Report for exercise 4 from group H

Tasks addressed:	4
Authors:	Ahmad Bin Qasim (03693345) Kaan Atukalp (03709123) Martin Meinel (03710370)
Last compiled:	2019–12–10
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%
	Task 5	33%
Kaan Atukalp (03709123)	Task 1	33%
	Task 2	33%
	Task 3	33%
	Task 4	33%
	Task 5	33%
Martin Meinel (03710370)	Task 1	33%
	Task 2	33%
	Task 3	33%
	Task 4	33%
	Task 5	33%

Report on task 1, Principal component analysis

First part: We used the dataset 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.

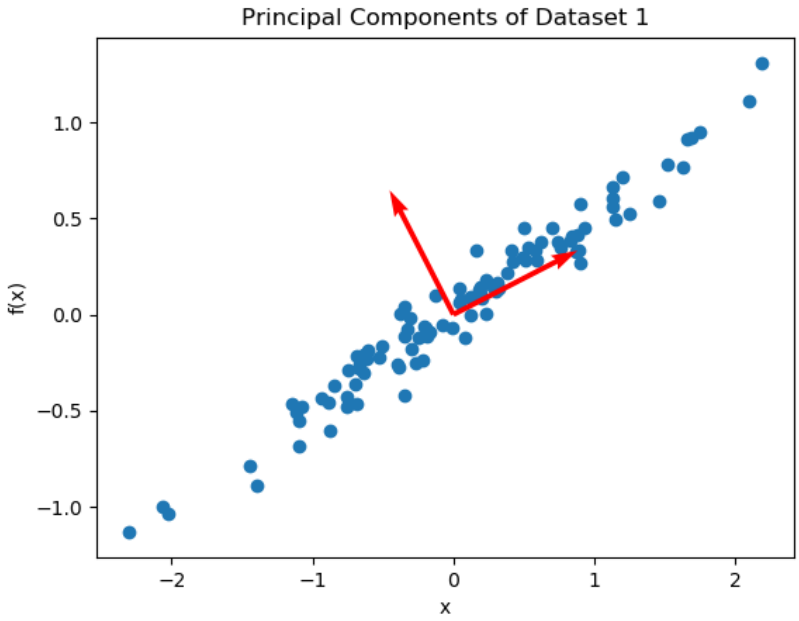


Figure 1: Plot of the dataset with the two first principal components

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.

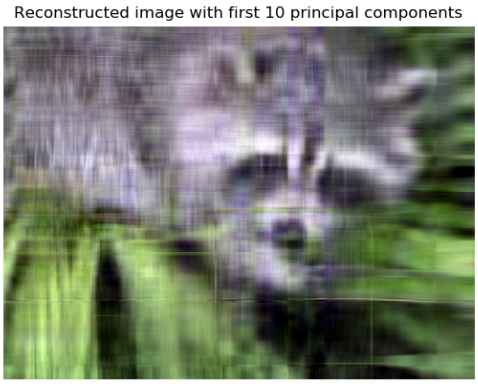
	Energy
Principal Component 1	0.993
Principal Component 2	0.07
First two components	1.00

Second Part: At first we used the rows of the images as single data points, but after seeing the reconstructed images we changed and considered the columns of the image as single data points and the resulting reconstructed pictures were much better reconstructed 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 racoon using all principal components

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

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



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

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

It can be seen that the reconstucted images using the first 120 or first 50 are close to the original picture. The reconstructed image where only the first 10 principal compenents are used seems blurry but the image can still be recognized.

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.

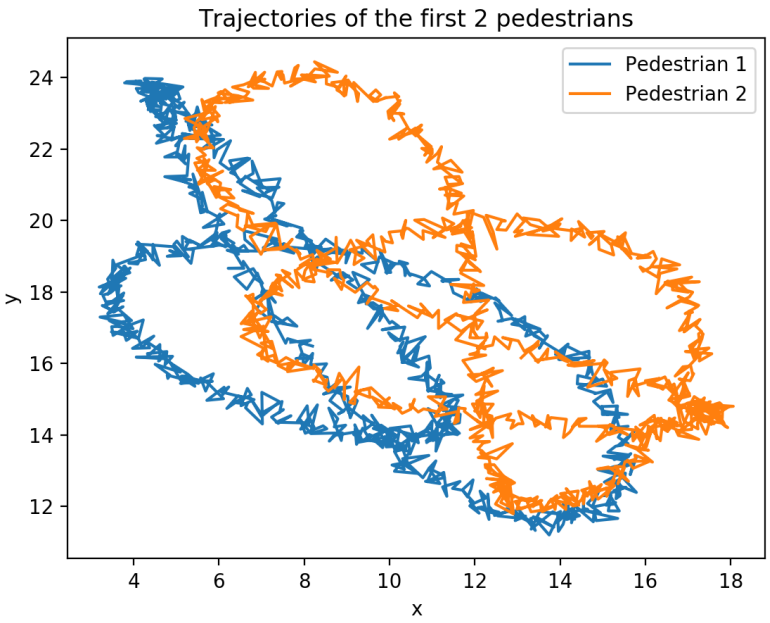


Figure 3: Visualized paths for the first two pedestrains over 1000 time steps

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 compenents cover a variance of almost 85% of the data.

