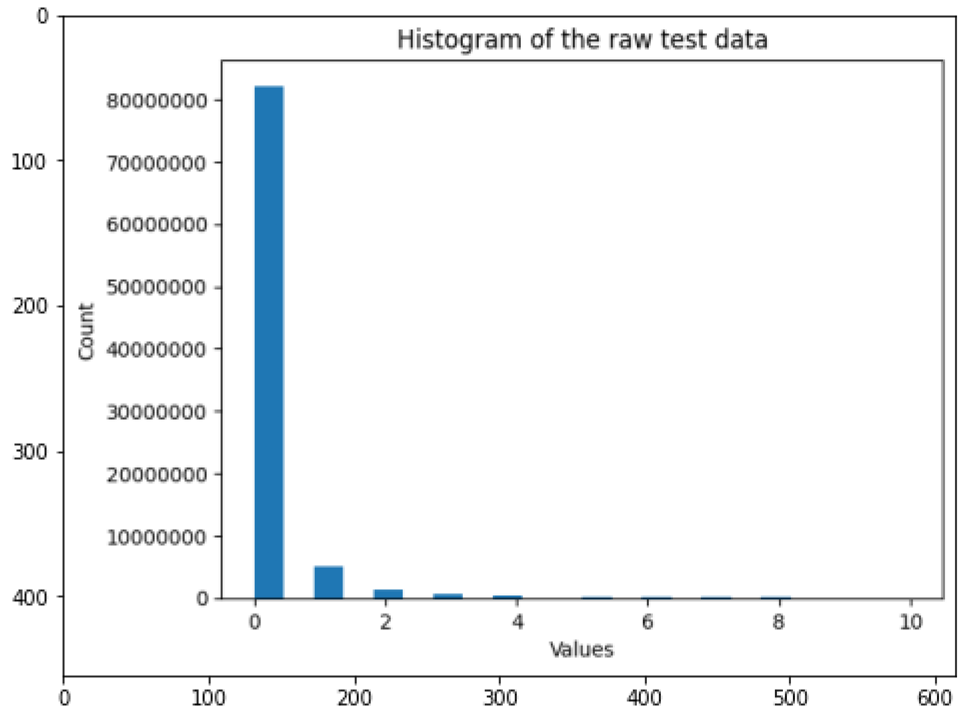
Natalia Rutecka

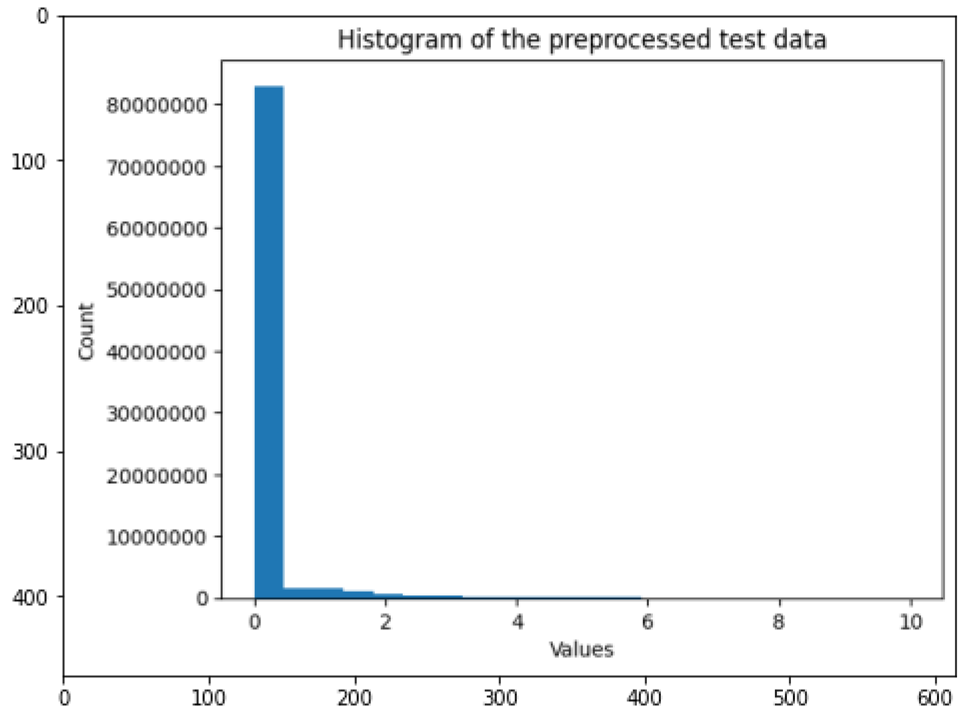
Report of project 1 (SDA 2 course)

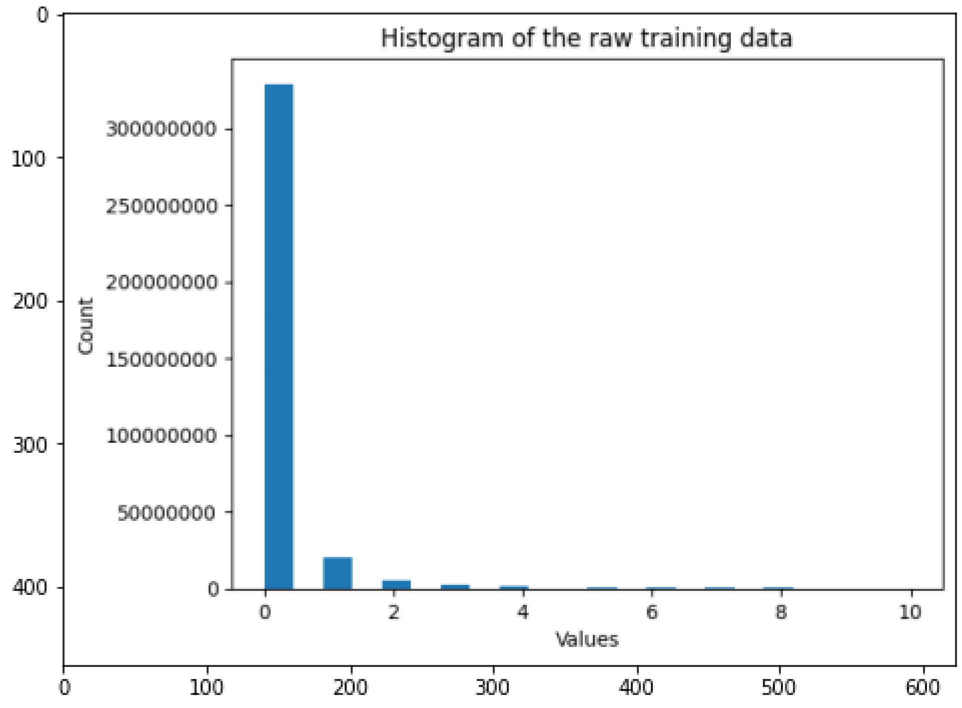
Part 1: Exploration

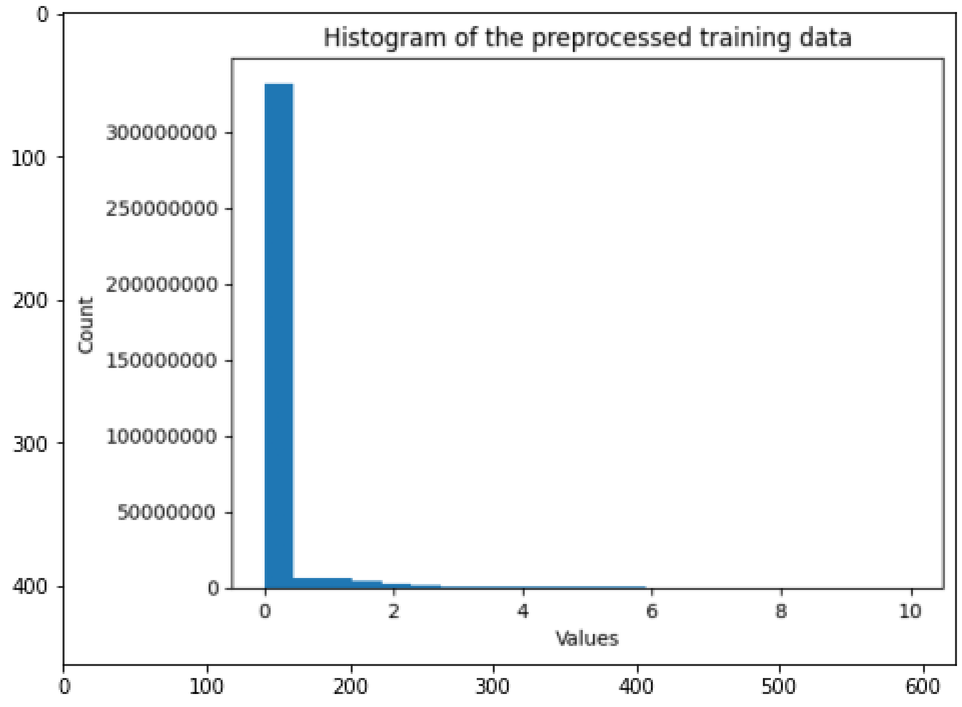
a) The number of variables - the genes whose expression is measured - is equal to 5000 in both training and test set. The training set and test set consist of 72208 and 18052 observations, respectively.

b) The histograms of raw and preprocessed data are depicted below. All histograms are clipped at max value 10 for readability.







