $l = 5$, the eigenfunctions ϕ_l are not a function of ϕ_1 anymore. This can be seen in Figure 6 where the eigenfunctions for $l = 2, 3, 4$ could easily be represented with commonly known polynomial or trigonometric functions, whereas the others show no relevant correlation for all values of ϕ_1.

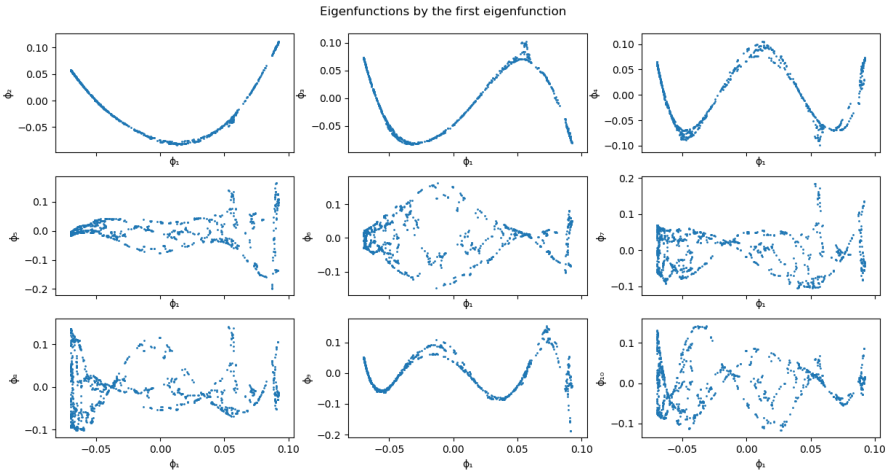


Figure 6: Values of eigenfunctions for $l = 2, 3, 4, 5, 6, 7, 8, 9, 10$ with respect to ϕ_1.

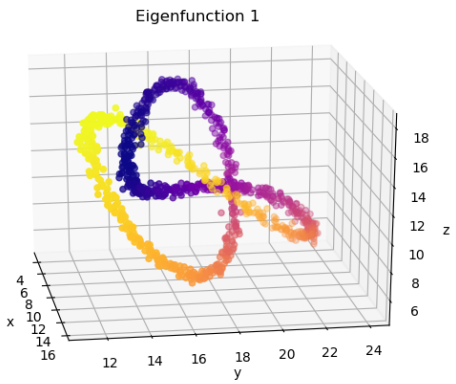
When PCA is applied to the data set, the resulting energy values for the first 3 principal components are as follows:

	energy
Principal Component 1	0.38
Principal Component 2	0.33
Principal Component 3	0.29

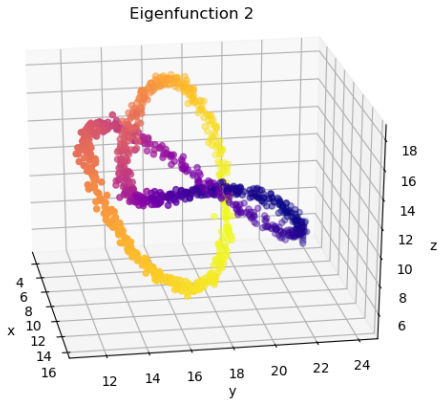
The first two principal components cover only 72% of the data, so we would lose 29% of the data. Thus, it makes more sense to take all principal components instead of the first two.

Third Part: By plotting the eigenfunctions in a 3D plot as seen in Figure 7, it can be seen that the first two eigenfunctions are enough to capture all the data. Not a single pair of points seems to have the same value (colour) pair in the first 2 eigenfunctions, which can be seen when the Figures 7a and 7b are compared side by

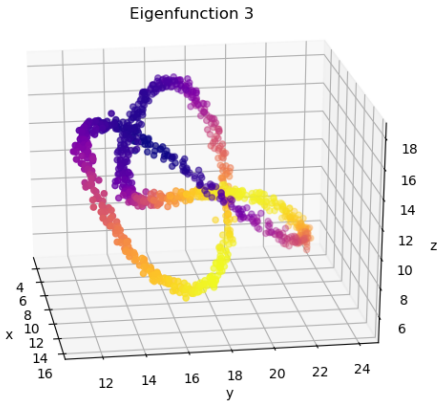
side.



(a) Values assigned by the first eigenfunction



(b) Values assigned by the second eigenfunction



(c) Values assigned by the third eigenfunction

Figure 7: Graphs showing the values assigned for the first three eigenfunctions. Brighter colour indicates a higher real number.

