Denoising diffusion weighted magnetic resonance images using a variational autoencoder

Author: Julius Glaser
Supervisor: Dr. Zhengguo Tan

1 Introduction

Since the invention of Magnetic Resonance Imaging (MRI) back in 1970s [1, 2], tremendous development has been made and especially in the recent years deep learning algorithms boosted the performance of several MR-imaging techniques. One of these techniques is the diffusion weighted imaging (DWI) technique. Diffusion imaging is an important tool in the area of clinical neurology and with further advancements more and more important in neuroscience studies, due to its ability to deliver unique insight in neuroanatomy [3].

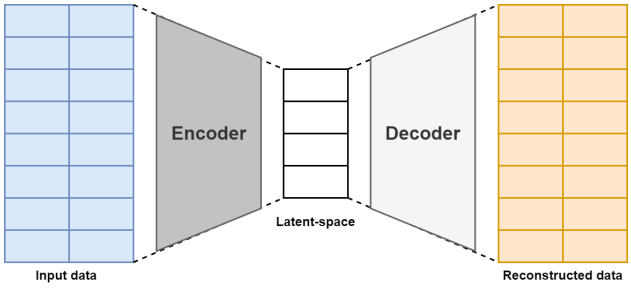
In DWI one samples the movement of the water molecules in the tissue. These samples resamble a sphere and with this samples one can reconstruct the eigenvalues and eigenvectors, which define the sphere and therefore knows in which direction the molecules prefer to move. which is equivalent to the direction of the fibres.

The downside of DWI is its tendency to have low signal to noise (SNR), due to the low signal that is generated by the diffusion movement of the water molecules. To denoise the resulting images several deep learning methods exist [4, 5, 6], but the architecture of a variational autoencoder (VAE) [7] was never tried out for this purpose. This was the main topic for this project, to train a VAE for denoising of diffusion weighted images and compare its performance to a standard denoising autoencoder (DAE), which was used in [5].

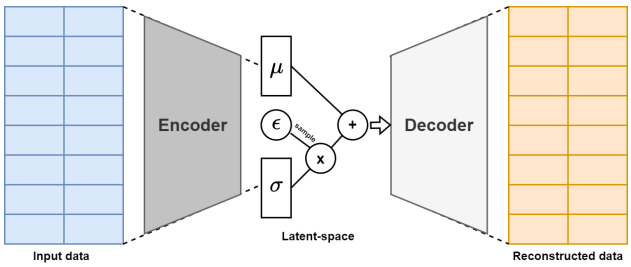
An autoencoder is a deep learning structure, which receives the input and scales it in dimension in the so called encoder part down with each layer until it reaches the so called latent-space or also called ”bottleneck”. The idea of this latent-space is that the network learns a representation of the input in a much smaller dimension and after the latent-space the dimension is layer for layer increased again to the input dimension. For a DAE the input is a image or signal with noise and the output should be the same image or signal but without the noise. The architecture of the DAE can be seen in figure 1 a).

For a VAE the only difference in the architecture is that the latent space does not contain typical neurons, but consists in way of probability density functions (PDF). For the decoder part a sample from each PDF is taken and fed into the decoder, which scales the dimension up again to the scale of the input. For training this architecture the so

called ”reparametarization trick” is used, because one can not backpropagate through the sampling operation and could not train the network like this. So this ”trick” means that one does not sample from the latent PDF itself but from some distribution, in general a Gaussian-distribution with a mean of 0 and a variance of 1 and just scale this sample by using equation 1 linear with mean and variance of the PDFs the latent-space resambles. The architecture of a VAE can be seen in figure 1 b).