	energy
Principal Component 1	0.473
Principal Component 2	0.376
First two principal compenents 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.

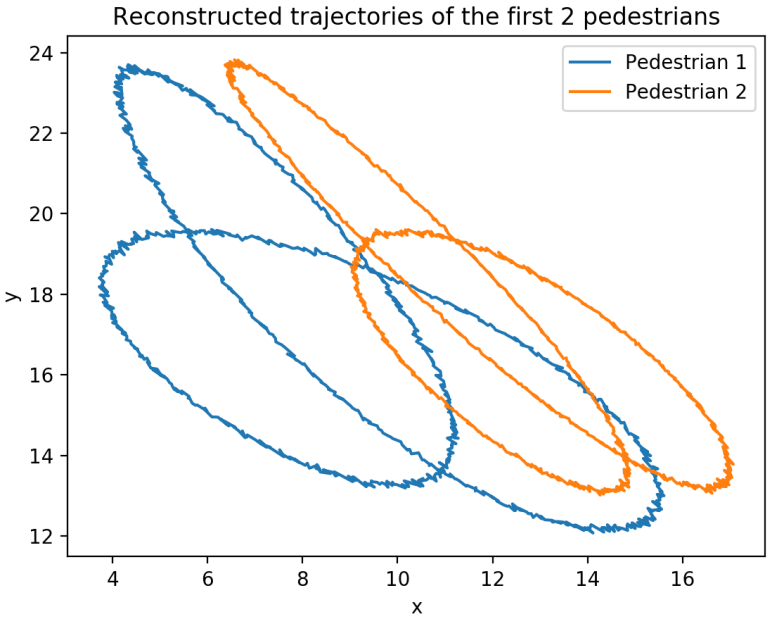


Figure 4: Reconstructed trajctories from using the first two principal components

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 almsot 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 accurate. Especially for part two of task one it can be seen that even the first 50 principal component are sufficient to visualize the picture of the racoon. We measured the accuracy by using the energy which describes how much variance of the original data can be desbribed 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 Anaylsis 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

Part 1:

Part 2: For a value $l = 5$ the eigenfunction ϕ_l is not anymore a function of ϕ_1. This can be seen in Figure....
Energy values: pc3 0.28867937283177014 pc2 0.3292051971173148 pc1 0.3821154300509151

The first two Principal components cover only 71.1% So we would lose 28.9% of the data. In comparison to that we can cover almost the whole dataset by taking the first three principal components. Consequently, it makes much more sense to take all principal components instead of two.

Part 3: By plotting the eigenfunctions in a 3d plot, we can see that the first two eigenfunctions are enough to capture all the data.

Report on task 3, Training a Variational Autoencoder on MNIST

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 are unbounded. Other activation functions bound the output value to a certain range. We considered different activation functions but discarded it because of the bounded output. [1]