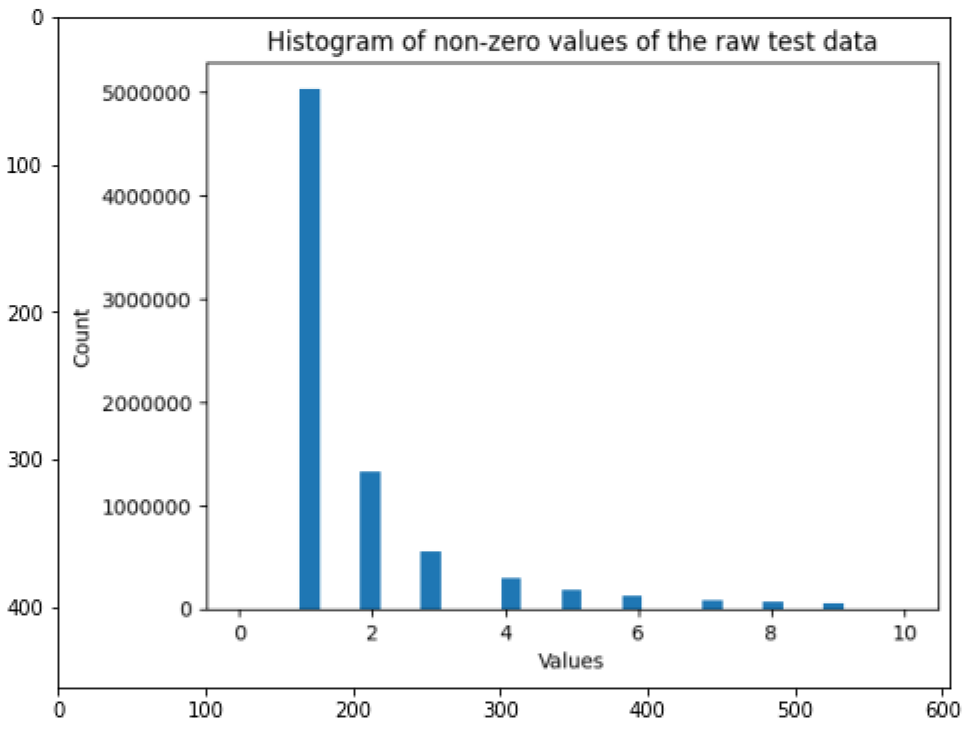


c) Traditionally, the scRNA-seq data is preprocessed by:

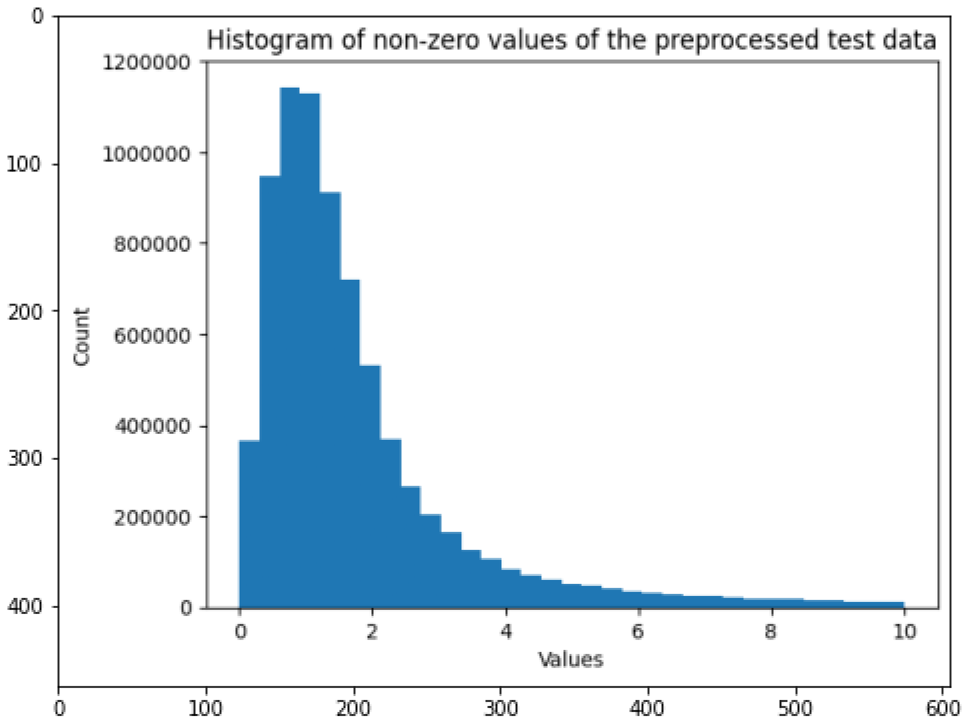
- normalising the data, so that for each observation (cell) the sum of its values is equal to 1000
- applying log1p to each value
- scaling the data to unit variance

Here, neither the preprocessed training or test data has unit variance (the variance is 2068692.875 for the training data and 5809936 for the test data). In both training and test dataset the max value increases after the preprocessing, which suggests that log1p transformation hasn't been applied. The sums of values also differ among observations, which means that the normalisation to any constant number of reads hasn't been performed. However, the preprocessing step includes a more complex normalisation - each cell is divided by a size factor stored in the data as "GEX size factor".

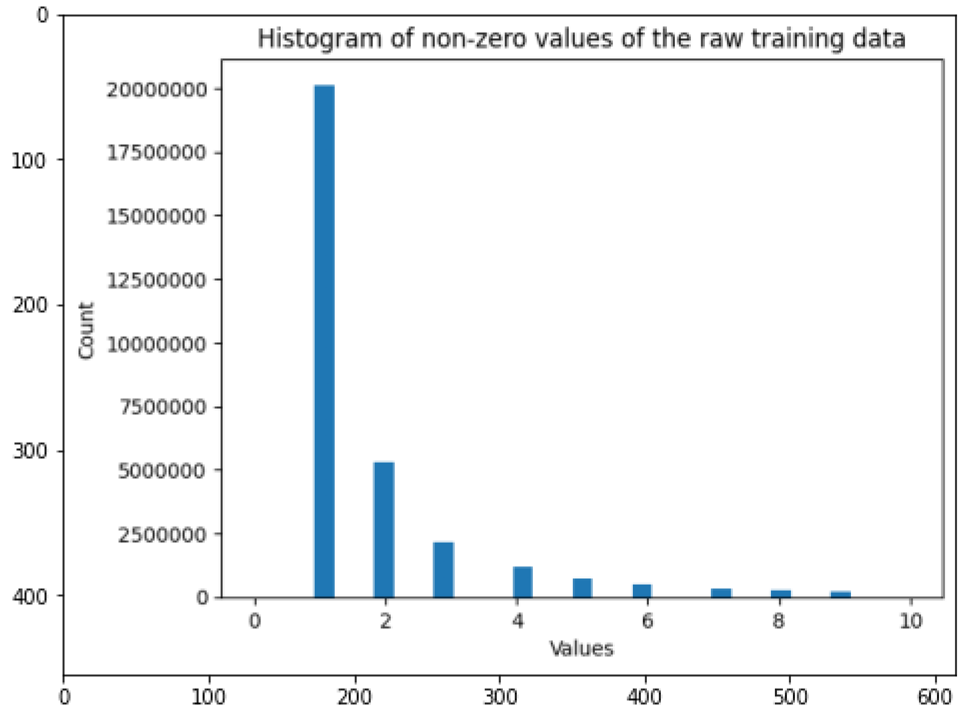
d) The histograms non-zero values in the raw and the preprocessed data are shown below. Again, the values larger than 10 have been clipped for readability.