(a) *Autoencoder Architecture*



(b) *Variational autoencoder architecture*

Figure 1: *Autoencoder and variational autoencoder architecture*

$$z = \mu + \sigma\epsilon \tag{1}$$

The difference in the probabilistic approach in VAE leads to the usage of the so called Kullbeck-Leibler divergence (KLD) in the loss function of the VAE. The KLD is a measure for the divergence of two distributions and can be seen as a kind of regularization of the network, to prevent it from overfitting, which in this regard would lead to peaky latent distributions with close to no variance and some mean. The KLD is calculated using equation 2, if one assumes Gaussian distributions it simplifies to 3 [7]. z are the latent variables and $q_\Phi(z)$ and $p_\Theta(z)$ are the distributions, $p_\Theta(z)$ is actually the real distribution one wants to estimate using $q_\Phi(z)$, the simpler distribution one uses. In 3 the σ and μ represent the standard deviation and mean of $q_\Phi(z)$ and J is the dimensionality of the latent space.

$$-D_{KL}((q_\Phi(z)||p_\Theta(z)) = \int q_\Theta(z)(logp_\Theta(z) - logq_\Theta(z))dz \tag{2}$$

$$-D_{KL}((q_\Phi(z)||p_\Theta(z)) = \frac{1}{2}\sum_{j=1}^{J}(1 + log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2) \tag{3}$$

The complete loss function of the VAE is equal to equation 4, so the loss that is back-propagated in the network is a combination of the reconstruction loss (the loss between input and output difference) and the KLD-loss (the loss between the difference in the distributions between the encoder and decoder stage). β is a scaling factor for the KLD.

$$L = -L_{reconstruction} + \beta L_{KLD} \tag{4}$$

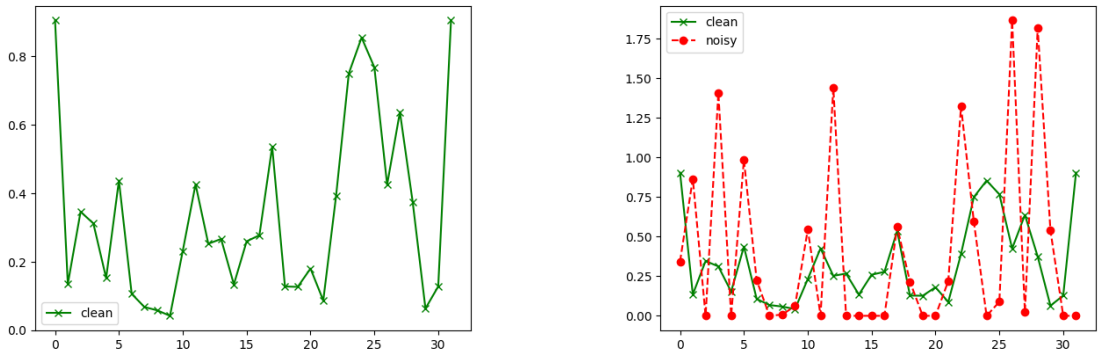
The network was already implemented in PyTorch and only slight adjustements were taken on side of the architecture. For the training an artificial diffusion data set was used and the network was at first only trained and tested on this data until a sufficable performance was achieved and then used on a set of real diffusion images and its performance evaluated and compared to the performance of a DAE on this data.

2 Methods

The neural networks are trained on simulated diffusion data, because there is no ground truth available for diffusion images, but for simulated data it is of course. The data is simulated using equation 5.

$$S_k = S_0 e^{-b\hat{g}_k^T \mathbf{D} \hat{g}_k} \tag{5}$$

In this equation the matrix $\mathbf{D}$ describes the shell one samples points from. In total 32 points are sampled from this shell in directions of $\hat{g}$ and with b-values of 1000 (besides the first and last value of these 32 values, which have a b-value of 50, the so called b0-values). The S_0-value is just equal to one and after discartion of irregular values one gets a data set of simulated data of the size of 428.522 and each of these entries has 32 values. A typical signal is shown in 2 a). To this signal then different gaussian noise distributions were added to create a noisy data set, therefore the standard-deviation of the gaussian noise were varied (constantly zero mean) and samples from this distribution were added to each sample of the signal and values below 0 were prohibited. The standard deviation of the distributions ranged from 0 (clean signal) to 0.91 (extremly noisy signal). A example for a noisy data sample which was used for testing after training is shown in 2 b), as a comparison the clean data is shown as well, which is the same as in figure a). One can clearly observe that no real visible structure is left in the data. This data set consisting of the noisy data was shuffeled and a split of 80% / 20% train and test data was used.