Report on task 3, Training a Variational Autoencoder on MNIST

While initializing the VAE model class object, a configuration dictionary can be sent as an argument to the init function of the VAE class, in order to dynamically adjust the model parameters. There are a number of configurations that can be set using the dictionary [1]. Most of the model configurations set for task 3, are specified in the exercise sheet [2]. In addition to the configurations provided in the exercise sheet, related to the model, the following notable additions are made:

- The output of the decoder is not the mean of likelihood but the predicted data points. This modification within the decoder can be considered as probabilistic as well because it is equivalent to modeling likelihood as Gaussian with identity covariance
- Mean Squared Error (MSE) is used to calculate the reconstruction loss with the predicted data points from decoder and the input data as the target.

Table 1: The VAE class initialization arguments

Argument	Description
latent_vector_size	The dimensions of the latent vector
print_output	Print the output plots (True or False)
batch_size	Batch Size to be used
learning_rate	Learning rate
epochs	Number of epochs
train_dataloader	Pytorch dataloader for training set
test_dataloader	Pytorch dataloader for test set
dataset_dims	The dimensions of the dataset
test_count	The number of test set data points
kl_annealing	The flag for using KL annealing approach (explained in task 4, True or False)
beta_limit	The limit upto which the weight for latent loss is increased to, in case of KL-annealing
generated_loss_scale	The reconstruction loss weight constant

Table 2: Task 3 model configurations

Argument	Value
latent_vector_size	2 or 32
print_output	True
batch_size	128
learning_rate	0.001
epochs	50
train_dataloader	mnist training set loader object
test_dataloader	mnist test set loader object
dataset_dims	784
test_count	16
kl_annealing	False
beta_limit	-
reconstruction_loss_scale	1

1. We used a linear activation function to approximate the mean and standard deviation of the posterior distribution because, the mean and standard deviation values can be unbounded. We considered different activation functions but discarded each one of them because of their bounded output, as the range of mean and standard deviation of the posterior distribution is $(-\infty, +\infty)$. [3]

Table 3: Different activation functions which we considered

Activation Function	Bounds
Sigmoid	(0,1)
Tanh	(-1,1)
Relu	max(0,x)

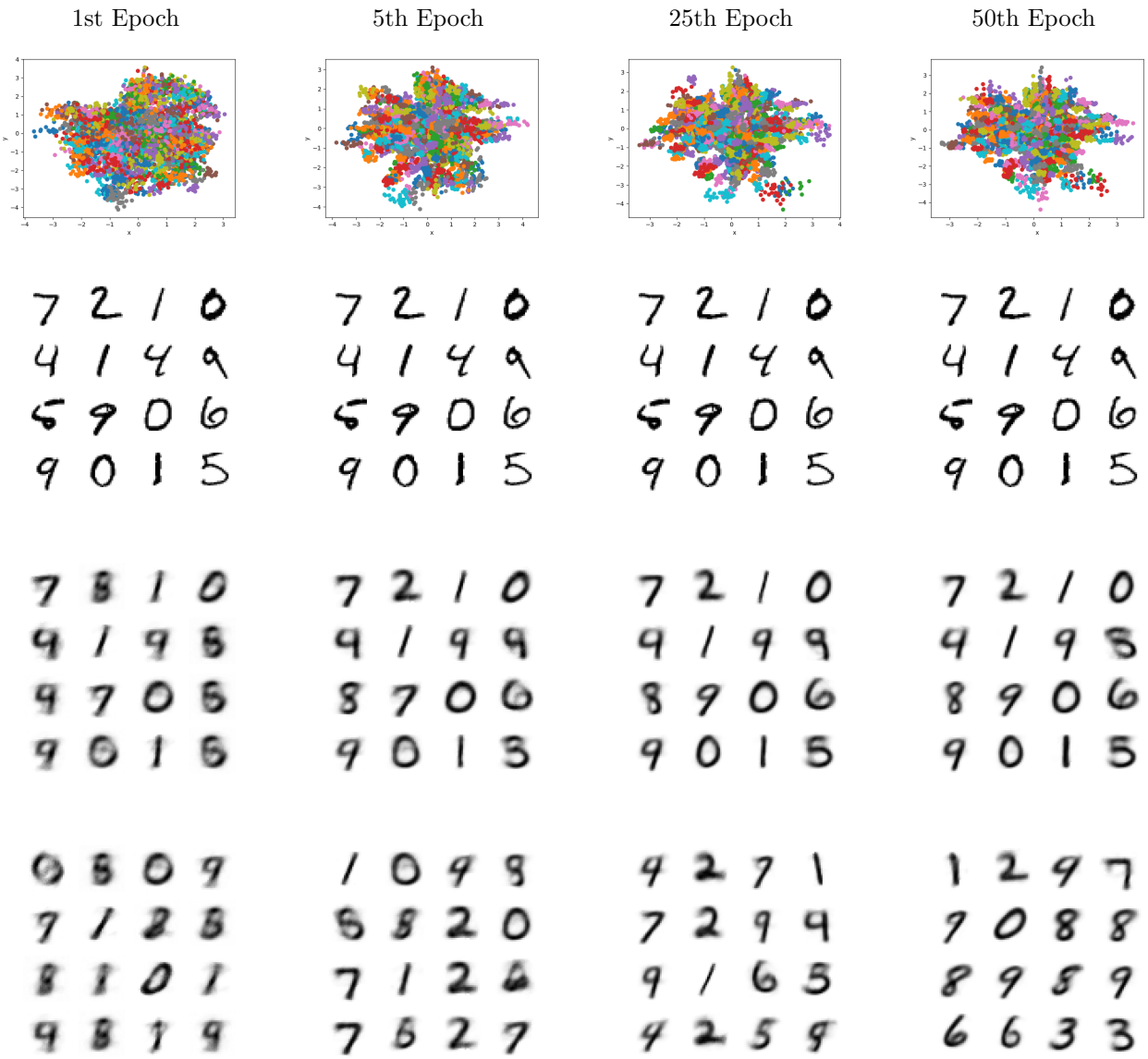
2. If the reconstructed image are much better then the generated images then, it means that the model has been overfitted to the input data distribution. This can happen when the KL-divergence loss (latent loss)