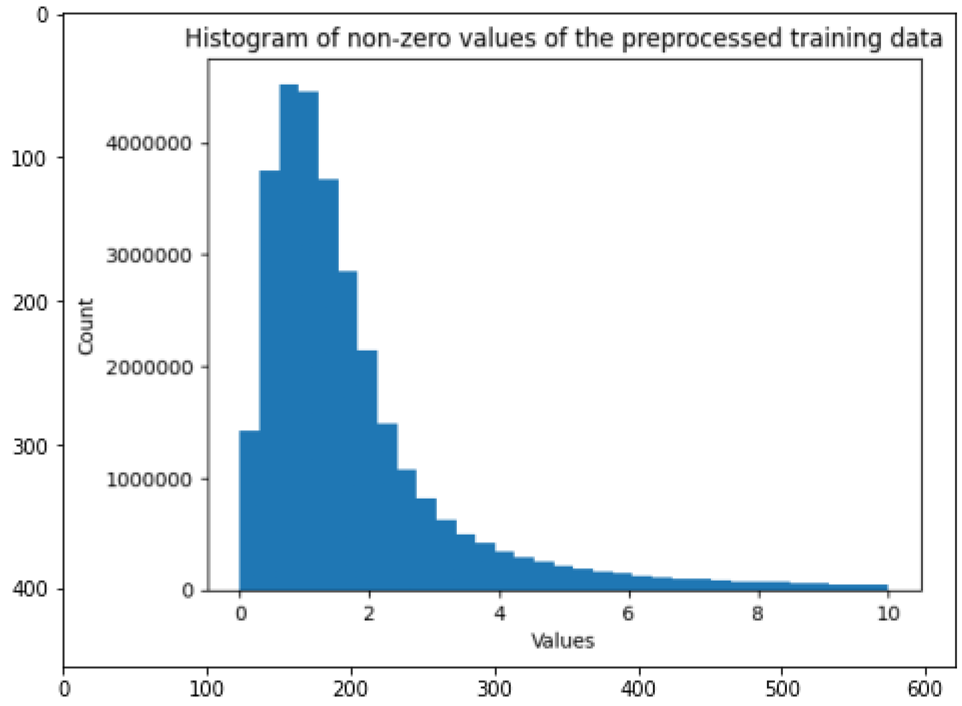


Histogram of non-zero values of the raw test data
Count
Values



Histogram of non-zero values of the preprocessed test data
Count
Values





(e) Both the raw and the preprocessed datasets contain vast numbers of zero values. In all cases the number of zeros is an order of magnitude larger than the number of any other value. This fact stems from the nature of the gene expression. Since transcription of a gene requires a certain region of a chromatin to be open, there is a limited number of genes that can be expressed at once in a given cell. Moreover, many genes are restricted in when and where they can be expressed and their gene expression is known to occur in so called transcriptional bursts.

The abundance of zeros paired with a very large variance in the data might make it hard to model it using common distributions. In case of the raw data, the most sensible option seems to be a negative binomial distribution, which is a discrete, two-parameter distribution that models the number of failures in a sequence of i.i.d. Bernoulli trials before a specified number of successes. When using a small probability of success, the distribution is able to produce data with arbitrarily high variance.

When it comes to the preprocessed data, the shape of the distribution seems to resemble the shape of the raw data with a higher resolution (meaning that it also contains non-integer values). In this case I would use a gamma distribution, which is a continuous, two-parameter distribution that produces similar shapes of distributions to the negative binomial. Since gamma distribution only has positive values, I would transform the preprocessed data by adding a small epsilon to its values.

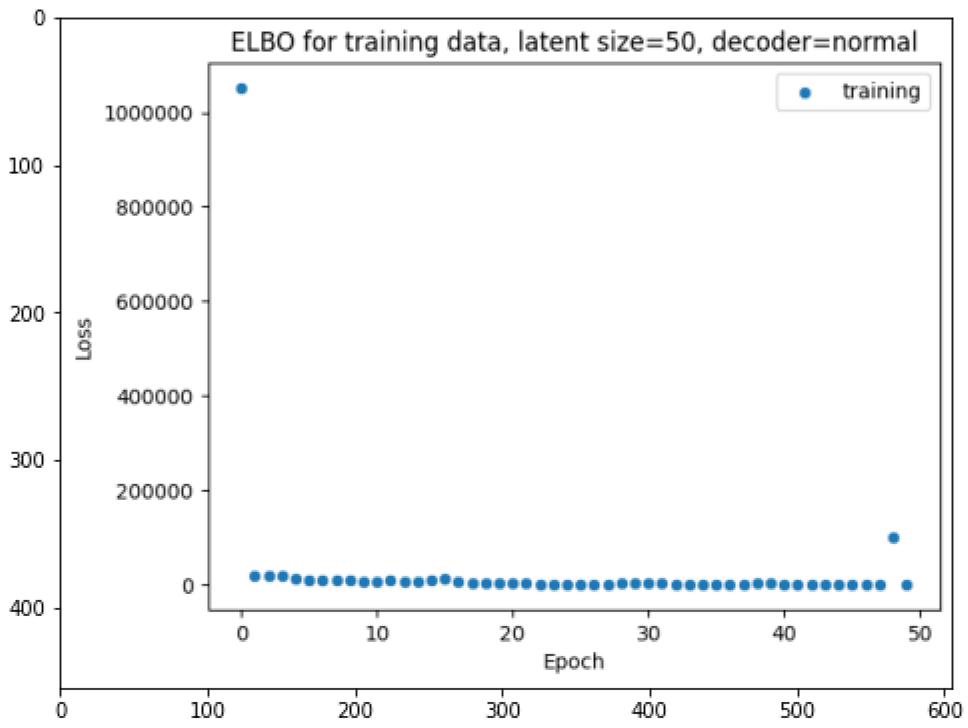
(f) The adata.obs object has 26 variables, providing summary statistics or additional information about the observations. It includes data connected to genes/cells (eg. cell type, cell cycle, percent of UMI counts mapped to mitochondrial genes) or cell donors (eg. age, BMI). The cells are of 45 cell types and come from 9 patients and 4 labs.