(a) *A clean data sample from the simulated data* ***(b)*** *A noisy data sample from the noised data*

Figure 2: *Sample examples for training*

The network architecture was already implemented in PyTorch and followed the following structure:

- Input size: 32
- Encoder: 3 fully connected layers with descending amount of neurons according to the latent space size
- Latent space: two parallel fully connected layers for mean and variance, size of parameter latent space
- Decoder: 3 fully connected layers with ascending amount of neurons according to the latent space size
- Output size: 32

The size of the latent space was varied between 7 and 10 for performance comparison. the DAE architecture, which was used to compare results, had the same structure as the VAE and a latent space of size 7 and was trained using the same data samples. The seed of the training was fixed to yield the same random results for every training for comparison between training runs.

The trained models were checked on their performance using MSE-loss between reconstruted and noisy samples in the simulated data and the original D-value was calculated from the reconstruction. After the training the networks were used in a real world test application using real noisy diffusion data and the performance was compared using the DAE and VAE for reconstruction, but because no groundtruth is available for this dataset, no clear metrics could be used.

The networks were trained for 40 epochs on an AMD Rome 7502, 2,5 GHz CPU for 12 hours. They were not trained on GPU, because for the small data dimension of 32 the operation of writing it over to the GPU and giving it back to CPU is not efficient such that it was faster to do the whole training on CPU. Different learning rates and optimizer were tried out. For the loss-function L1-, MSE- and Huber-loss were compared. The weight for the KLD was varied and in one case reconstruction without KLD-loss enterily tried out.

3 Results

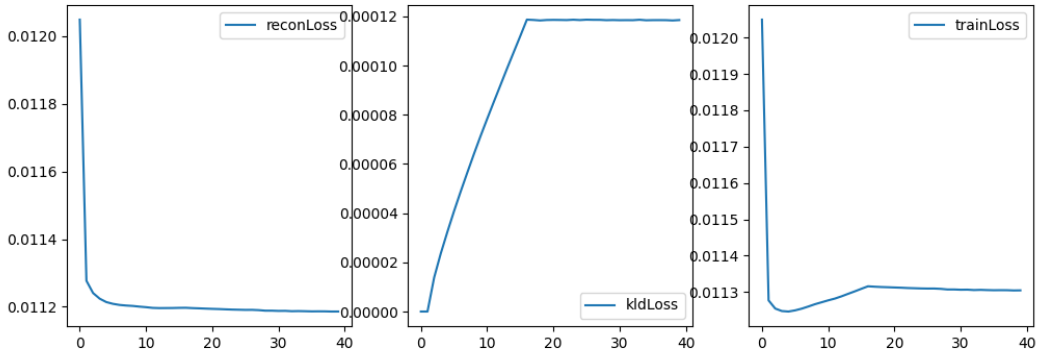
Using different setups the final network parameters, which yielded the best performance for the VAE and were also used mainly for the DAE are shown in table 1. The KLD loss weight was increased from 0 to the final value linearly, because it showed better performance and adaption to the data than just as starting it delayed to the training start.

Optimizer	ADAM
Reconstruction loss-function	Huber
Number latent variables	7
Learning rate	0.0004
KLD maximal weight	0.0015
KLD weight increase	0.0001 per epoch
KLD starting epoch	2
Epochs	40

