DD2434 - Machine Learning, Advanced Course
Group 31

Learning Useful Representation of Data Through Variational Inference

Maxellende Julienne
maxjul@kth.se

Erik Hamberg
erikhamb@kth.se

Harry Vuong
hvuong@kth.se

Abstract

This project aims to recreate the variational autoencoders as presented in the paper *Auto-Encoding Variational Bayes* (2013, P. Kingma & Welling) [5]. By changing the loss function of a vanilla auto-encoder to add a condition on the density of the latent space, this new architecture can generate new images with the decoder part. In addition we implemented vanilla autoencoder to compare results to the variational autoencoder's. The success of this project was measured by the negative variational lower bound (ELBO) and the quality of the generated samples from the variational autoencoder. All in all the implemented variational autoencoder seemed to produce comparable negative ELBO despite not being exact and the generated samples were comparable to the paper in terms of quality.

Link to our GitHub repository: https://gits-15.sys.kth.se/erikhamb/MLA_Project

1 Introduction

In this project, we aim to re-implement the paper *Auto-Encoding Variational Bayes* (2013, P.Kingma & Welling) [5]. The new architecture proposed is an amelioration of the vanilla auto-encoder to learn a useful representation of data: the latent space is forced to follow a certain distribution (here gaussian). Let's introduce the three main notions used in the paper to create this new architecture.

Variational encoder A variational autoencoder (VAE) learns a compact, lower-dimensional representation of a dataset in a way that can capture useful underlying structure of the data. As an auto-encoder, it consists of two parts: an encoder, which maps the input data to a lower-dimensional representation, and a decoder, which maps the lower-dimensional representation back to the original data space. In a VAE, the lower representation $p(z)$ is encouraged to follow a simple distribution, which are then used to generate a new data point from the latent space using the decoder $p(x|z)$. To do so, the VAE is trained to maximize the likelihood $p(x) = p(x|z)p(z)$ of the data under the model, which leads to the learning of a compact, continuous latent representation that can be used to generate new data points the latent space. VAEs are thus probabilistic models as they can generate new samples, whereas autoencoders are deterministic and can only reconstruct the input data.

Re-parametrization trick It is generally not possible to directly sample from the latent space, because the latent variables are typically correlated with the input data and are not independent. To overcome this issue, VAEs use a technique called "re-parameterization" to allow sampling from the latent space. Here the latent variables are assumed to be Gaussian distributed with mean μ and standard deviation σ, the re-parameterization trick can be used to sample from the latent space by sampling a noise variable ϵ from a standard normal distribution and adding it to the mean:

$$z = \mu + \sigma \odot \epsilon \text{ with } \epsilon \sim \mathcal{N}$$

This allows the VAE to sample from the latent space using independent noise variables, rather than the correlated latent variables, which makes it easier to learn a compact, continuous latent representation that can be used to generate new data points.

Loss function and variational bound An auto-encoder can be trained by minimizing the reconstruction error as presented in the paper *Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion* [9]. However, to learn useful latent representations, we can add a regularization term on the latent space distribution using variational inference as presented in the paper *Representation Learning: A Review and New Perspectives* [2]. Indeed, a regularization term on the latent distribution $p(z)$ encourages it to follow a certain chosen distribution $q(z|x)$. The paper uses variational bayesian inference to maximize the likelihood, using the concepts explained above and the results of the paper *Variational Bayesian Inference with Stochastic Search* [6]. This paper shows that we can maximize the likelihood by maximizing a variational bound.
Here are the mathematical justifications. Using variational inference, the log-likelihood for a sample can be rewritten as follow :

$$\log p_\theta(x^{(i)}) = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right] + \mathbb{E}_{q_\phi(z|x)}[-\log q_\phi(z|x) + \log p_\theta(x,z)]$$

where $p_\theta(x|z)$ is the likelihood of the data given the latent representation, $q_\phi(z|x)$ is the approximate posterior distribution over the latent variables given the data, and $p_\theta(z)$ is the prior distribution over the latent variables.
As the first term is positive (we recognise a KL divergence), we can keep only the last term and transform the equality in a superior equality. The second term is called the Evidence Lower Bound (ELBO).
We rewrite the ELBO as:

$$\text{ELBO} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}[q_\phi(z|x)||p_\theta(z)] \tag{1}$$

The first term, $\mathbb{E}_{q_\phi(z|x)}[\log p(x|z)]$, measures the reconstruction error of the VAE, and encourages the decoder to reconstruct the input data accurately. The second term, $D_{KL}[q_\phi(z|x)||p_\theta(z)]$, measures the complexity of the latent representation and encourages the latent representation to be as simple as possible, i.e to follow the simple distribution wanted.

Our project During this project, we implemented a VAE and trained it using the MNIST data set by minimizing the ELBO as described above, with $q_\phi(z|x) \sim \mathcal{N}(0, I)$, the approximate posterior distribution over the latent variables given the data. The algorithm used to learn is the *auto-encoder variatonal bound (AEVB)*, derived from the stochastic gradient descent (SGVB) estimator presented in the paper *Stochastic Backpropagation and Approximate Inference in Deep Generative Models* [7]. The success of this implementation will be measured by how close our results are to the results of the negative ELBO for different latent spaces in the paper. The marginal log likelihood $\log p(x)$ will not be treated in this project. Additionally, we will make comparisons of the loss and generated samples from VAE and a vanilla autoencoder (AE).

2 Method

Architecture Two models were implemented, variational autoencoder (VAE) and vanilla autoencoder (AE). Both use MLPs (multi-layer perceptrons) with linear layers as their components, tanh as the activation function on the output of the hidden layers, and sigmoid activation on the output of the decoder. Only one hidden layer is implemented in both models. For more information on the architecture we refer to appendix A. The encoder and decoder of VAE follow equation 12 on page 11 as presented in [5]. We used a gaussian decoder since the MNIST is continuous data. For a concise explanation of the workflow of VAE, we refer to appendix A.3

Data The MNIST data was preprocessed to a continuous range $[0, 1]$ with a size of 70k, where 60k constituted the training set and the rest the test set. Although the paper did not mention any preprocessing steps or possibly omitted these, we made the choice early on to scale it this way based on some repositories that we had seen [1] [2] [3]. However, later on we realized that the repository linked by the author in their presentation on ICLR14 used binarized MNIST [4], but due to time constraints of having to retrain everything this change was not implemented. Note that the repository is not the author's original code and was created by a third party, so it may not be fully reflective of the paper as it is instead stated as an improvement to the original paper.