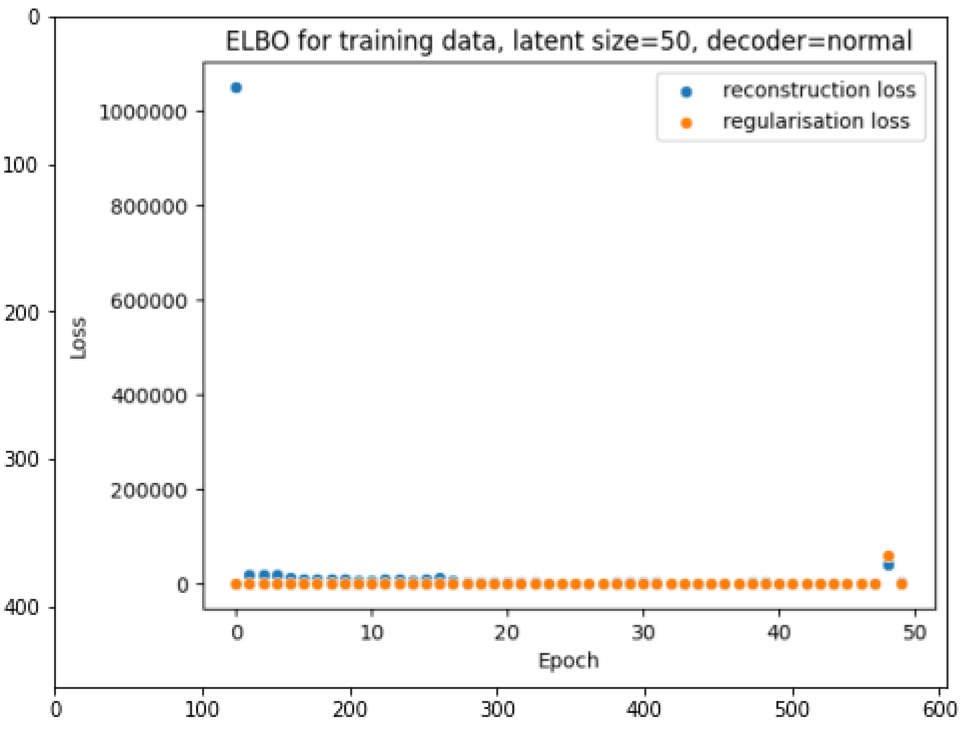
Part 2: Vanilla VAE

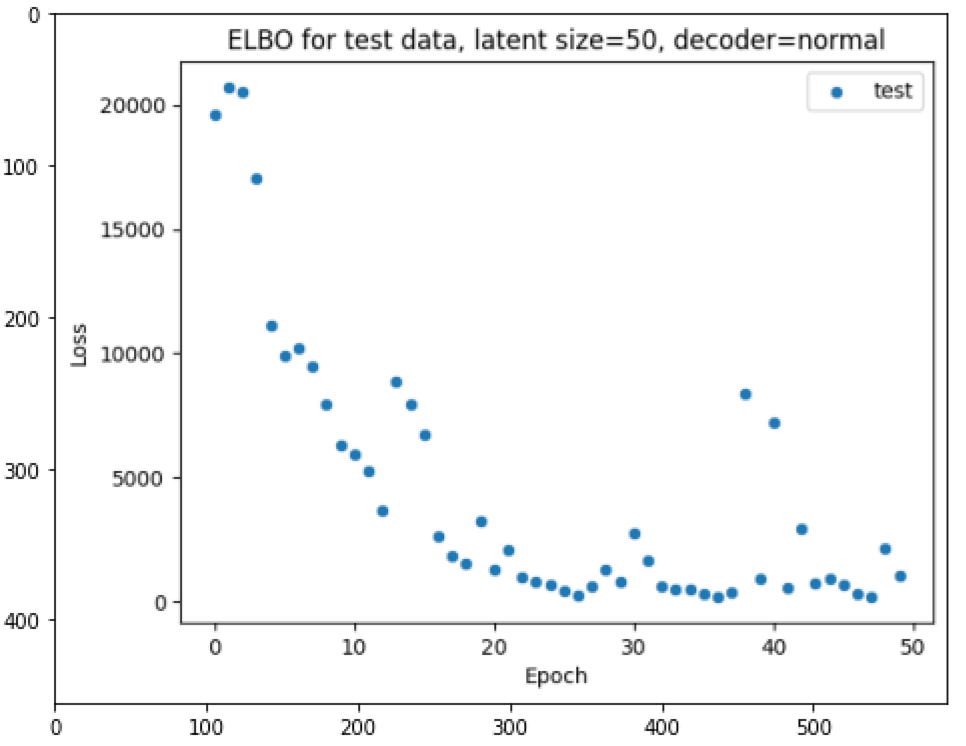
a) **Implementation details** The implemented VAE consists of a Gaussian encoder and a Gaussian decoder. Both the encoder and the decoder consist of linear layers. In each layer, except for the last ones, a sigmoid is applied to the calculated values of the neurons. The last layer of the encoder consists of k neurons encoding mean of the normal distribution and k neurons encoding logarithm of its variance, where k is a size of the latent space. The input to the decoder is sampled from the distribution encoded by the last layer of the encoder using the reparametrization trick. The last layer of the decoder consists of n neurons encoding mean of a normal distribution and n neurons encoding logarithm of its variance. Here, n denotes the size of the input (in case of the gene expression dataset n=5000).

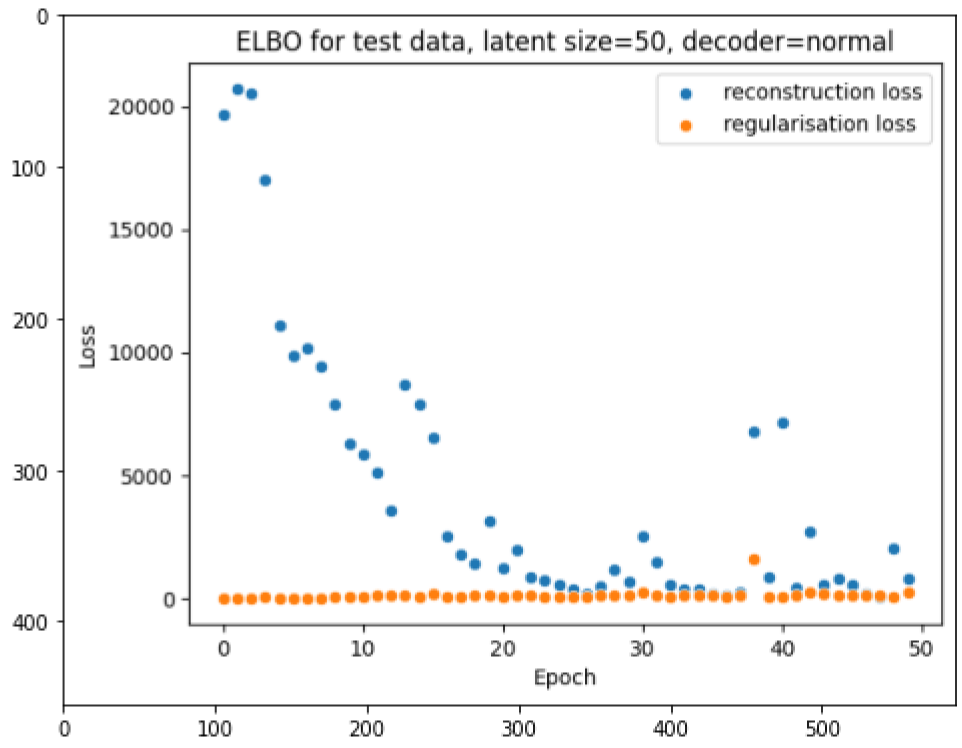
Used parameters During training and evaluation the preprocessed values (data.X) were used as the input. Both training and test datasets were divided into batches of size 64. The sizes of layers of the encoder and the decoder, including input and output layers, were equal to [5000, 500, 300, 50] and [50, 300, 500, 5000] respectively. Note that in case of the last layers, size s denotes the number of neurons encoding mean of the normal distribution, so the total number of neurons is actually equal to 2s (s for mean and s for log variance). The number of epochs during training was equal to 50 and the used optimiser was Adam with learning rate 0.003. The loss of a single observation X is defined as -logP + KL(R||N(0,1)), where logP is log prob of the distribuition encoded by the last layer of the decoder evaluated at the input X, KL is KL-divergence and R is the distribution encoded by the last layer of the encoder.

Learning curves In the figures below, reconstruction loss denotes -logP and regularisation loss denotes KL(R||N(0,1)). Loss for an epoch is defined as a mean of the batches' losses.