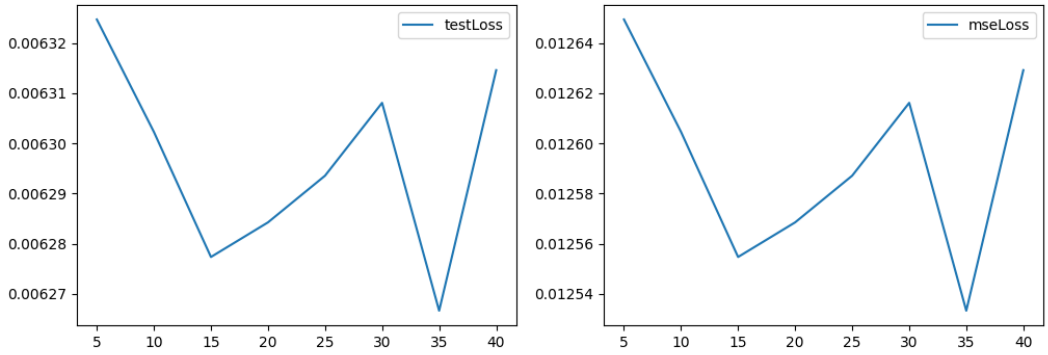
Table 1: *Network parameters*

The loss curves of the different loss values (reconstruction-, KL- and summed-loss) are plotted in figure 3 a) and the loss curves and a MSE-loss curve of testing every 5 epochs beginning at the 5th epoch are shown in b). After 40 epochs the loss in training seems converged. To yield a good comparison the MSE loss between the noisy simulated data and the reconstructed data was calculated and the final loss was 0.01262. The example reconstructed sample shown in 2 is shown in 4 a) and for comparison the results of the best fitted DAE are shown in b). As one can see, even with the high amount of noise the VAE is able to reconstruct the simulated data well, while the DAE is not able to do so.

The resulting D-values, which are the really interesting reconstruction, were evaluated using MSE and MAE and are shown in table 2. As one can see the VAE is performing better than the DAE, even though the data were not that noisy for the DAE compared to

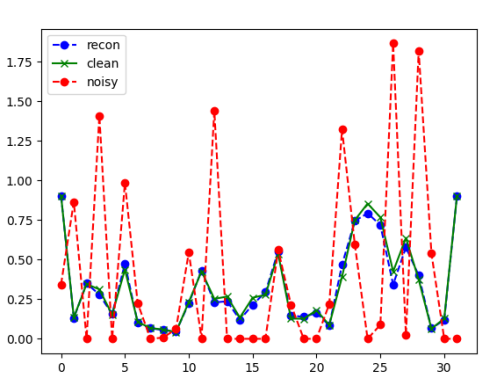


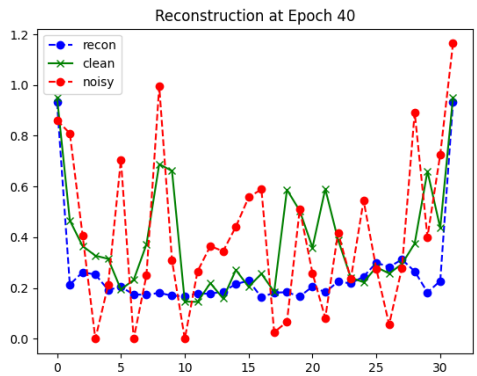
***(a)** Loss curves in training*



***(b)** Loss curves of testing*

***Figure 3:** Loss curves of VAE training and testing*





***(a)** Reconstruction in VAE* ***(b)** Reconstruction in DAE (different sample)*

***Figure 4:** Reconstruction results for simulated data samples for VAE and DAE*

the data of the VAE.

Clean vs.	MSE	MAE
Noisy VAE	2.48e-06	0.000956
Reconstructed VAE	2.85e-07	0.000412
Noisy DAE	1.74e-06	0.000840
Reconstructed DAE	4.03e-07	0.000534

Table 2: *Loss of D-values using DAE and VAE reconstruction compared to clean D*

Using the two auto-encoder models for a real in-vivo data set lead to the images shown in figure 5. There a certain slice with a certain b-value is shown. To increase the visibility of the different reconstructions the difference image of VAE and DAE reconsturction is plotted as well. One can see a clear difference in the reconstruction of the VAE and DAE. The DAE is getting rid of a lot of noise but the structure of the diffusion weighting is also lost, which is preserved in some way in the VAE reconstruction. If one takes a look at the histogram of the difference images of reconstruction and original, which are shown in figure 6 one can see that the DAE results are differing in some random way, while the VAE results at least resambles a bit gaussian noise distribution. Not all slices were considered in the histogram plotting because for the neck-region the DTI results are in general of poor quality.