Objective function There are two versions of the SGVB estimator proposed in the paper, that estimates the variational lower bound. The second verison of SGVB estimator has lower variance compared to the first version according to the authors[5] and is the one used in the paper for their experiments. Thus, the objective function we is given below for each datapoint x^i

$$L(\theta, \phi; x^i) = \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log((\sigma_j^i)^2) - (\mu_j^i)^2 - (\sigma_j^i)^2\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_\theta(x^i|z^{i,l}) \tag{2}$$

where $z^{i,l} = \mu^i + \sigma^i \odot \epsilon^l$ and $\epsilon^l \sim N(0, \mathbf{I})$. Note that in the experiments the authors mention that for a sufficiently large mini-batch size L is set to 1. In the equation the second RHS term corresponds to the reconstruction loss, while the first is the analytical form of $-D_{KL}(q_\phi(z|x)||p_\theta(z))$ that acts as a regularizer that encourages the approximate posterior $q_\phi(z|x)$ to be close to the prior $p_\theta(z)$[5]. For the entire dataset, we calculate the average variational lower bound as

$$L_{average}(\theta, \phi; x) = \frac{1}{N} \sum_{i=1}^{N} L(\theta, \phi; x^i) \tag{3}$$

As for the loss function when comparing the ground truth with the reconstructed data we use a binary cross entropy (BCE) loss function even though the data is continuous. There are some speculations as to why this would still work, although possibly at the expense of yielding different losses. An intuitive explanation is given in [1], stating that pixel intensities in the range $[0, 1]$ can be seen as the probability of the pixel being on or off. Thus, in this sense BCE is a natural choice as it can be applied to random variables that work in the same way. For completeness, we included small experiments with MSE loss to see if the loss would be comparable to the results of the paper. More details are in the results section.

[1] `https://github.com/pytorch/examples/blob/main/vae/main.py`

[2] `https://github.com/NoviceStone/VAE/blob/master/train_mnist.py`

[3] `https://keras.io/examples/generative/vae/`

[4] `https://github.com/y0ast/Variational-Autoencoder/blob/master/run.py`

Parameters In regards to step sizes for the optimizer, specific ones for the experiments with different latent spaces were not provided, but the authors hinted at using a hyperparameter search chosen from a set of values $\{0.1, 0.01, 0.02\}$ based on the first few iterations. We performed our own hyperparameter search with 2 epochs for each combination of latent space and learning rate from the given set and used the training loss as a metric to pick the best learning rates. We provide the step sizes associated with each experiment in the result section. Weight decay was sampled from $\mathcal{N}(0, 0.001)$ instead of $\mathcal{N}(0, 1)$ that was the papers and this was an oversight on our part.

The optimizer used was SGD with adagrad according to the paper, and training was done with 1670 epochs rounded up from 1667 as the train and test loader were done in iterations of 600 with a mini-batch size of 100 due to the default settings provided by Pytorch dataloaders. Even though the paper only provides the number of training samples as their training iterations we could calculate the number of epochs as $1667 \approx 10^8/(6 * 10^4)$, where each epoch works with 60k datapoints.

Initialization of model weights and biases were sampled from $\mathcal{N}(0, 0.1)$, which was also an oversight. The paper uses $\mathcal{N}(0, 0.01)$.

Autoencoder Regarding the vanilla autoencoder, the architecture can be found in appendix A.2. The model has been implemented to match VAE as closely as possible while still making it an AE. The training settings are the exact same for the AE as VAE.

Sampling Sampling images from the trained model is done by drawing a sample from a standard Gaussian distribution and using it as a latent variable as input to the decoder to reconstruct as a data sample.

3 Results

Let's present the results of our implementation and some first-hand observations.

Experiment settings The main experiments consisted of runs over 5 different latent spaces, for both VAE and AE, with hyperparameters set to follow the paper as closely as possible. These were constant batch size 100, hidden dimension 500, weight decay drawn from a gaussian $\mathcal{N}(0, 0.001)$. For latent spaces 3,10 and 20, the learning rate was set to 0.02 while for latent spaces 5 and 200, the learning rate was 0.01, which was decided by our hyperparameter search.

Plotted figures The main results are the negative variational lower bound (negative ELBO) of VAE with MNIST and BCE losses of AE and are displayed below. For enlared plots we refer to appendix B. The generated samples from both models is below but for enlarged versions we refer to appendix C and the learned 2D manifold for the VAE is in appendix D. The learned manifolds were included to show if VAE learns sensible representations in the latent space. The manifold is a 2D projection of the latent variables produced by the model in the latent space.

Negative ELBO Note that the negative ELBO is presented in the VAE losses, and since we are maximizing the negative ELBO, higher loss is desirable moving towards zero. However, when referring to the loss in the following sections we will treat it as the absolute loss, and thus lower loss indicates better loss.

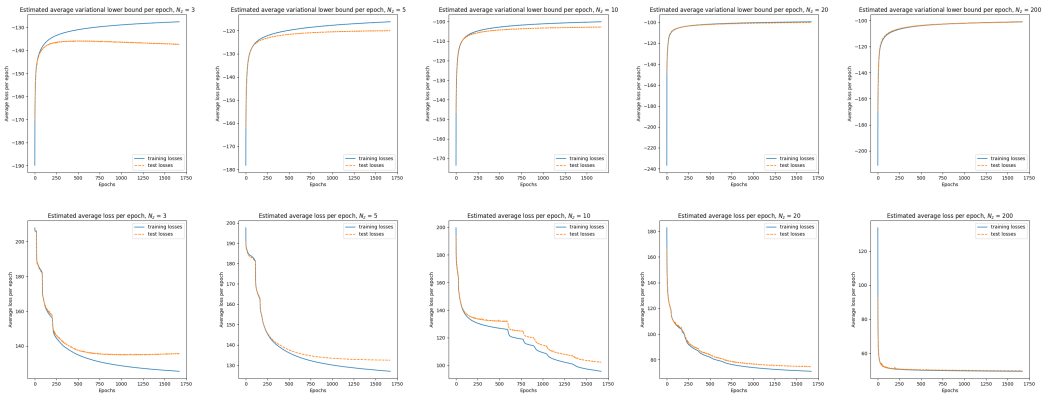


Figure 1: VAE and AE epoch loss for training (blue) and test (orange) data with latent dimensions: 3, 5, 10, 20 and 200

The negative ELBO for the VAE losses in appendix has comparable values as the paper result. Although the appearance seems a bit different due to not using the same metric on the x-axis as the paper, we can observe a similar trend; test and train loss are comparable at the early iterations, but sets apart usually around 100-250 epochs in our case, which perhaps is similar in the paper as well, although we cannot say with full confidence as the figures are smaller and harder to read exactly so we have to eyeball it. It looks like in general the test losses in the paper starts converging around 10^7, which is approximately 1670 epochs, so it is in our range, but it does not fully converge to the same extent as our experiments. Note that we use epochs, where each epoch constitutes 60k training samples in the paper. Thus for the latter epochs, between 200 to 1670, these steps constitutes approximately steps between 10^7 and 10^8 in the paper, so the scaling of the x-axis is not the same. As a consequence the convergence and stagnation of the loss plots looks more amplified than they are compared to the paper. As for the AE loss the test loss does not stagnate or improve/decrease slightly in the same way as the VAE, except at latent space 200 which follows the same trend as VAE losses. AE test losses do decrease during the iterations showing that the model is learning. However, because AE does not have the same notion of a negative ELBO as VAE, the losses are possibly not comparable to each other.