of the approximated posterior distribution does not converge while the reconstruction loss of the input distribution converges.

In order to solve this problem, the reconstruction loss and the KL-divergence loss can be weighted to give more weight to the latter.

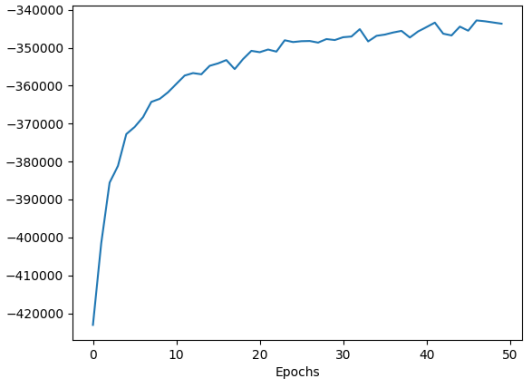
3. After training the Variational Autoencoder model, We plotted the desired graphs. The latent space of the encoder is 2 dimensional. [8] shows, the results obtained using 2 dimensional latent space. Each row within the figure, depict the latent space representation, original images, reconstructed images and generated images respectively at epochs 1, 5, 25 and 50.

Figure 8: Results obtained with a 2 dimensional latent space



4. As the VAE is trained for more epochs, the loss decreases. [9]

Figure 9: Epochs vs -ELBO loss



5. The mnist images generated using 32 dimensional latent space are not as good as the images generated using 2 dimensional latent space in terms of representing the input mnist dataset, as the former contain some visually distorting artifacts which are not present in latter. [10]. An explanation for this result is that, in case of 32 dimensional latent space, the VAE over-fits itself on the input data distribution. This does not happen when 2 dimensional latent space is used because it is more challenging for the encoder to map input data distribution within 2 dimensional latent space. A strong evidence for this explanation can be put forth by comparing the reconstructed images obtained using 32 dimensional latent space [11] to the generated images [10], the former being visually superior to the latter. The loss in this decreases more rapidly and within 50 epochs, the total loss per epoch reaches a lower value compared to the VAE trained with 2 dimensional latent space. [12]

