INSTITUT
POLYTECHNIQUE
DE PARIS

Deep Learning II Project
RBM, DBN and DNN on Binary AlphaDigits and MNIST

Axel Dumont

Mars 2024

Contents

INSTITUT POLYTECHNIQUE DE PARIS

1 Introduction

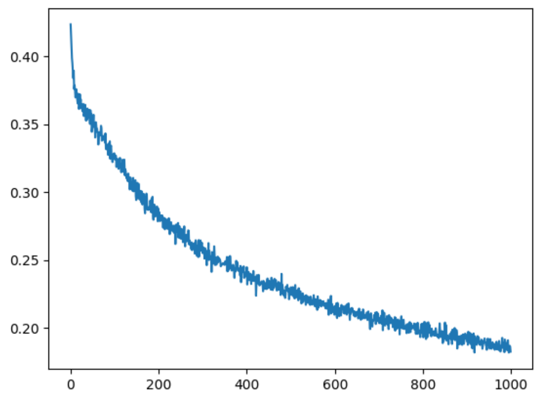
We aim in this project to test the generative power of differents models based on RBM or a pile of RBM (for Deep Belief Networks and Deep Neural Networks with a classification layer) on both datasets Binary AlphaDigits and MNIST. We will then compare the results obtained with a VAE.

The main study aims at comparing two DNNs, one pretrained and one untrained while fine tuning hyperparameters to observe their performaces

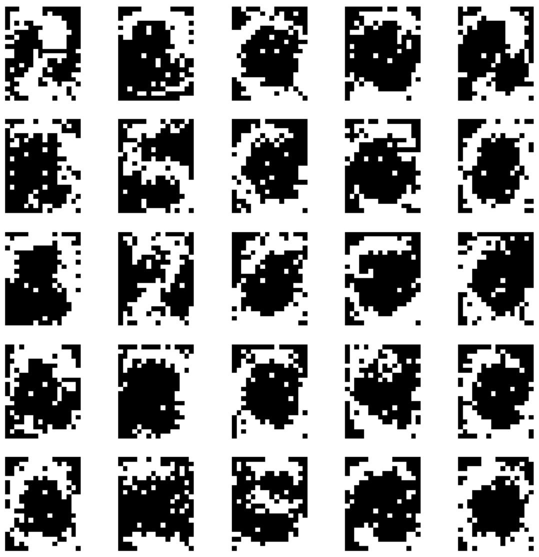
2 RBM

2.1 Simple Training on Binary AlphaDigits

First, after implementing the structure of an RBM, we decided to test it simply on the digits of the database, and we could then plot the evolution of the recreation loss $\frac{1}{n}\sum_i(\hat{y}_i - y_i)^2$ and generate a few sample from this trained RBM.



(a) evolution of recreation loss



(b) Sample generation

Figure 1: RBM on digits

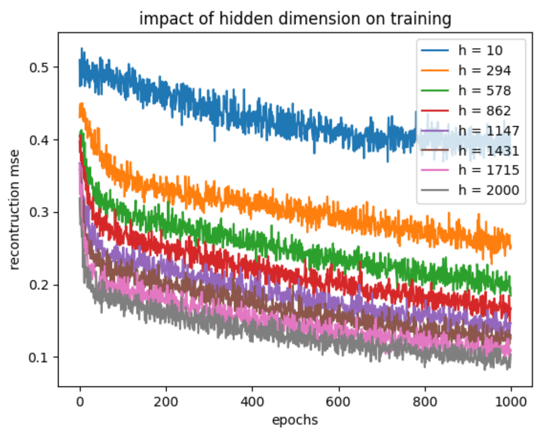
We start to recognize numbers even though it is very noisy for 1000 epochs and 50 iterations of Gibbs sampling.