Latent space Larger latent spaces results in better losses in accordance with the paper. We can see that for each increasing latent space dimension, the test loss converges to a lower loss value compared to its predecessor. The largest latent space also performs well in terms of loss, where it does not overfit more because of more latent variables, comparable with the paper, showing that the regularization in the ELBO works for our model.

Generated samples The generated samples show that lower latent space seems to result in higher quality. Additionally, it shows that smaller losses, as seen in experiments with higher latent space, does not result in better samples. For instance, experiments with latent space 200 which has the lowest loss for both models results in worse generated samples than latent space 3, which has considerably higher loss. This is in agreement with the results of the paper. Regarding how the generated samples compare to the paper is hard to say, since they look similar to the extent that it is hard to tell which one is better without picking apart the finer details of the images. Because it is hard to discern the differences in quality, this to us indicates that the quality is comparable to the paper. The manifold plotted with the decoder of our model is comparable in quality to the manifolds in the paper, suggesting that the model representation in the latent space is as good as the paper's.

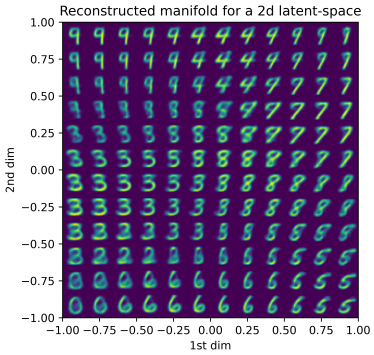


Figure 2: VAE manifolds, with a 2D model

Figure 3: Randomly sampled numbers from the latent space of the VAE (above) and the AE (below) from latent spaces 3, 10, and 200 (from left to right). The omitted models can be found in section C in the appendix.

4 Discussion

Generated images On the topic of generated images, we can see how the quality of the randomly generated numbers deteriorates as the latent space grows for the VAE. We can imagine that a lower latent dimension enforces a stronger representation. Another reason for this is could be due to the fact that the representations that are generated by the encoder become more sparse as the size of the latent space grows. In a sufficiently small space, all or most points will be close to an actual encoding of an image. Concerning the AE, because the assumption of a Gaussian latent space is not applied, i.e we did not enforce a Gaussian encoder, we did not expect to get sensible results, but as clearly shown for instance for latent space 3 the autoencoder could produce sensible images ressembling numbers assuming a gaussian distribution for the latent space. One possible reason as earlier stated could be that the AE samples can be close to an encoding of a specific image in the latent space when the dimensions are very few. It seems that it is however clear that in higher dimensions the decoder just produces noise, not resembling numbers at all, possibly demonstrating that the latent space of the AE is not following a gaussian distribution.

Latent space In the results we made the observation that larger latent spaces lead to smaller losses. This could be explained by the fact that the model has a larger latent space which makes more complex representations of the images possible. The size of the latent space in a model, in a real

setting, depends on what the user wants to achieve. If the goal is to de-noise data the reduction of loss should be weighted to the complexity of the representation. A simpler representation might simply capture the structure of the data and ignore the noise.

We can see the same patterns in the loss for the AE as in the VAE. That is, as the latent space grows, the loss decreases. The reason for this is likely due to the same reason as stated for the VAE. However, convergence for the AE seems to be slower. Why the convergence is slower for the AE is not clear to us, but it is not a surprise that it behaves differently as the models are not the same.

Convergence Another observation we made is that our VAE model almost fully converges after 100-250 epochs, whereas in the paper where it does not happen until very late or at the end of the training. Why this is the case is hard to say, but possible reasons could be pertaining to the choice of decoder, the use of continuous MNIST and oversights on the hyperparameters such as weight decay and model initialization. However, a more thorough set of experiments would need to carried out in order to verify these suspicions.

All in all, despite the differences we observed, it seems like the loss values are at comparable levels and the generated samples are comparable to the paper, although not entirely the same as the paper, meaning that we find the implementation of VAE and AE successful.

Improvements The first improvement for the project merely for improving reproduction would have been to adhere to the model architecture fully and parameters even more to the paper. These are

- Switch from weight decay $\mathcal{N}(0, 0.001)$ to $\mathcal{N}(0, 1)$ according to the paper
- Set model initilization to $\mathcal{N}(0, 0.01)$ according to the paper instead of $\mathcal{N}(0, 0.1)$
- Switch Gaussian decoder to Bernoulli decoder
- Use binarized MNIST data instead of 8-bit greyscale MNIST

In addition to these, if improving the paper results are desired then the repository linked by the author in their presentation at ICLR14, suggests that using an Adam optimizer and Relu activation results in faster convergence[5]. Additionally, it would be interesting to use a Gaussian decoder together with MSE as loss function to see whether it would give better or comparable samples as the paper. There is a derivation that is easy to make, that shows for a Gaussian decoder the reconstruction loss would be proportional to MSE[6].

Critique A possible critique to the author is the lack of details in terms of dataset or parameters. They did not mention preprocessing steps of the data and if they treated the MNIST as binary or continuous. What was more confusing was that the authors linked to a continuous 8-bit greyscale MNIST. Given that no preprocessing steps were shared it made sense to treat it as continuous data. Looking at other repositories of third parties reimplementing the VAE, we found that they used BCE with continuous MNIST yielding results close to the paper, which is why we opted for it instead of using MSE. However, the repository linked in the authors presentation treats the data as binary and also uses BCE together with a Bernoulli decoder. Unfortunately, as this was not linked or mentioned at all in the paper, and the fact that we discovered this too late we could not implement these changes.

Seeing as this was a reproduction project and that our main focus was on reproducing a VAE model that would produce comparable or same negative ELBO results and samples, it is not really relevant for this project to critique the method in comparison to other methods or the benchmarks in the paper, as the experiments done for this very project are not conducive for this (we did not implement any baselines), as such we will leave this out in the discussion.

Future work For future work there are multiple models that in recent time has been developed as improvement of VAE. Some of the popular ones are VQ-VAE and Beta VAE. VQ-VAEs are VAEs with discrete latent spaces. They are useful in speech recognition due to the fact that phonemes benefit from discrete representations, as well as with images with discrete descriptions of their contents.[8] Beta VAEs are VAEs with a hyper-parameter, β. This modification forces the latent variables to be

[5] `https://github.com/y0ast/VAE-Torch`

[6] `https://stats.stackexchange.com/questions/347378/variational-autoencoder-why-reconstruction-term-is-same-to-square-loss`

independent and gives us an untangled representation in the latent space. This is useful as it makes the latent space explainable and gives us the possibility to change individual characteristics of the reconstruction separately.[4]

References

[1] Anil A. Bharath Antonia Creswell, Kai Arulkumaran. On denoising autoencoders trained to minimise binary cross-entropy. 2017. arXiv:1708.08487v2.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. doi:10.1109/TPAMI.2013.50.