Figure 10: Generated results obtained with a 32 dimensional latent space

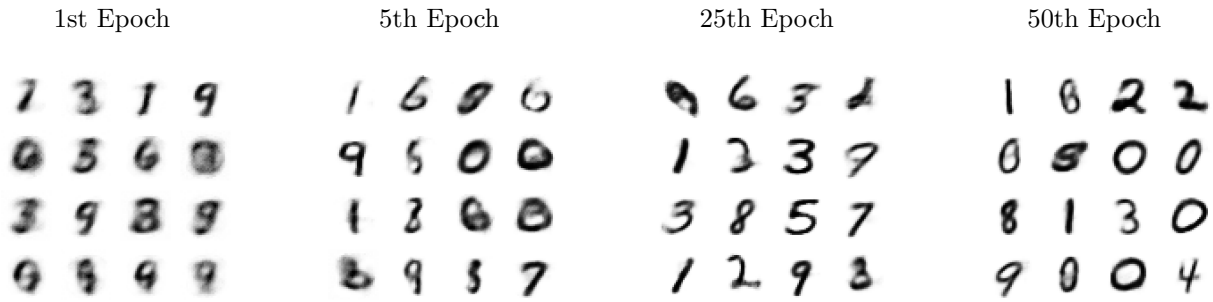


Figure 11: Reconstructed results obtained with a 32 dimensional latent space

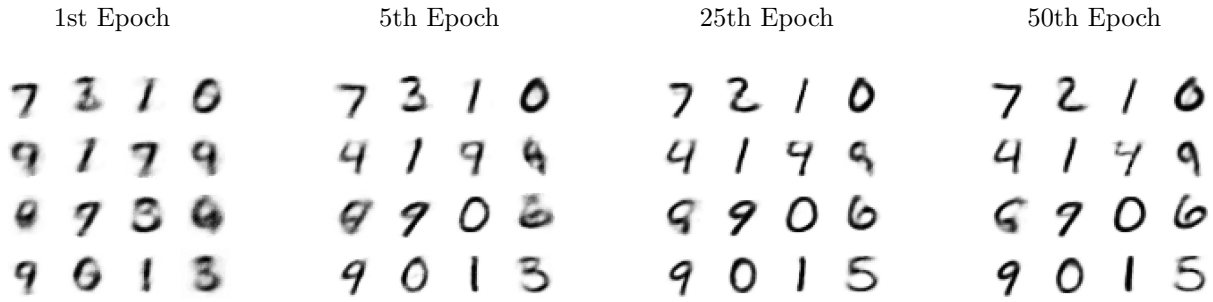
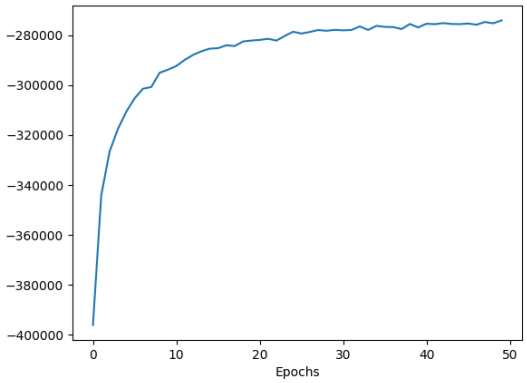


Figure 12: Epochs vs -ELBO loss



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

The following changes are made in the model configurations compared to task 3 [4]:

- The VAE is trained for 1000 epochs, as 50 epochs are not enough for the VAE to achieve a reasonable approximation of posterior distribution.
- The learning rate of the adam optimizer is set to 0.0001 because the higher learning rate of 0.001 used in task 3 proves to be too high for training the FireEvac data. It results in rapid changes in model accuracy and the model fails to converge.
- Sigmoid activation function is used for the last layer of decoder, as the FireEvac data is normalized using max normalization, so the input data has a range of (0, 1) i.e. the same as the range of sigmoid function.
- The problem of posterior collapse in VAEs is a well-known one. The objective function of a VAE consists of two terms (1), where $\mathbb{Q}(z|X)$ is the approximate posterior, $\mathbb{P}(z)$ is the prior, X is the input data and $\mathbb{P}(X|z)$ is the likelihood. First term is also labeled as reconstruction loss while the second as latent loss.

$$L = -\mathbb{E}[\log \mathbb{P}(X|z)] + D(\mathbb{Q}(z|X)||\mathbb{P}(z)) \tag{1}$$