ELBO for training data, latent size=50, decoder=normal
reconstruction loss
regularisation loss
Loss
Epoch
1000000
800000
600000
400000
200000
0
0
10
20
30
40
50
0
100
200
300
400
500
600


ELBO for test data, latent size=50, decoder=normal
test
Loss
Epoch
20000
15000
10000
5000
0
0
10
20
30
40
50
0
100
200
300
400
500

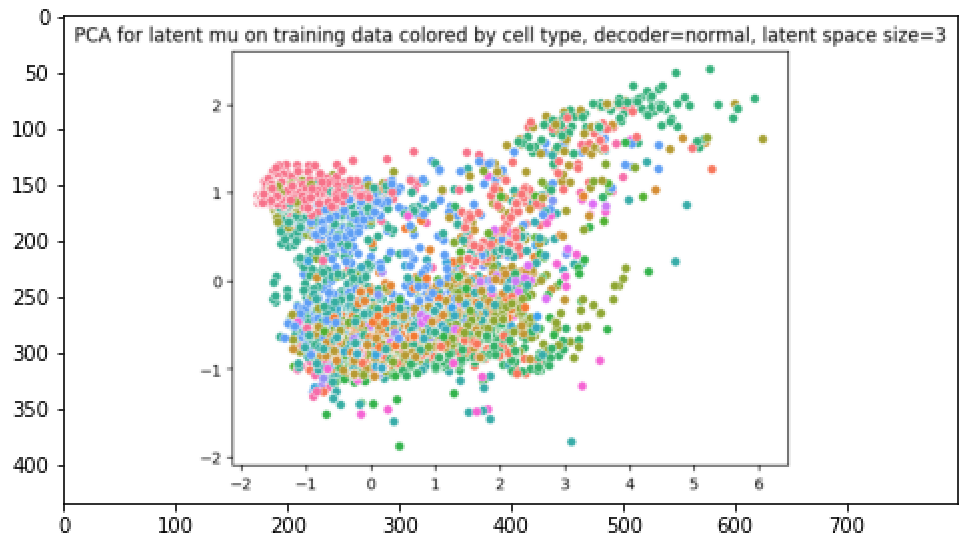


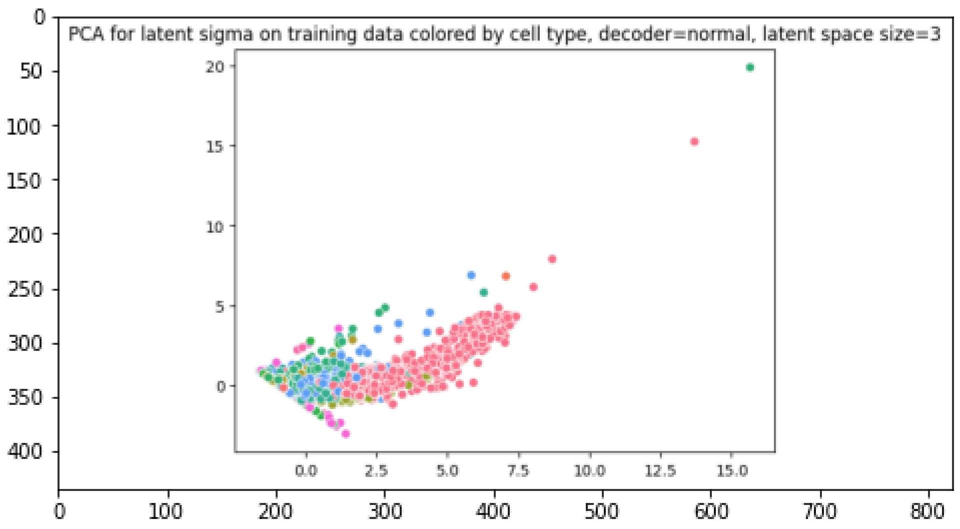
The training seems to be very unstable - the losses for some epochs are unexpectedly high. It is unclear whether it is a result of using normal distribution or some other problem with the network's parameters or implementation. Reconstruction losses are high on both training and test datasets, although they seem to slowly decrease with the number of epochs.

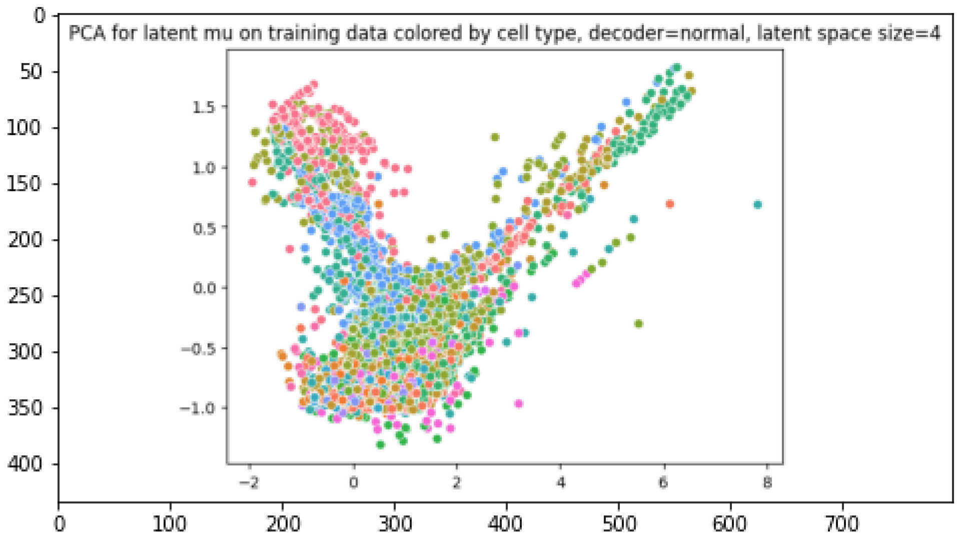
b) To complete this task, the values of the last layer of the encoder (mu and sigma vectors) were computed for the whole training set. The number of components for a given threshold was set to the maximum over mu and sigma of the number of components needed to explain the percent of variance determined by the threshold.

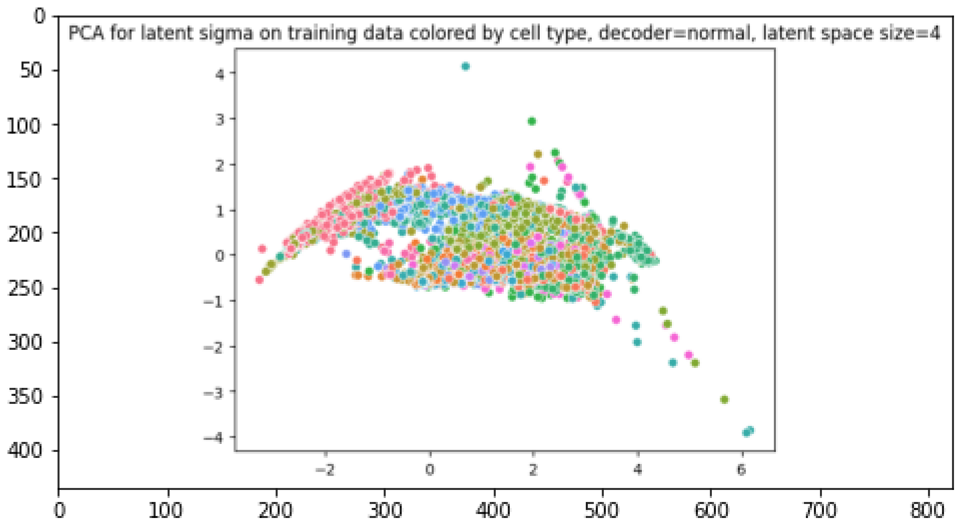
	explained variance	latent space size	-ELBO on last epoch
0	0.99	6	4746.590829
1	0.95	4	1882.237397
2	0.80	3	2619.268360

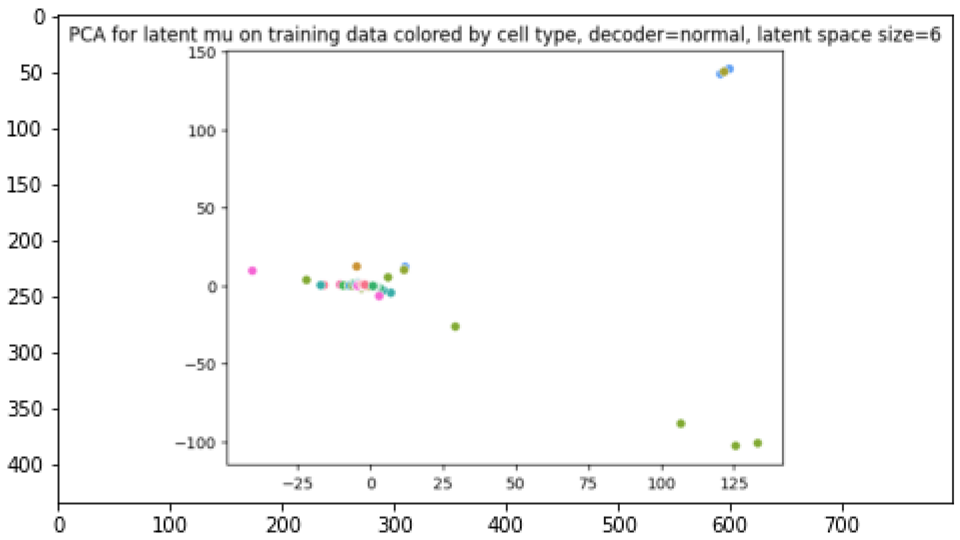
c) PCA plots:

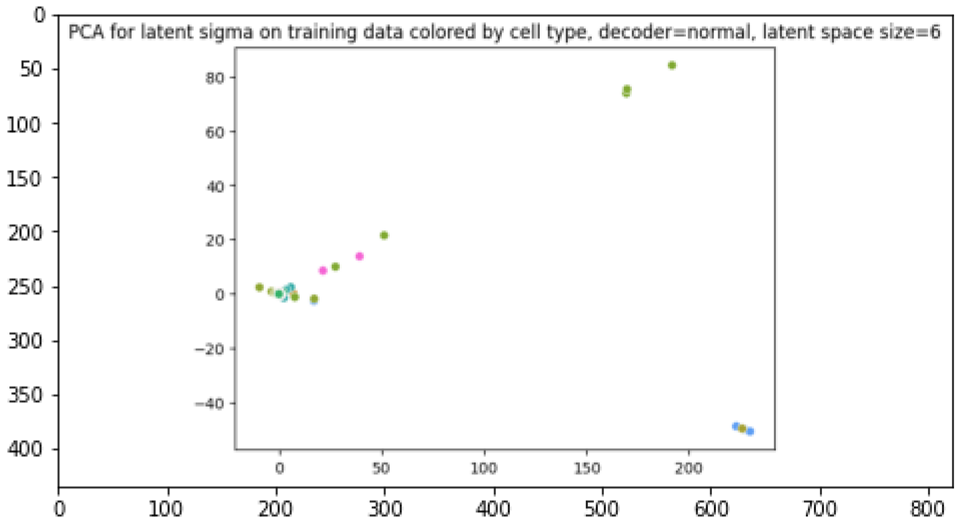


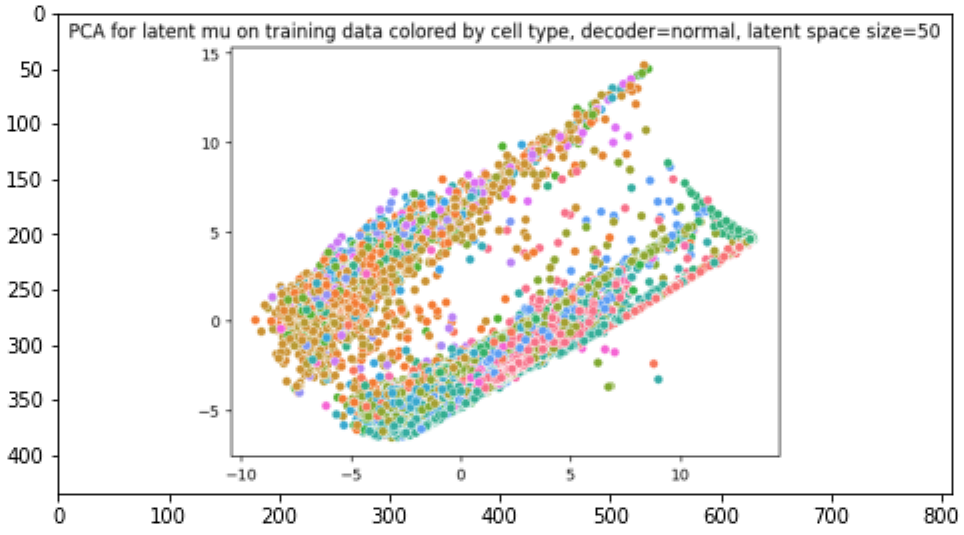

PCA for latent sigma on training data colored by cell type, decoder=normal, latent space size=3

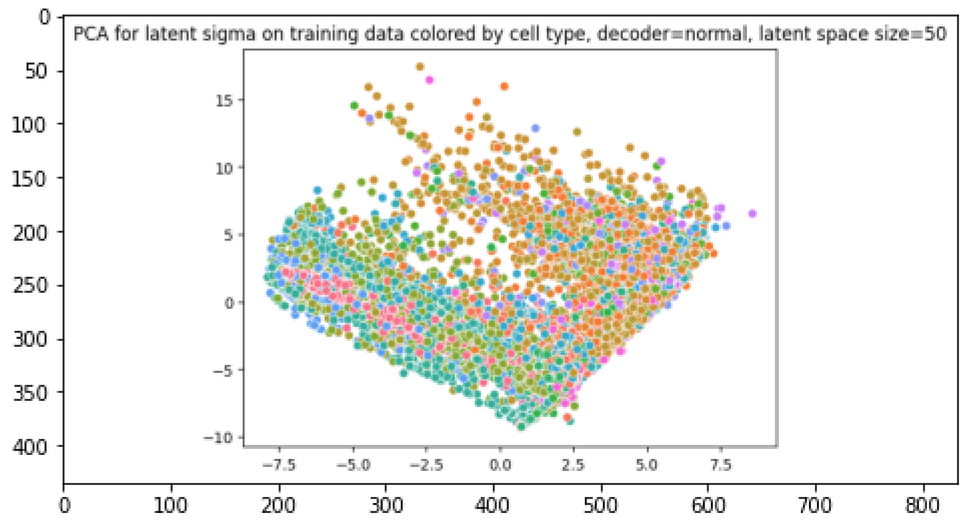

PCA for latent mu on training data colored by cell type, decoder=normal, latent space size=4


PCA for latent sigma on training data colored by cell type, decoder=normal, latent space size=4


PCA for latent mu on training data colored by cell type, decoder=normal, latent space size=6


PCA for latent sigma on training data colored by cell type, decoder=normal, latent space size=6


PCA for latent mu on training data colored by cell type, decoder=normal, latent space size=50



d) Since the lowest score was achieved for the latent space size equal to 4, this model was chosen as the final one. Usage of the preprocessed dataset data.X is motivated by the fact that it is normalized, which suggests that observations should be comparable to each other. This fact should in general make it easier to model the data using VAE, although some training problems can be observed for the VAE with gaussian encoder.

Part 3: VAE with gamma decoder

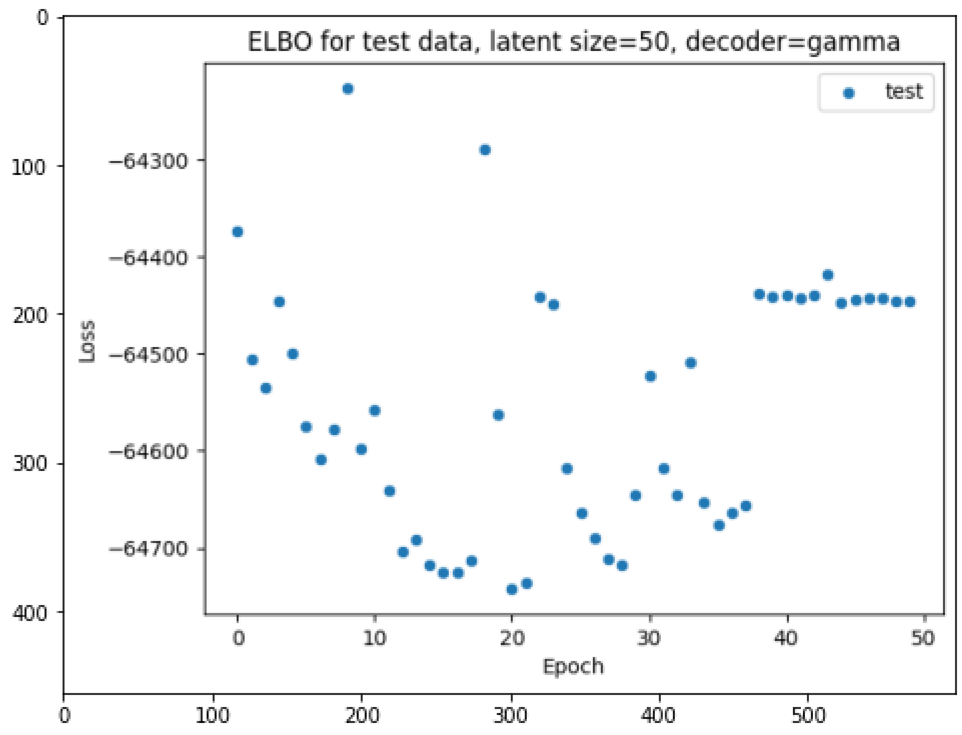
a) Following the explanations provided in Part 1 e) where I decribe the distribution of the data and in Part 2 d) where I mention the reasoning behind choosing the dataset, I implemented a VAE with gamma decoder and tested it on a preprocessed dataset.