[3] Gundersen. The reparameterization trick. http://gregorygundersen.com/blog/2018/04/29/reparameterization/, 2018.

[4] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017. URL: https://openreview.net/forum?id=Sy2fzU9gl.

[5] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. 2013. arXiv:1312.6114v11.

[6] John Paisley, David Blei, and Michael Jordan. Variational bayesian inference with stochastic search, 2012. URL: https://arxiv.org/abs/1206.6430, doi:10.48550/ARXIV.1206.6430.

[7] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models, 2014. URL: https://arxiv.org/abs/1401.4082, doi:10.48550/ARXIV.1401.4082.

[8] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *CoRR*, abs/1711.00937, 2017. URL: http://arxiv.org/abs/1711.00937, arXiv:1711.00937.

[9] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010. URL: https://dl.acm.org/doi/10.5555/1756006.1953039, doi:10.5555/1756006.1953039.

A Model architecture

Note that for both models hd denotes hidden dimension and ls latent space. Both models have the same hidden dimension, which is 500.

A.1 VAE

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
          Flatten-1                  [-1, 784]               0
           Linear-2                   [-1, hd]         392,500
             Tanh-3                   [-1, hd]               0
           Linear-4                   [-1, ls]           1,002
           Linear-5                   [-1, ls]           1,002
           Linear-6                   [-1, hd]           1,500
             Tanh-7                   [-1, hd]               0
           Linear-8                   [-1, ls]           1,002
           Linear-9                   [-1, ls]           1,002
          Linear-10                  [-1, 784]         392,784
         Sigmoid-11                  [-1, 784]               0
================================================================
Total params: 790,792
Trainable params: 790,792
Non-trainable params: 0
```

For the encoder we have the variational parameters $\phi = \{w_1, w_2, w_3, b_1, b_2, b_3\}$, where the w's and b's are the weights and biases of the MLP for the encoder, in the approximated distribution $q_\phi(z|x)$. Flatten-1, Linear-2 and Tanh-3 together corresponds to the hidden layer in the encoder and the output $h_{encoder}$ of this layer is used to obtain μ and $log(\sigma)$ by linear layers Linear-4 and Linear-5 respectively. These are then used in the reparametrization trick to produce the encoded data z. The generative parameters of the decoder corresponds to $\theta = \{w_4, w_5, w_6, b_4, b_5, b_6\}$, where the w's and b's are the weights and biases of the MLP for the decoder. Linear-6 and Tanh-7 corresponds to the hidden layer in the decoder in which its output $h_{decoder}$ is used by Linear-8 and Linear-9 to obtain μ_θ and $log(\sigma_\theta)$ for $p_\theta(x|z)$. Finally, Linear-10 and sigmoid-11 uses $h_{decoder}$ to output the reconstructed data $\hat{x}$.

A.2 AE

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
          Flatten-1                  [-1, 784]               0
           Linear-2                   [-1, hd]         392,500
             Tanh-3                   [-1, hd]               0
           Linear-4                   [-1, ls]           1,002
             Tanh-5                   [-1, ls]               0
           Linear-6                   [-1, hd]           1,500
             Tanh-7                   [-1, hd]               0
           Linear-8                  [-1, 784]         392,784
          Sigmoid-9                  [-1, 784]               0
================================================================
Total params: 787,786
Trainable params: 787,786
Non-trainable params: 0
----------------------------------------------------------------
```

Flatten-1, Linear-2, Tanh-3 corresponds to the hidden layer in the encoder. Linear-4 and Tanh-5 is the output layer for the latent space producing latent variable z. Linear-6 and Tanh-7 is the hidden layer

in the decoder. Linear-8 and Sigmoid-9 is the output layer in the decoder producing the reconstructed data $\hat{x}$. Weights and biases $\{v1, v2, c1, c2\}$ for the encoder and $\{v3, v4, c3, c4\}$ for decoder.

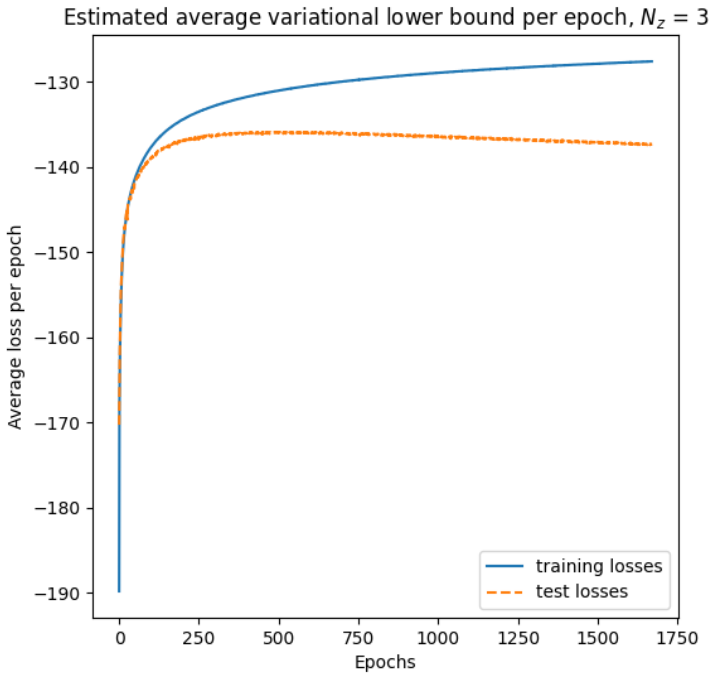
A.3 Workflow of VAE

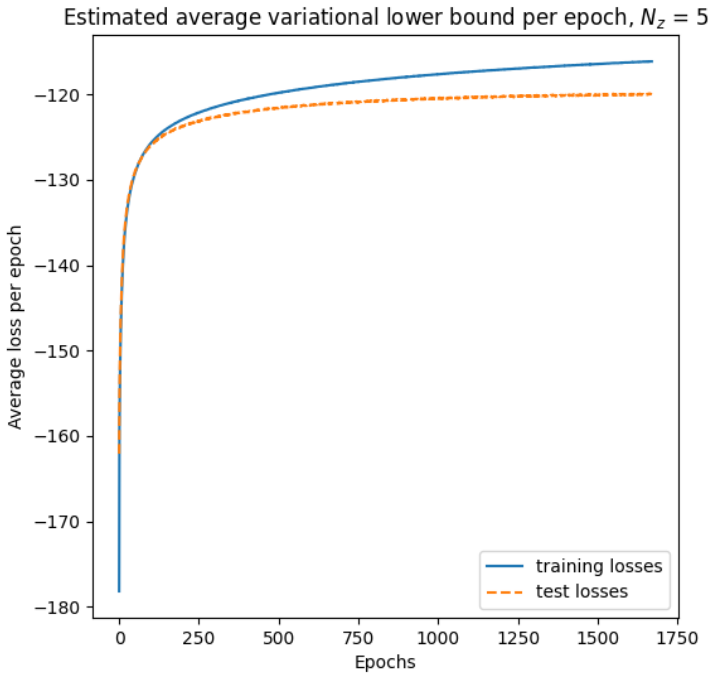
The workflow of the implemented VAE model can be summarized by the operations below taken from an article by Gundersen[3]. This is for a single datapoint x, where x is continuous. MSE can be swapped with BCE (binary cross entropy) if x is binary data.