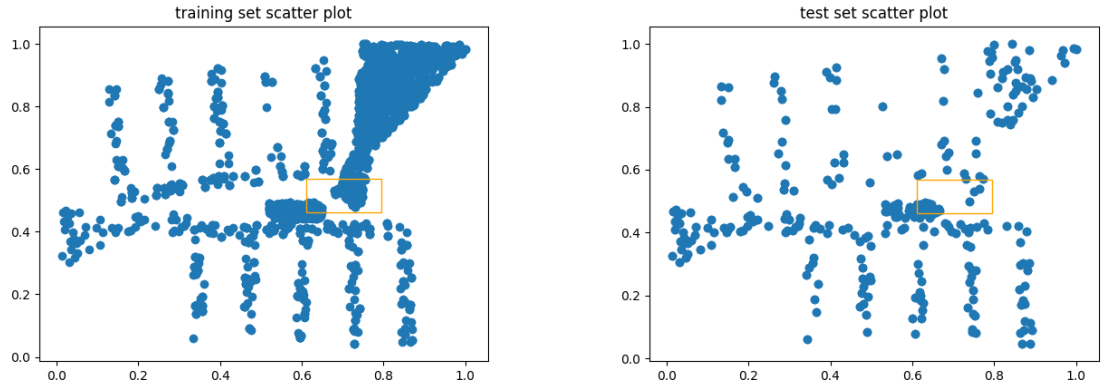
  Posterior collapse occurs when the second term, $\mathbb{Q}(z|X)$ converges to zero while the reconstruction loss is still high i.e. the approximate posterior becomes equal to the prior. In order to solve this problem, an approach called KL-annealing is used. Under this approach, the VAE is trained with reconstruction loss only for a number of epochs and then the latent loss term is introduced gradually, by increasing its weight from 0 to 1 till the end of training. This approach keeps the latent loss, from converging to zero.
- Another issue that we faced while training the VAE on FireEvac dataset after solving the posterior collapse problem was the domination of latent loss over reconstruction loss. As the FireEvac dataset is two dimensional, the MSE values i.e. reconstruction loss obtained, are very small compared to the latent loss. Due to this, the VAE is optimized based on the latent loss to a much higher degree then, compared to the reconstruction loss. In order to balance the two loss terms, we use a reconstruction_loss_scale multiplier hyper-parameter, which is multiplied with the reconstruction loss. We set its value to 100.
- After a hyper-parameter search, we found that a 4 dimensional latent space, provides the best results for this task.

Table 4: Task 4 model configurations

Argument	Value
latent_vector_size	4
print_output	False
batch_size	1024
learning_rate	0.0001
epochs	1000
train_dataloader	FireEvac training set loader object
test_dataloader	FireEvac test set loader object
dataset_dims	2
test_count	1000
kl_annealing	True
beta_limit	1
reconstruction_loss_scale	100

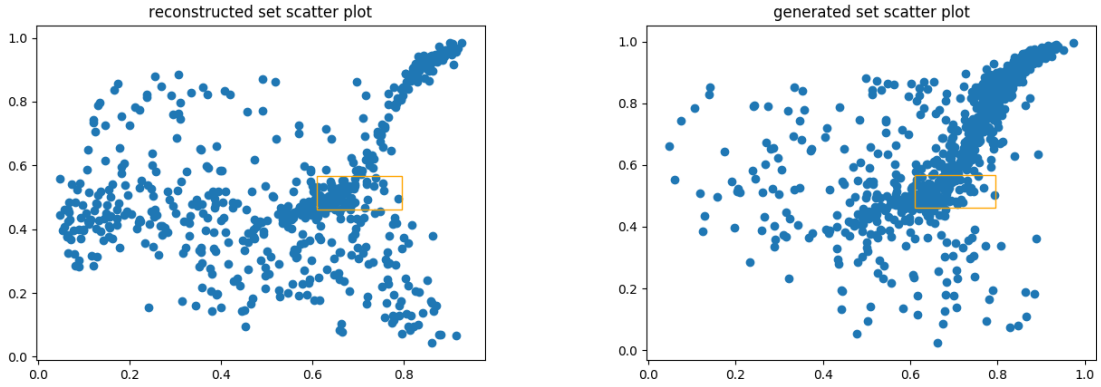
1. The training and test datasets reflect the positions of the students and employees in the campus hallways. The data has been normalized using max normalization.

Figure 13: The scatter plot of the training and test sets



2. The same VAE implementation is used for training a VAE on the FireEvac data as in task 3 with the modifications discussed above.
3. The scatter plot of the reconstructed data from the test set. [14]
4. The scatter plot of the generated data. [14]

Figure 14: The scatter plot of the reconstructed and generated sets



5. Approximately 861 samples are needed to reach the critical number at the main entrance. This value is approximated by repeatedly sampling from the approximated posterior distribution, after training of VAE is complete until the critical limit is reached 15. This process was repeated 10 times and the mean of the number of samples is noted.

Figure 15: Scatter plot of samples drawn, until critical limit is reached