Table 1: Different activation functions

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

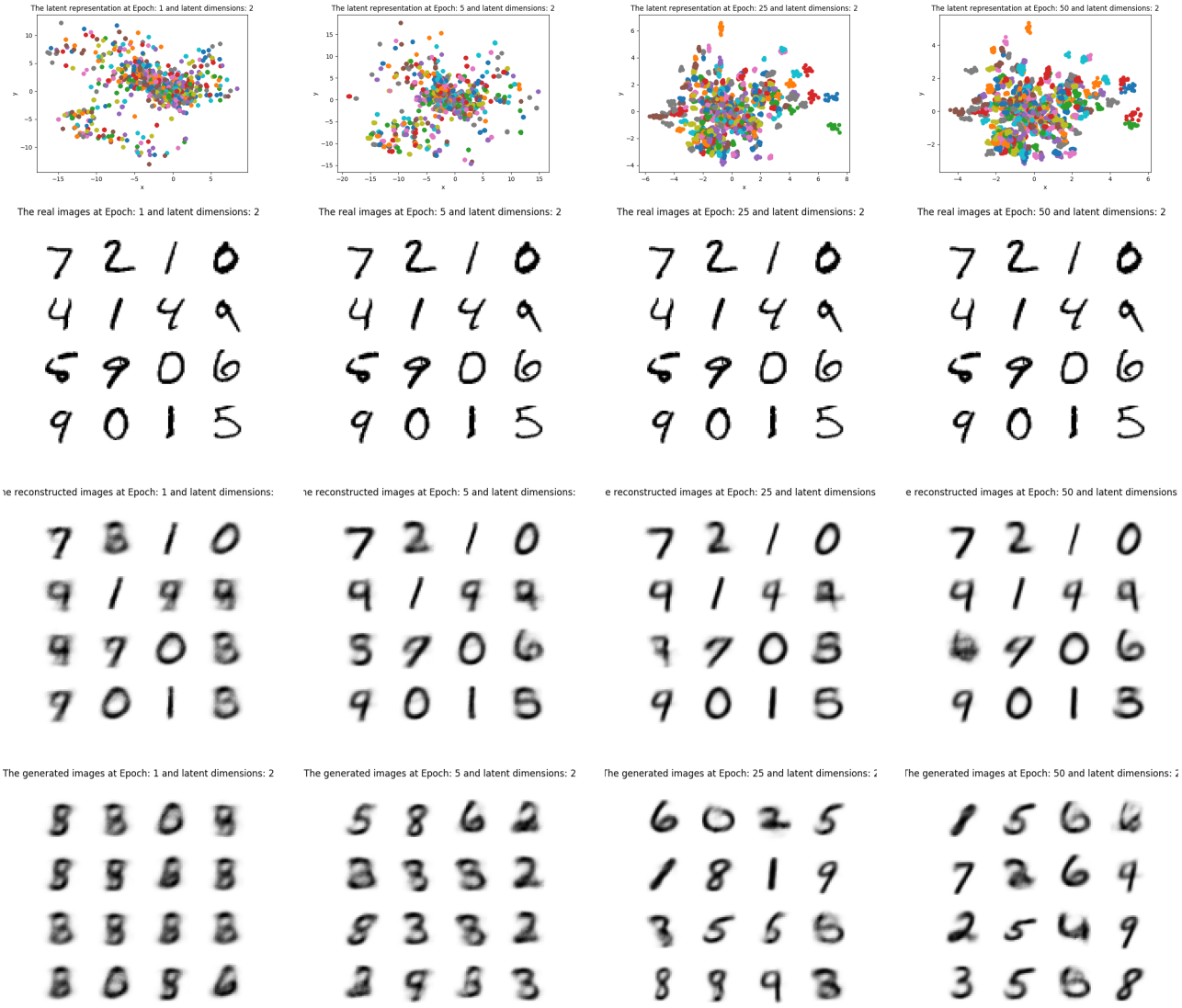
2. If the reconstructed image are much better then the generated images then, it means that the model has been overfitted to the input data distribution. This can happen when the KL-divergence loss (latent loss) of the approximated posterior distribution does not converge while the reconstruction loss 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. [5]. In addition to the configurations provided in the exercise sheet, related to the model, the following changes were made:

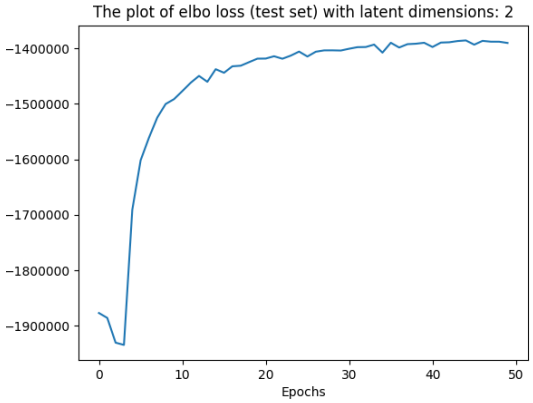
   - The output of the decoder is not the mean of likelihood but the predicted data points. This modification within the decoder can be considered as probabilistic as well because it is equivalent to modeling likelihood as Gaussian with identity covariance
   - Binary cross entropy is used to calculated the reconstruction loss with the predicted data points from decoder and the input data as the target.
   - The last layer of decoder uses a Sigmoid activation as the data in mnist is grayscale.

Figure 5: results obtained with a 2 dimensional latent space



4. As the VAE is trained for more epochs, the loss decreases. [6]

Figure 6: ELBO loss



5. While initializing the VAE model class object, a configuration dictionary can be sent as an argument to the init function, to dynamically change the model. There are a number of configurations that can be set using the dictionary. [2].