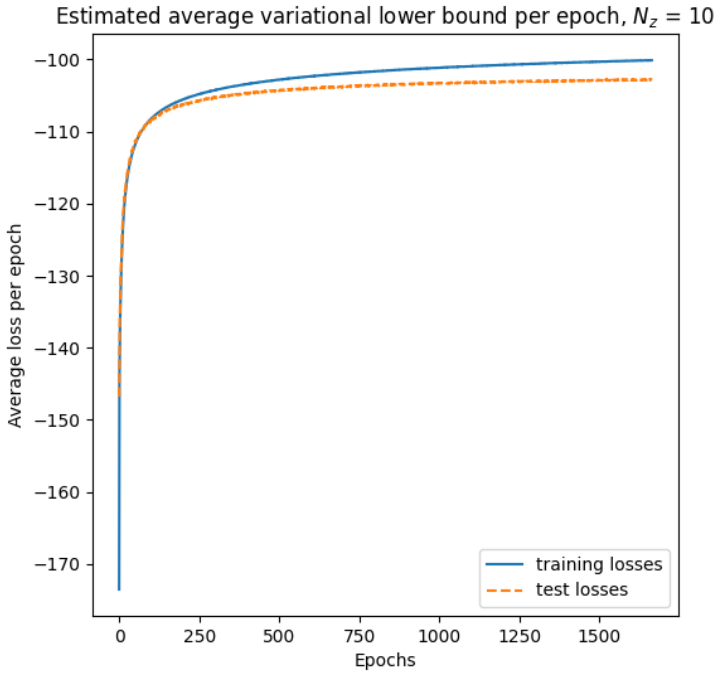
$$\mu_\phi, \sigma_\phi = M(x), \sum(x) \quad \text{Push x through encoder}$$
$$\epsilon \sim N(0, I) \quad \text{Sample noise}$$
$$z = \epsilon \odot \sigma_\phi + \mu_\phi \quad \text{Reparameterize}$$
$$\hat{x} = p_\theta(x|z) \quad \text{Push z through the decoder}$$
$$reconstruced\_loss = MSE(x, \hat{x}) \quad \text{Compute reconstruction loss}$$
$$var\_loss = -KL(q_\phi(z|\hat{x})||p_\theta(z)) \quad \text{Compute variational loss}$$
$$L = reconstruced\_loss + var\_loss \quad \text{Combine losses}$$