2.2 Effect of the hidden dimension on RBM

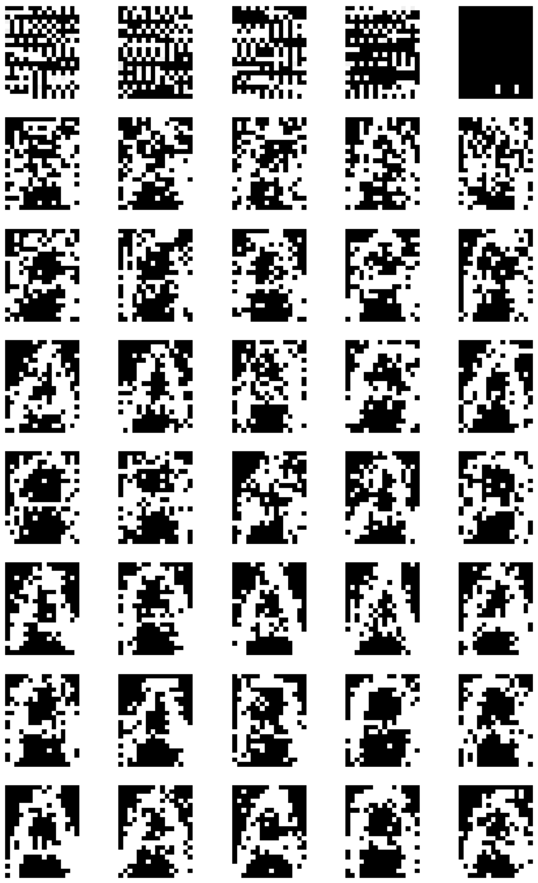
We then wanted to get a better understanding at what some hyperparameters were responsible of, and started by changing the size of the hidden dimension and see how it affected the generation. Each row of the generation is a different hidden dimension from least to highest and each column adds 100 iteration for the Gibbs sampler.

We see a clear improvement the higher the hidden dimension is

INSTITUT POLYTECHNIQUE DE PARIS



(a) Evolution of recreation loss

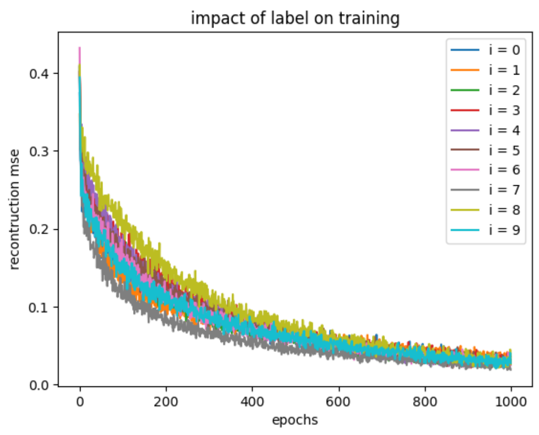


(b) Sample generation

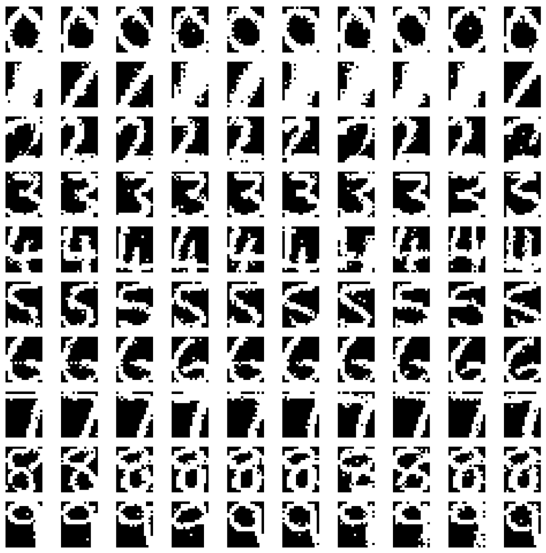
Figure 2: Hidden dimension of RBM

2.3 Training on only one label

We wanted to see here if one number in particular was difficult to learn for the RBM. We get in the end that the 8 decreases slower but all seem to converge at 1000 epochs of training.