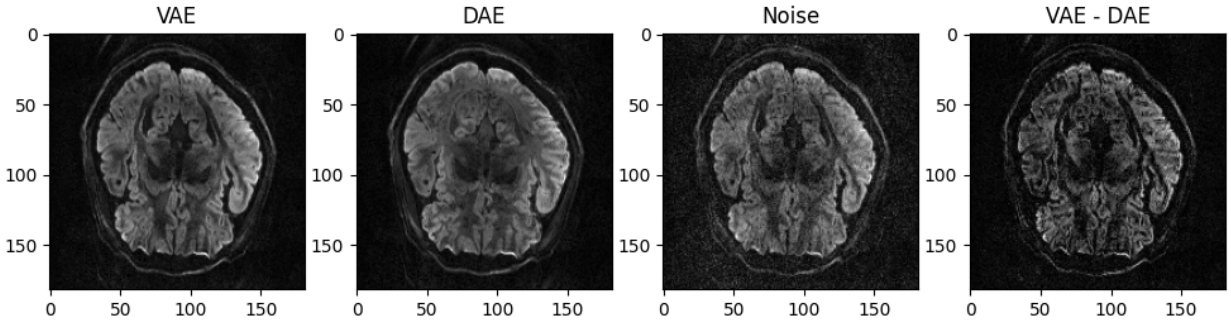


Figure 5: *Reconstruction of noisy in-vivo data using VAE and DAE*

For another amount of latent variables the results of the VAE are compared. The latent variable gaussian resamblance after training for 7 and 10 latent variables are plotted in figure 7 a) and b). One can see that any additional latent variable on top of the 7 learns the same distribution with a high variance and some mean, which can be seen as complete random distribution with little impact. For the results in the in-vivo test case the contributions of the latent variables to the reconstruction are shown in figure 8 a) and b). The additional latent variables have no impact on the reconstruction as clear to see.

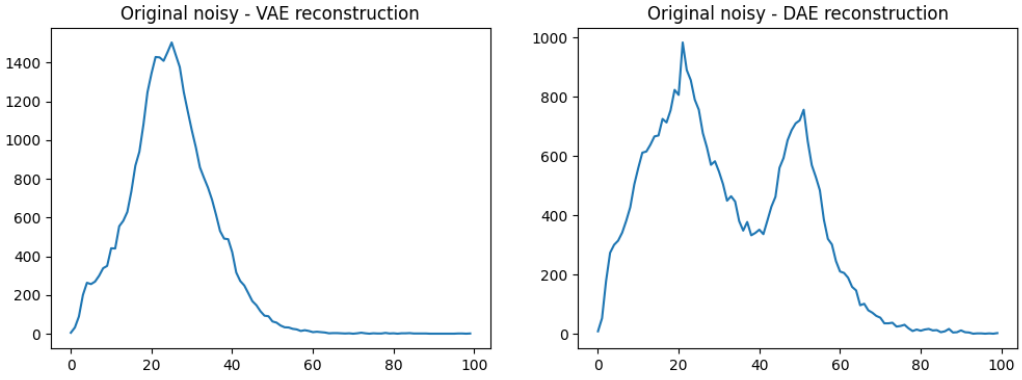
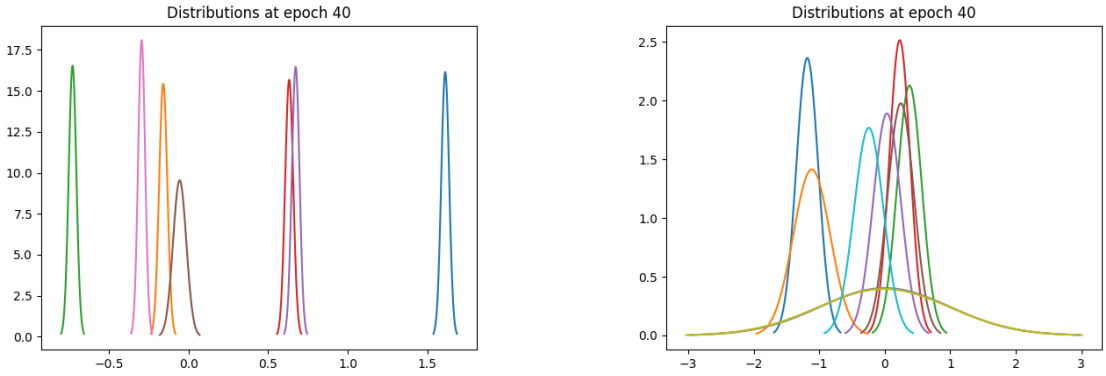
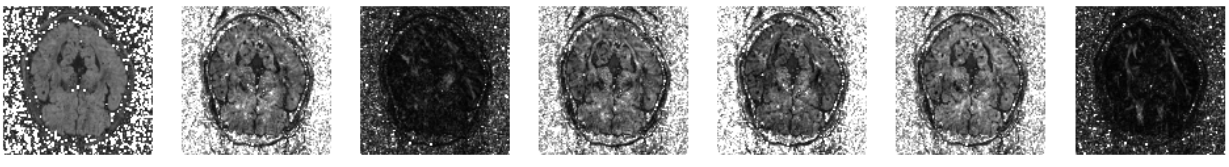