Table 2: 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
mode	The dataset mode (mnist or mi)
test_count	The number of test set data points

The mnist images generated using 32 dimensional latent space are more sharper. [7]. 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. [8]

Figure 7: results obtained with a 32 dimensional latent space

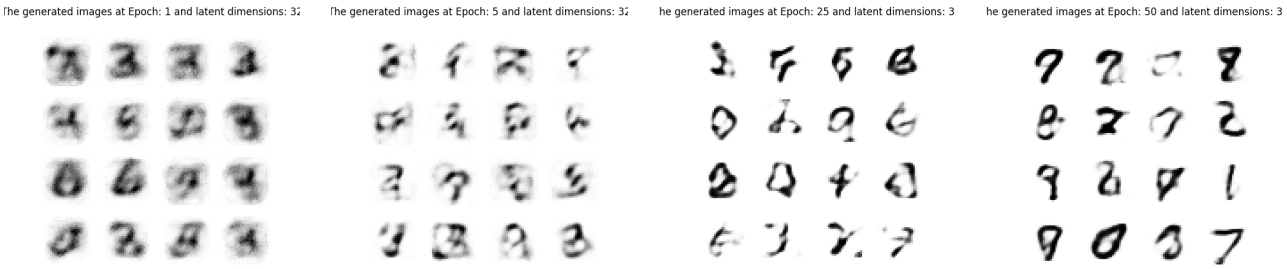
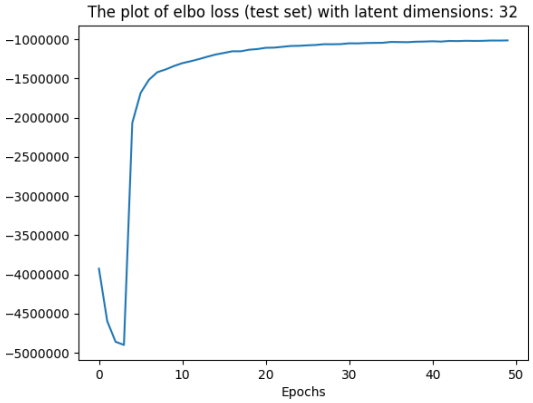


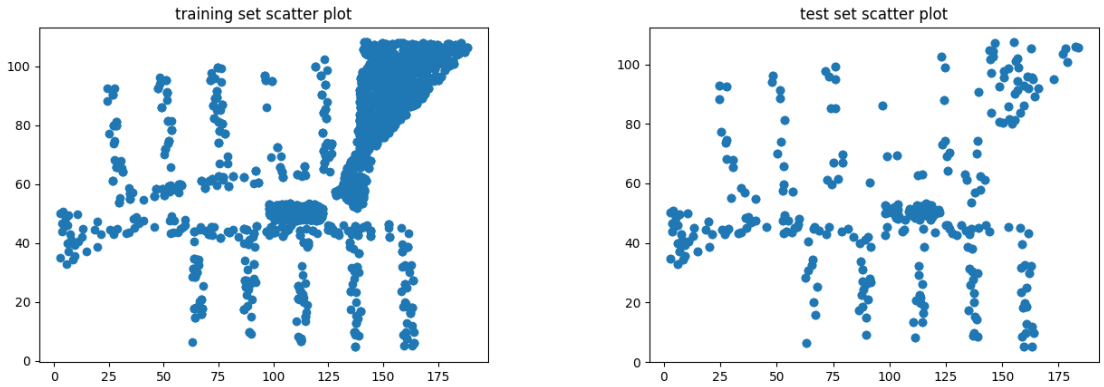
Figure 8: ELBO loss



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

1. The training and test datasets reflect the positions of the students and employees in the campus hallways.

Figure 9: The scatter plot of the training and test sets



2. The same VAE implementation is used for training a VAE on the FireEvac data. The 'mode' argument is set to mi and the 'dataset_dims' argument is set to 2, as the FireEvac dataset is 2 dimensional. A number of modifications have to be made to train the FireEvac dataset:

   - The VAE is trained for 1000 epochs.
   - The learning rate of the adam optimizer is set to 0.0001 because the higher learning rate of 0.001 used in the previous task proves to be too high for training the FireEvac data. It results in a jittery model training.
   - Mean Squared Error is used for calculating the reconstruction loss. The predicted data is the output of the decoder and the target is the input data.
   - Relu activation function is used for the last layer of decoder, as the FireEvac data consists of positive unbounded coordinates.

3. The distribution of the data reconstructed from test set, is similar to test set distribution. [10]

Figure 10: ELBO loss