(a) Evolution of recreation loss

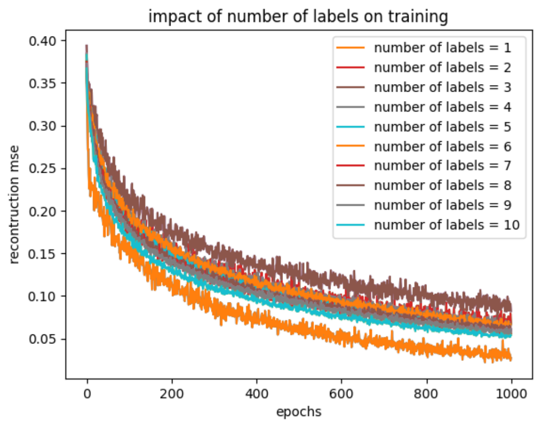


(b) Sample generation

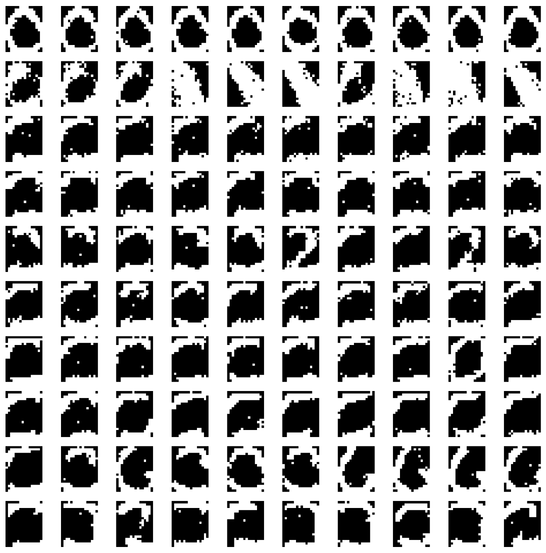
Figure 3: Hidden dimension of RBM

2.4 Number of labels in training

We are interested here in the number of labels the RBM can learn and still produce good results without upscaling its architecture.



(a) Evolution of recreation loss



(b) Sample generation

Figure 4: Number of labels in training

We see a quick collapse in generation after two labels for a fixed 800 hidden units

3 DBN

3.1 Simple training on Binary Alphadigits

Much like the RBM, we first took a network size standard and trained the DBN (a pile of RBM) on the digits of Binary AlphaDigits. This is for 4 RBMs with dimentions: (320,160) (160,100) (100,50) (50,10)