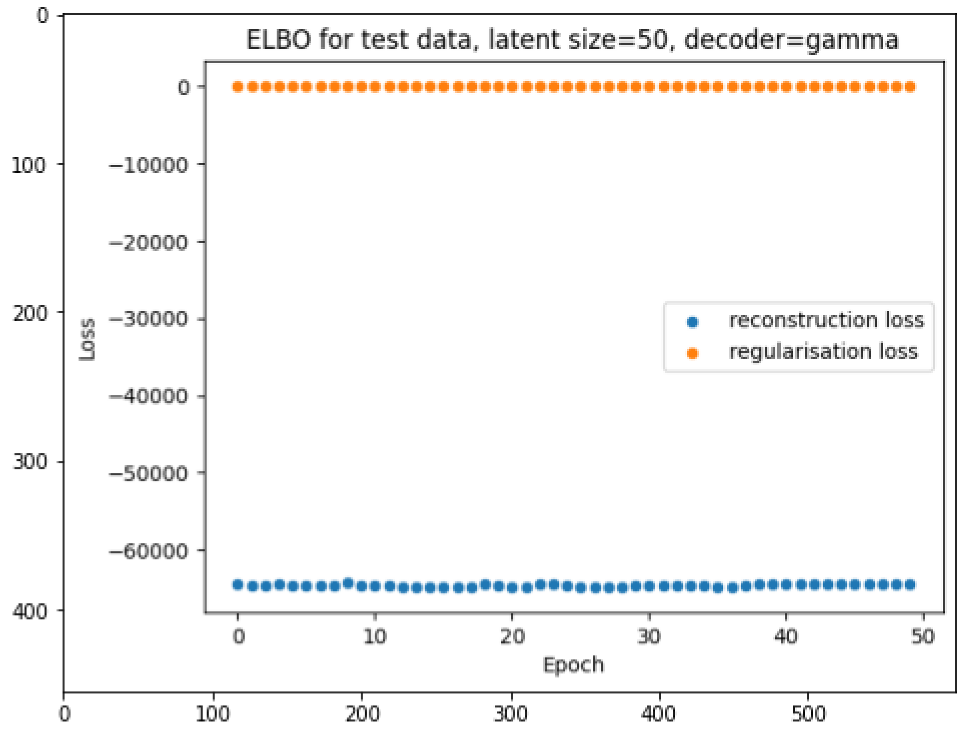
Here, the implementation was very similar to the one described previously for VAE with a gaussian decoder. The only difference was that here the last layer of the decoder consisted of n neurons encoding log of alpha parameter of a gamma distribution and n neurons encoding log of its beta parameter, where n is the size of the input=5000.

Again, I used batch size 64, number of epochs=50, Adam optimiser with learning rate=0.03 and sizes of the encoder and the decoder equal to [5000, 500, 300, 50] and [50, 300, 500, 5000] respectively.

The loss of a single observation X was again defined as -logP + KL(R||N(0,1)), where logP is log prob of the distribuition encoded by the last layer of the decoder evaluated at the input X, KL is KL-divergence and R is the distribution encoded by the last layer of the encoder. Since gamma distribution doesn't allow for non-positive values, logP was estimated by evaluating logP at the input with an added value of epsilon=1e-08 at each position.

b) The learning curves for the test dataset an latent size = 50 are depicted below.

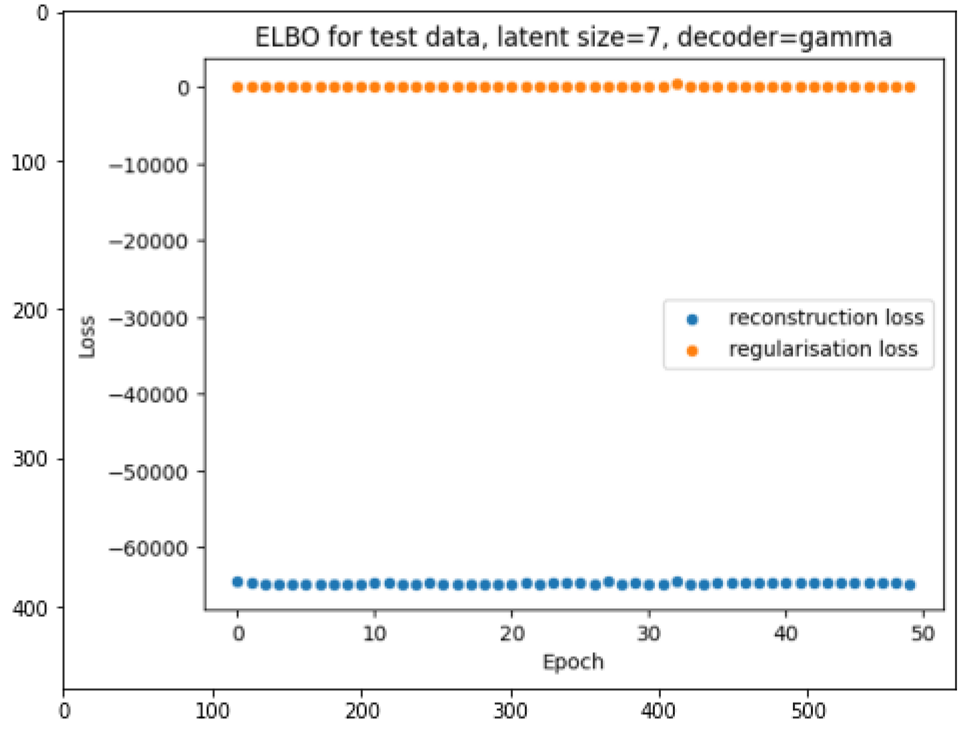


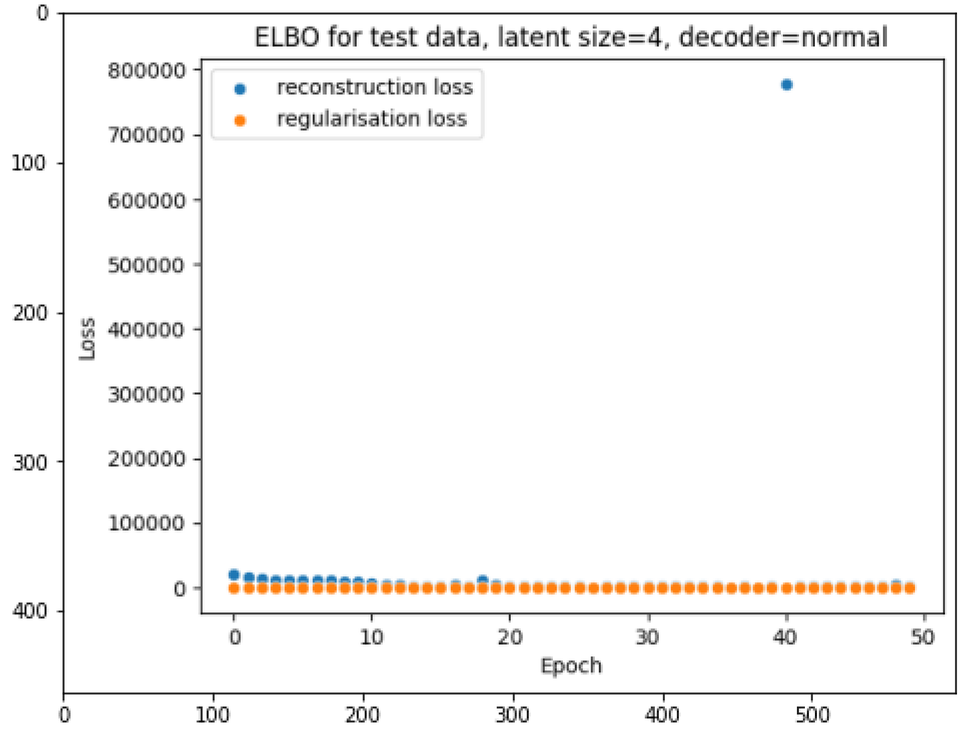


Again, the learning curve is irregular. However the values of losses are much lower compared to the values for VAE with a gaussian decoder. The losses are similar for different sizes of the latent space (see: table below), but the lowest one is achieved for the latent space of size 7.