Figure 6: *Histogram of the reconstructions over the slices 22-94 and all b-values*

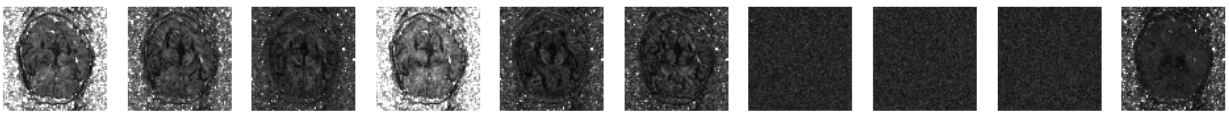


(a) *7 latent variables* ***(b)*** *10 latent variables*

Figure 7: *Probability distributions of the latent variables*



(a) *7 latent variables*



(b) *10 latent variables*

Figure 8: *Influence of the latent variables for the reconstruction of noisy image data*

For the case of training without the KLD-loss the results for the simulated data seem close to the results of the training with KLD-loss (9 b)), but needs a longer time to converge in the training (a)) and the resulting distributions are really peaky and have close to no variance as one can see in figure 9 c). The results in the simulated data look as good as for the VAE as shown in part d) and the results for the in-vivo data, which can be seen in e) are looking similar as for the VAE with KLD.

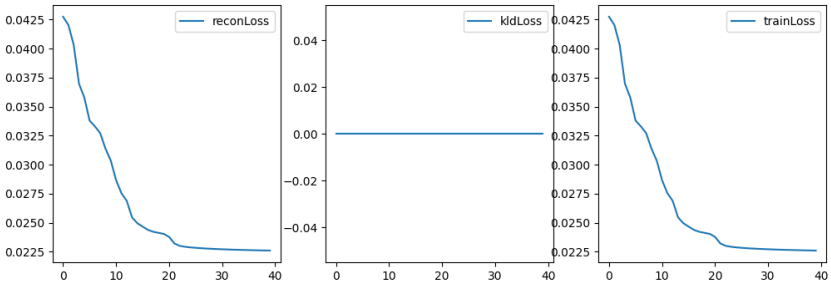
4 Discussion

As one could see in the previous chapter the results for the VAE reconstruction look very promising especially compared to the used DAE. Why the DAE is performing so bad is definitly a question to ask and something to investigate in the future, maybe is the random used training set just not preferable for the DAE and one should ensure using the same test and training set for all variations, because the data was shuffeled differently for the DAE set. A latent size of 7 seems the most preferable for the use case. The results of using VAE without the KLD show no real benefit in using the regularization, but faster convergence and slightly better train and test results. Further investigation is definitely useful at this point. Some kind of metric for the reconstruction of the in-vivo data could be useful but hard to find. The architecture of the auto-encoder was not differed in any way and for example the usage of convolutional layers could be of interest.

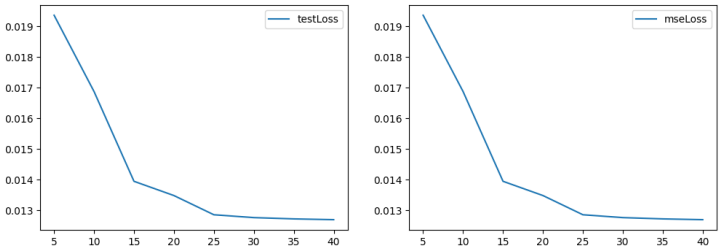
For a future project the target data could be changed as well and instead of using the 32 input-size model a multy shell model with input-size of 128 is of interest. And the data is considering only 6 parameters for the estimated shell, which specifically in a situation of crossing fibres leads to a problem in the diffusion estimation. Therefore more parameters than 6 should be considered and the ”ball-and-stick” model could be implemented. And maybe gaussian noise is not the actual noise distribution which is present in diffusion MRI, so the simulated data should consider other distributions like the Rician-distribution.