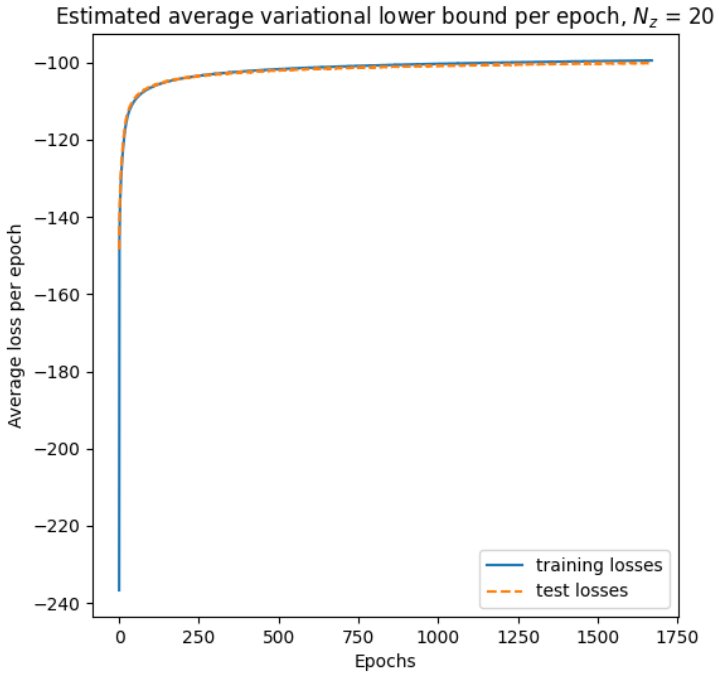
B Losses

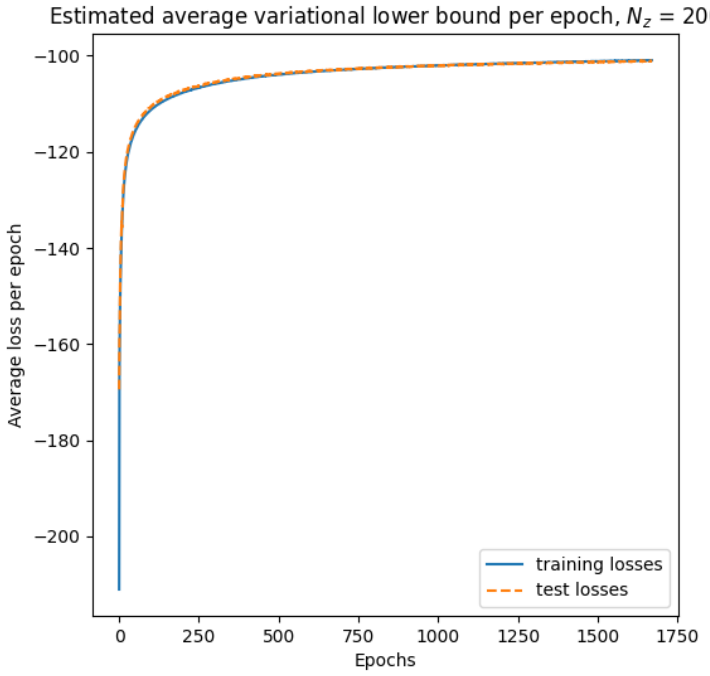
B.1 VAE



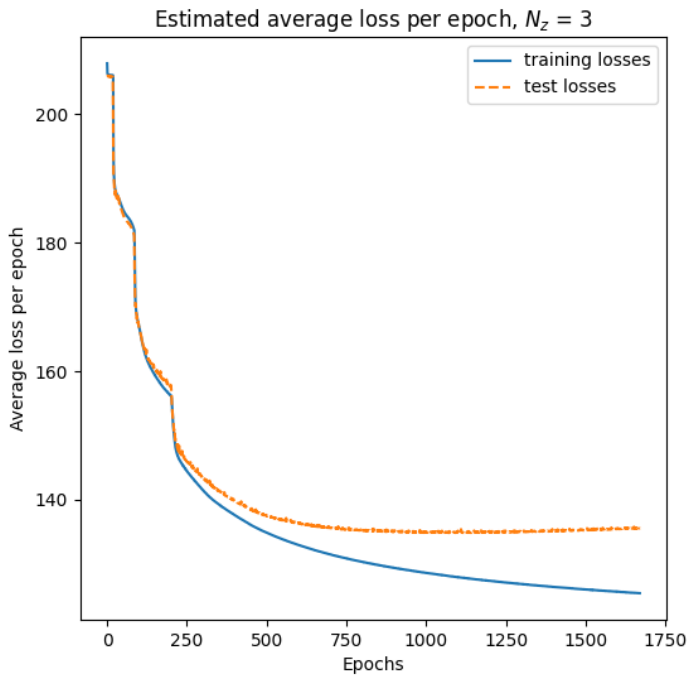


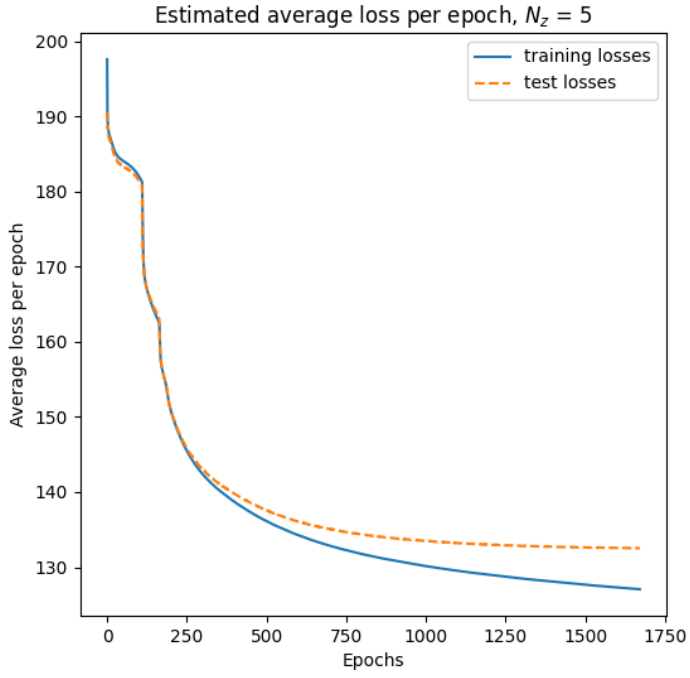

Estimated average variational lower bound per epoch, $N_z = 10$
Average loss per epoch
−100
−110
−120
−130
−140
−150
−160
−170
0
250
500
750
1000
1250
1500
Epochs
training losses
test losses

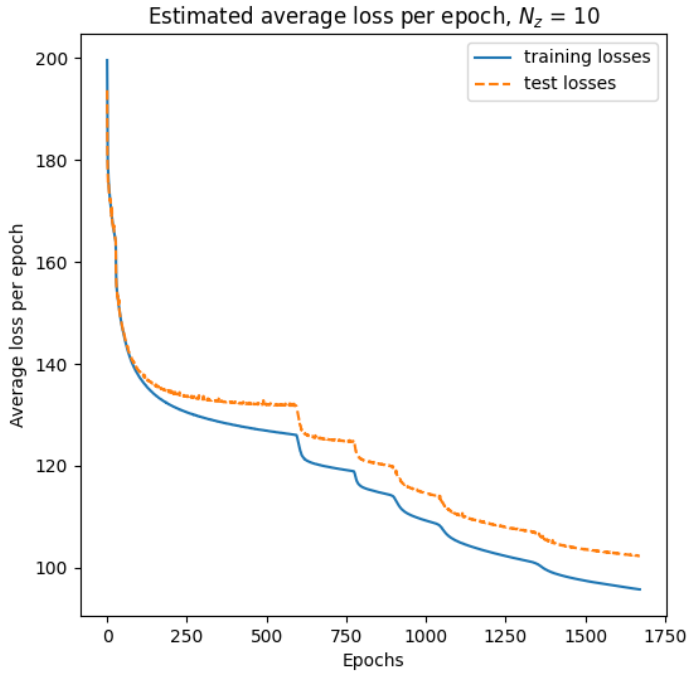

Estimated average variational lower bound per epoch, $N_z = 20$
Average loss per epoch
−100
−120
−140
−160
−180
−200
−220
−240
0
250
500
750
1000
1250
1500
1750
Epochs
training losses
test losses

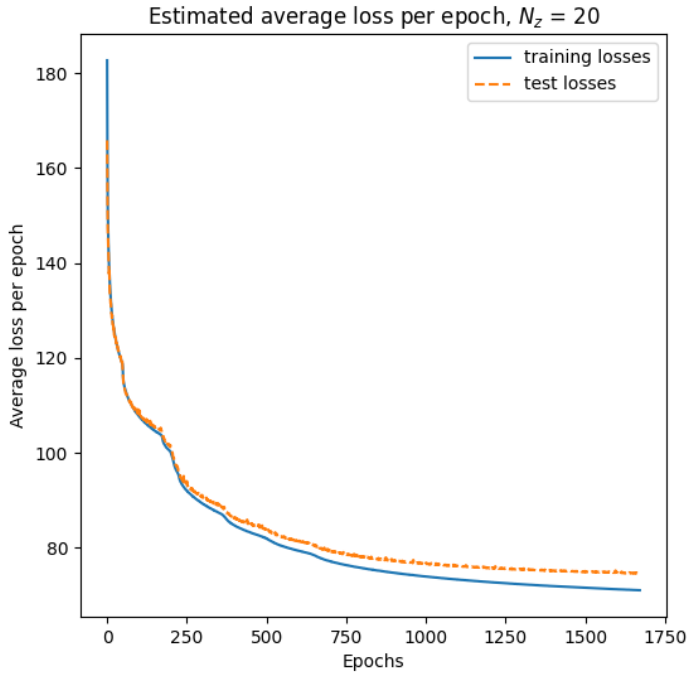

Estimated average variational lower bound per epoch, $N_z = 200$
Average loss per epoch
−100
−120
−140
−160
−180
−200
0
250
500
750
1000
1250
1500
1750
Epochs
training losses
test losses