	explained variance	latent space size	-ELBO on last epoch
0	0.99	7	-64691.820368
1	0.95	6	-64690.043788
2	0.80	3	-64635.703823

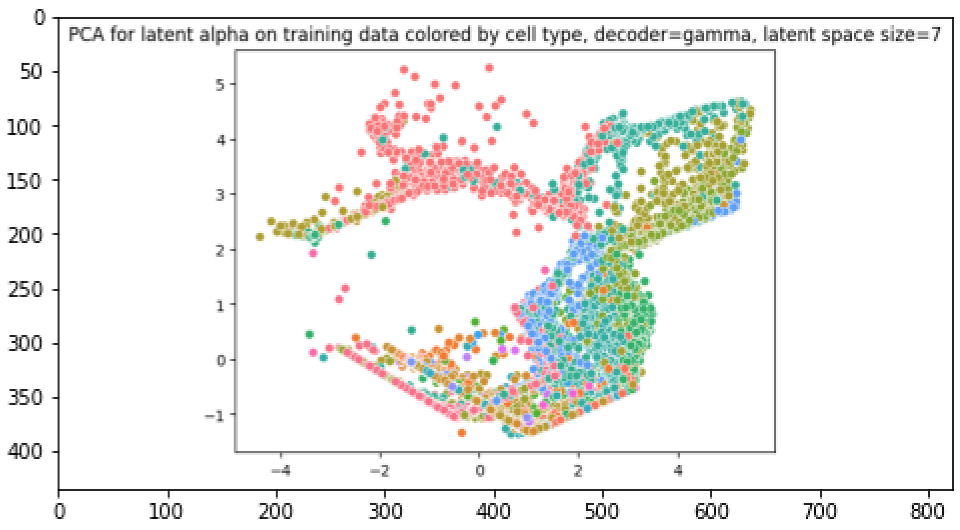
Below you can see the learning curves of the test data for the best models trained using gamma decoder (latent space size = 7) and normal decoder (latent space size = 4)

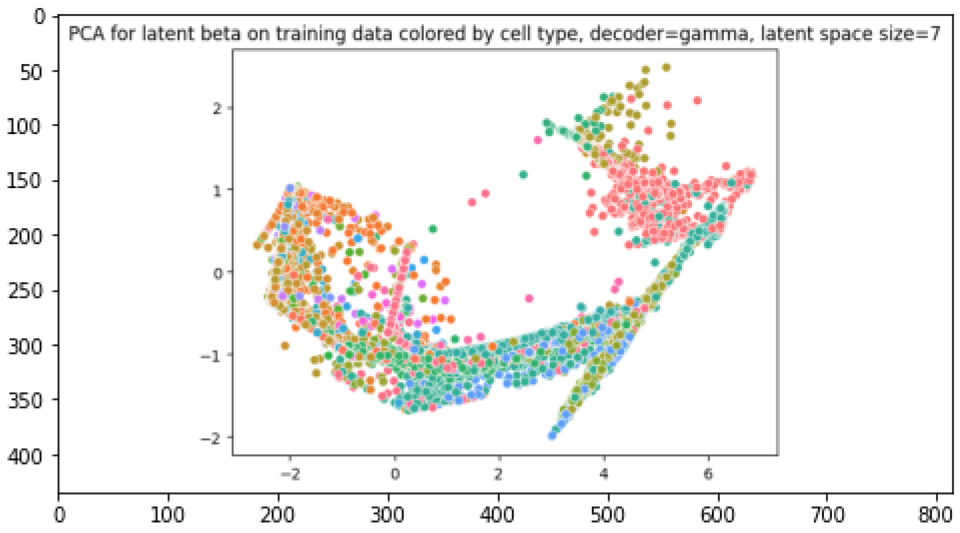


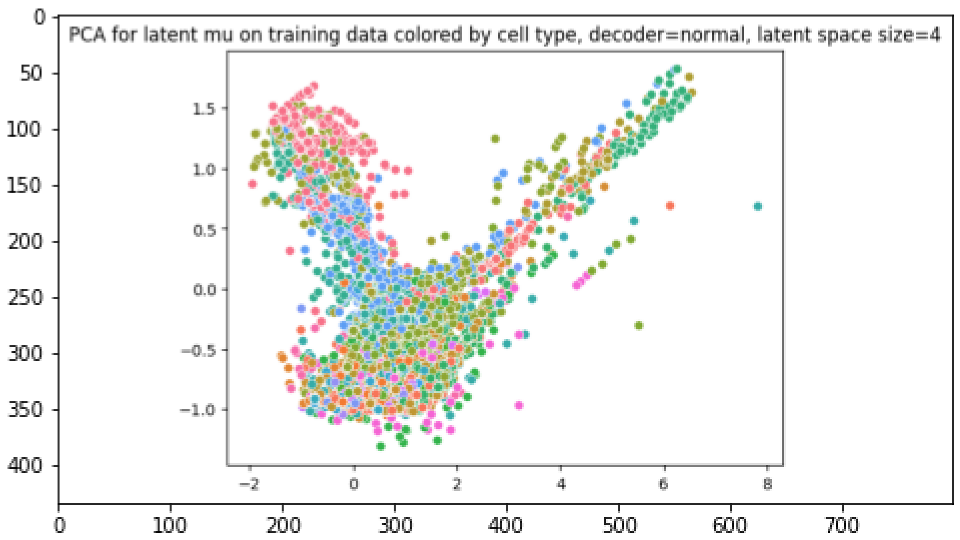


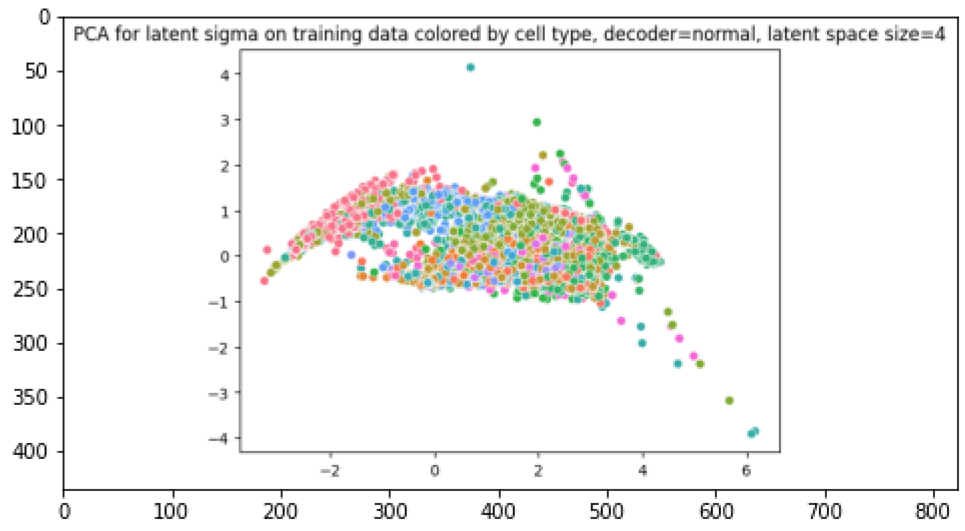
Clearly, the values of losses for the VAE with gamma decoder are significantly lower. Even though the learning curves look irregular in both cases, the gamma distribution seems to fit much closer to the data. To further improve the results one might try to modify the parameters of the network, which might lead to smoothening of the learning curves and finding even better local optima.

c) Latent spaces of the final models:


PCA for latent alpha on training data colored by cell type, decoder=gamma, latent space size=7


PCA for latent beta on training data colored by cell type, decoder=gamma, latent space size=7


PCA for latent mu on training data colored by cell type, decoder=normal, latent space size=4



In both latent spaces cell types are not clearly separated. The separation seems to be the most clear when it comes to the alpha parameter of the gamma distribution, meaning that the shape of the learned gamma distribution correlates with cell type of the input observation.