5 Conclusion

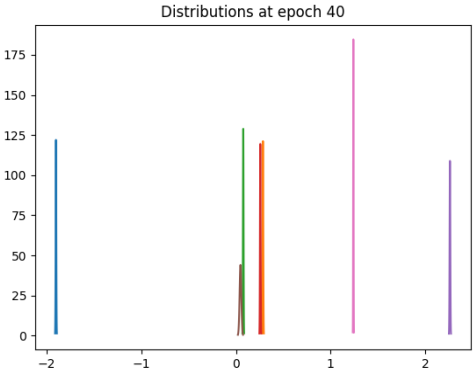
To give a short summary of the findings of the project the VAE performance for the denoising task is good, but further investigation is needed and a better performance comparison and evaluation is needed and the simulated data itself needs to be improved.



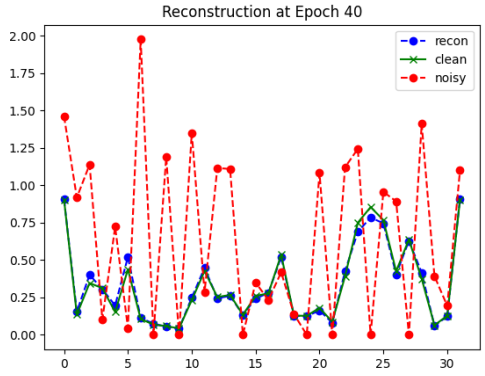
(a) *Train-loss over the epochs*



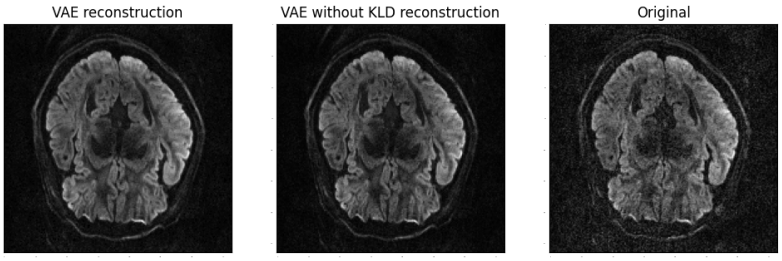
(b) *Test- and MSE-loss over the epochs*



(c) *Latent variables*



(d) *Latent variables*



(e) *Reconstruction comparison*

Figure 9: *Results for the simulated data for VAE training without KLD*

References

[1] P C Lauterbur. Image formation by induced local interactions: Examples employing nuclear magnetic resonance. *Nature*, 242:190–191, 1973. doi:10.1038/242190a0.

[2] Peter Mansfield. Multi-planar image formation using NMR spin echoes. *J Phys C*, 10:55–58, 1977. doi:10.1088/0022-3719/10/3/004.

[3] Sharma R Gupta AK Baliyan V, Das CJ. Diffusion weighted imaging: Technique and applications. *World J Radiol*, pages 785–798, 2016. doi:10.4329/wjr.v8.i9.785.

[4] Christiaens D Ades-Aron B Sijbers J Fieremans E. Veraart J, Novikov DS. Denoising of diffusion mri using random matrix theory. *Neuroimage*, pages 394–406, 2016. doi:10.1016/j.neuroimage.2016.08.016.

[5] Merry Mani, Vincent A. Magnotta, and Mathews Jacob. qmodel: A plug-and-play model-based reconstruction for highly accelerated multi-shot diffusion mri using learned priors. *Magnetic Resonance in Medicine*, 86(2):835–851, 2021. doi:10.1002/mrm.28756.

[6] Jakub Jurek, Andrzej Materka, Kamil Ludwisiak, Agata Majos, Kamil Gorczewski, Kamil Cepuch, and Agata Zawadzka. Supervised denoising of diffusion-weighted magnetic resonance images using a convolutional neural network and transfer learning. *Biocybernetics and Biomedical Engineering*, 43(1):206–232, 2023. doi:10.1016/j.bbe.2022.12.006.

[7] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. arXiv:1312.6114, doi:10.48550/arXiv.1312.6114.