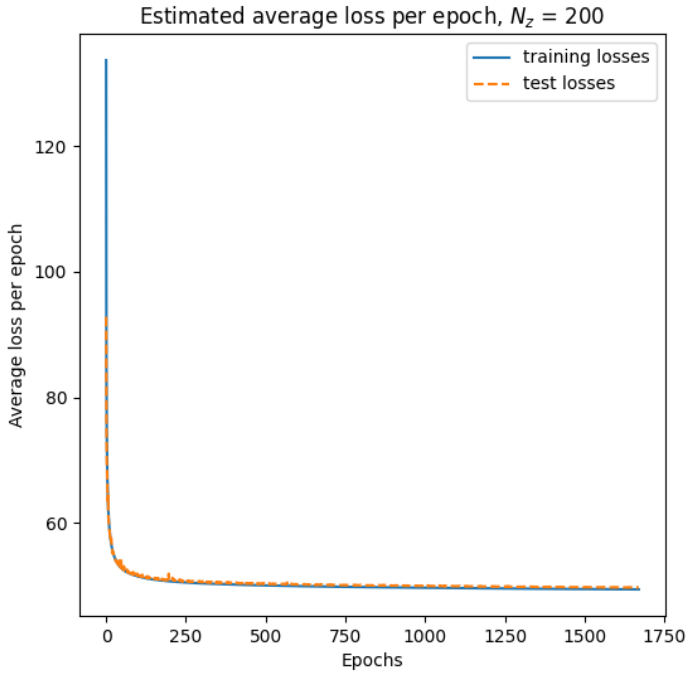
B.2 AE


Estimated average loss per epoch, $N_z = 3$
training losses
test losses
Average loss per epoch
200
180
160
140
0
250
500
750
1000
1250
1500
1750
Epochs


Estimated average loss per epoch, $N_z = 5$
training losses
test losses
Average loss per epoch
200
190
180
170
160
150
140
130
0
250
500
750
1000
1250
1500
1750
Epochs


Estimated average loss per epoch, $N_z = 10$
training losses
test losses
Average loss per epoch
Epochs
200
180
160
140
120
100
0
250
500
750
1000
1250
1500
1750

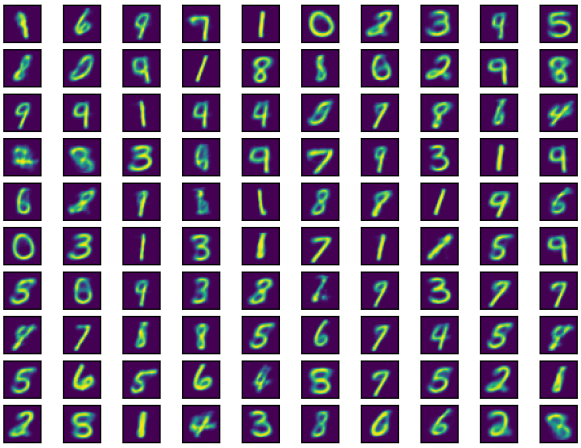

Estimated average loss per epoch, $N_z = 20$
training losses
test losses
Average loss per epoch
Epochs
180
160
140
120
100
80
0
250
500
750
1000
1250
1500
1750


Estimated average loss per epoch, N_z = 200
training losses
test losses
Average loss per epoch
120
100
80
60
0
250
500
750
1000
1250
1500
1750
Epochs

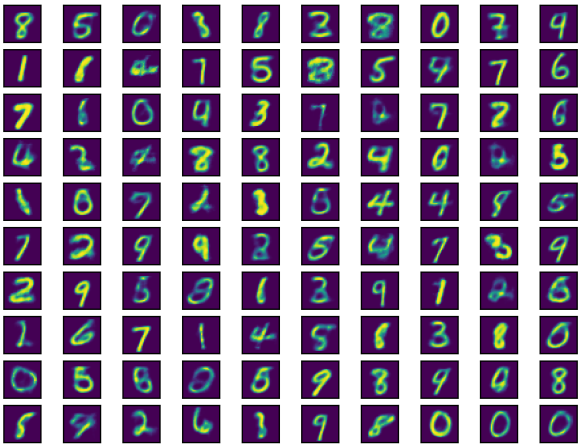
C Generated samples using VAE and AE decoder

C.1 VAE

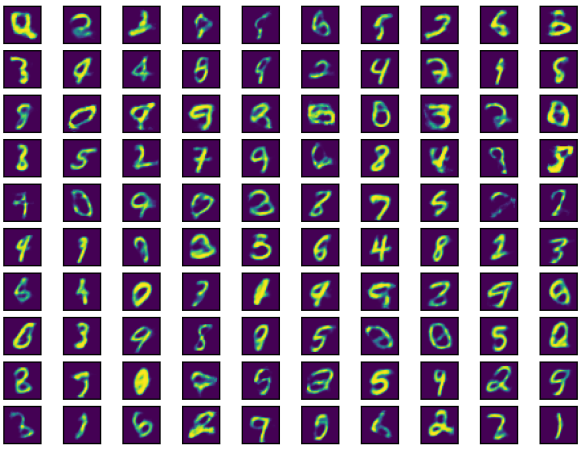
Images generated with the trained decoder
1670 epochs, 3d latent space



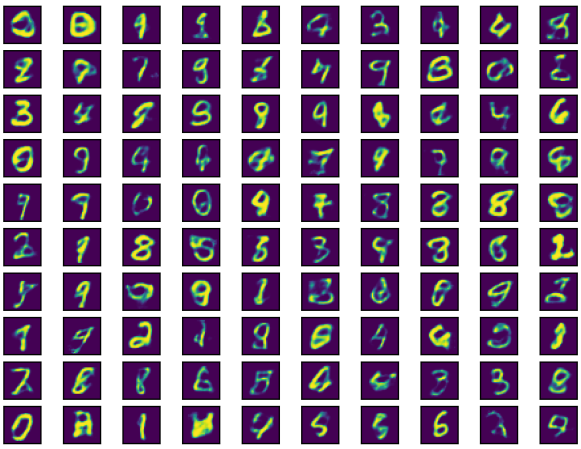
Images generated with the trained decoder
1670 epochs, 5d latent space



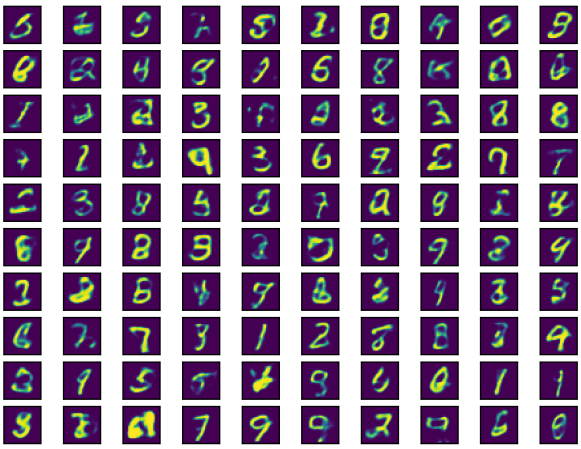
Images generated with the trained decoder
1670 epochs, 10d latent space



Images generated with the trained decoder
1670 epochs, 20d latent space

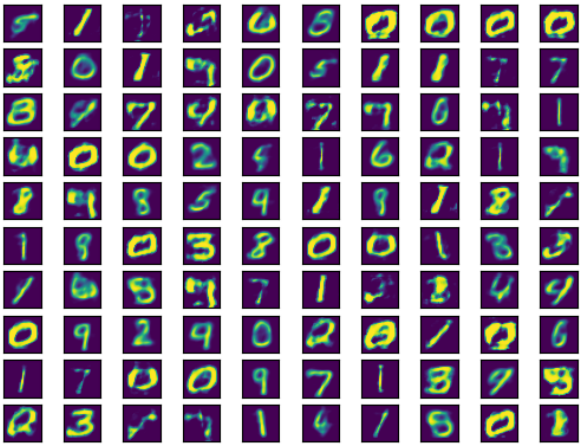


Images generated with the trained decoder
1670 epochs, 200d latent space

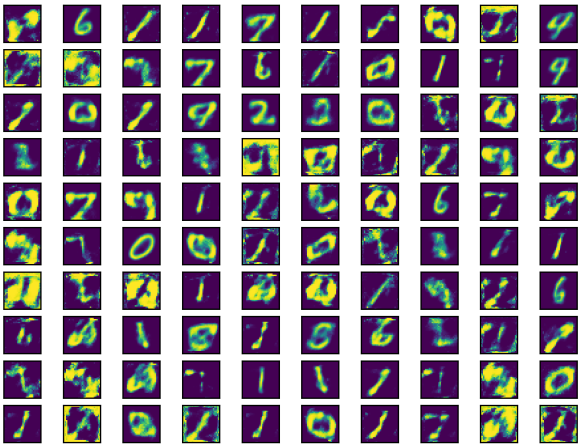


C.2 AE

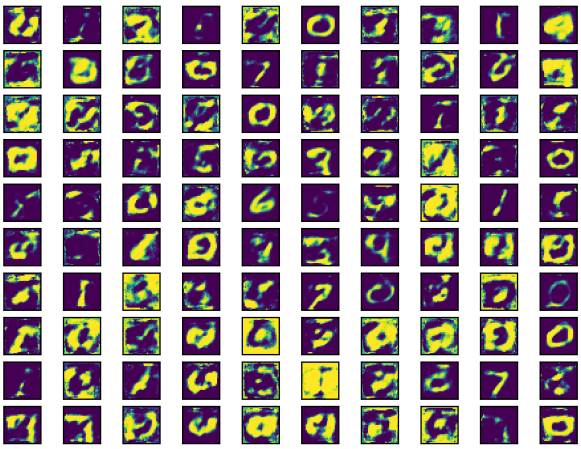
Images generated with the trained decoder
1670 epochs, 3d latent space



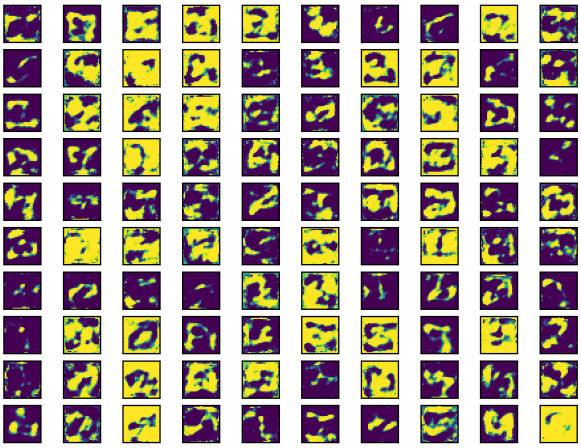
Images generated with the trained decoder
1670 epochs, 5d latent space



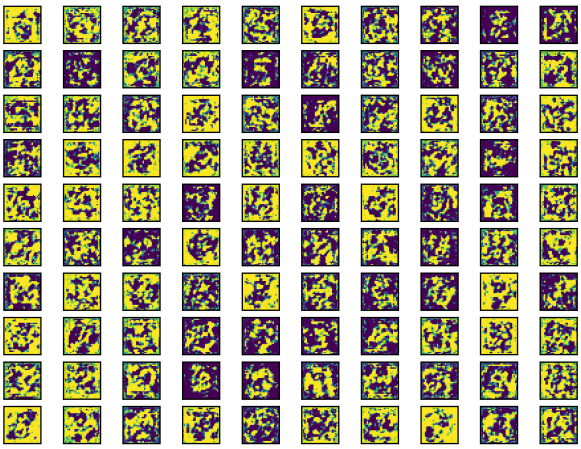
Images generated with the trained decoder
1670 epochs, 10d latent space



Images generated with the trained decoder
1670 epochs, 20d latent space



Images generated with the trained decoder
1670 epochs, 200d latent space



D Manifold

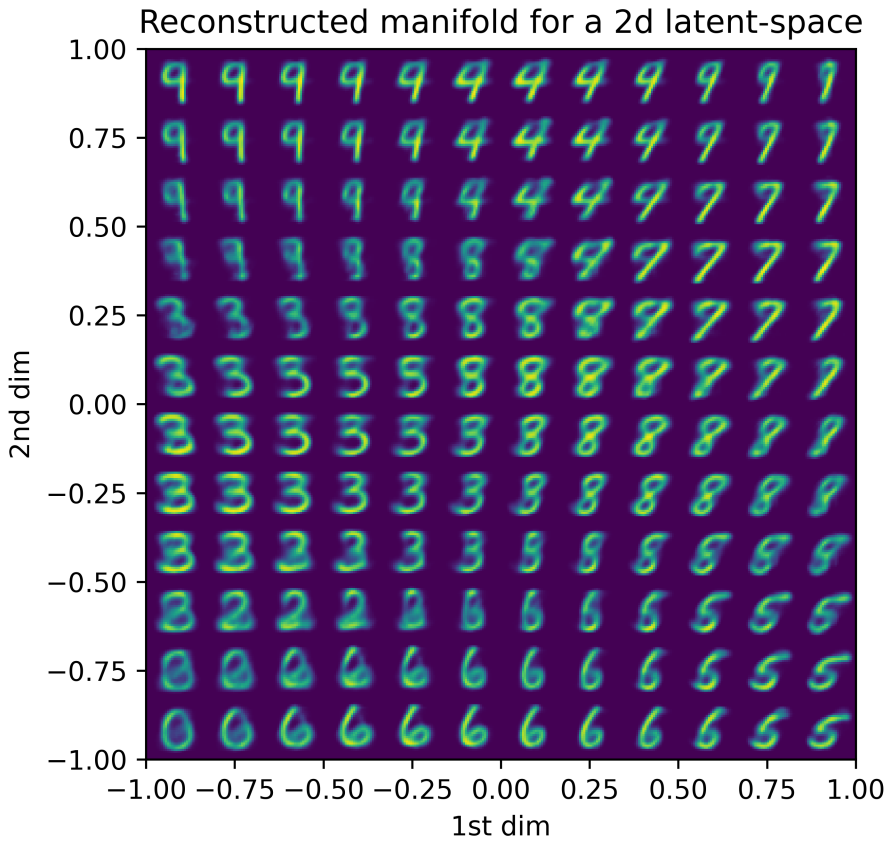


Figure 4: VAE manifold ls2