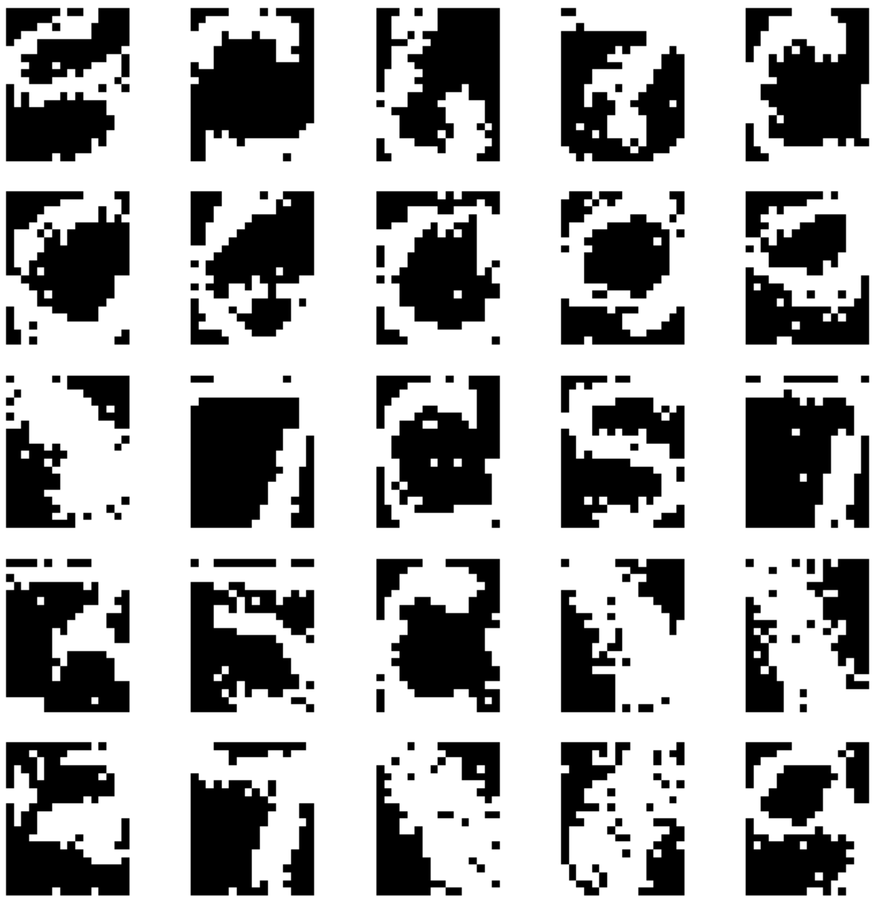


Figure 5: DBN on only digits

3.2 Effect of the number of hidden layers

We wanted to get the effect of the number of hidden layers on the generation of the DBN:

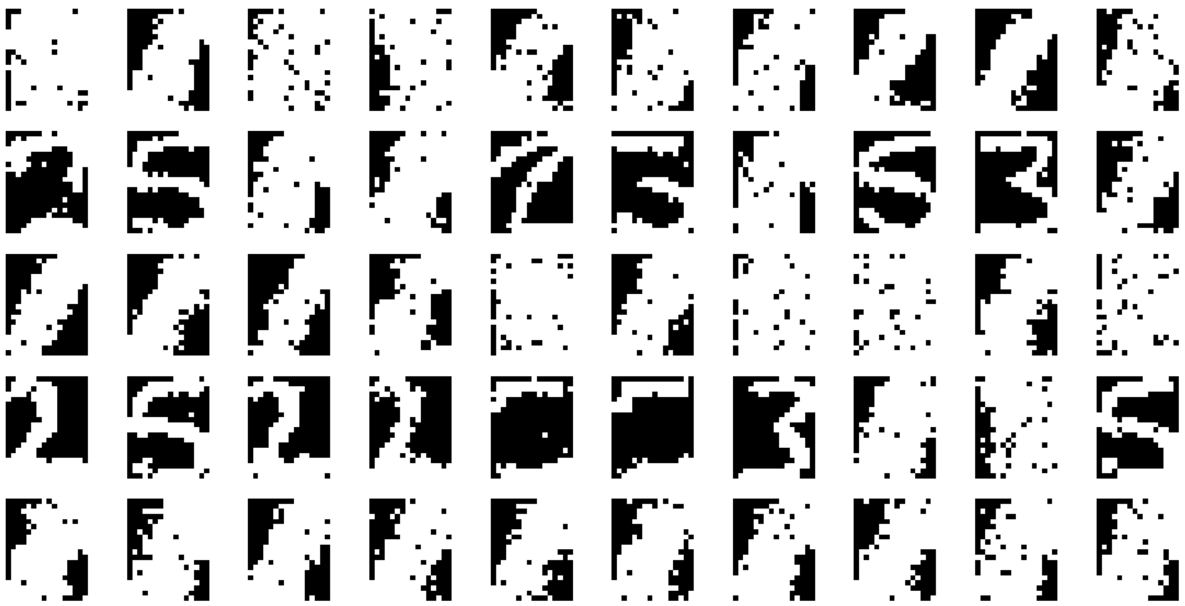


Figure 6: Effect on generation by increase of hidden layers

We see a better generation with the number of hidden layers, especially for the 1 that gives all blank images for fewer hidden layers.

4 DNN

We now observe different tuning of parameters for a DNN, a DBN with a classification layer to classify MNIST data. The idea here is to compare two networks, with one pretrained (RBM that are not the classification layer are trained) and one untrained. Both will go through retropropagation to update their weightsand obtain a better classification score.

4.1 Simple Training on MNIST

As in the two previous parts, we first trained our DNNs with a fixed network size on MNIST and compared the evolution of the loss and accuracy for both networks.

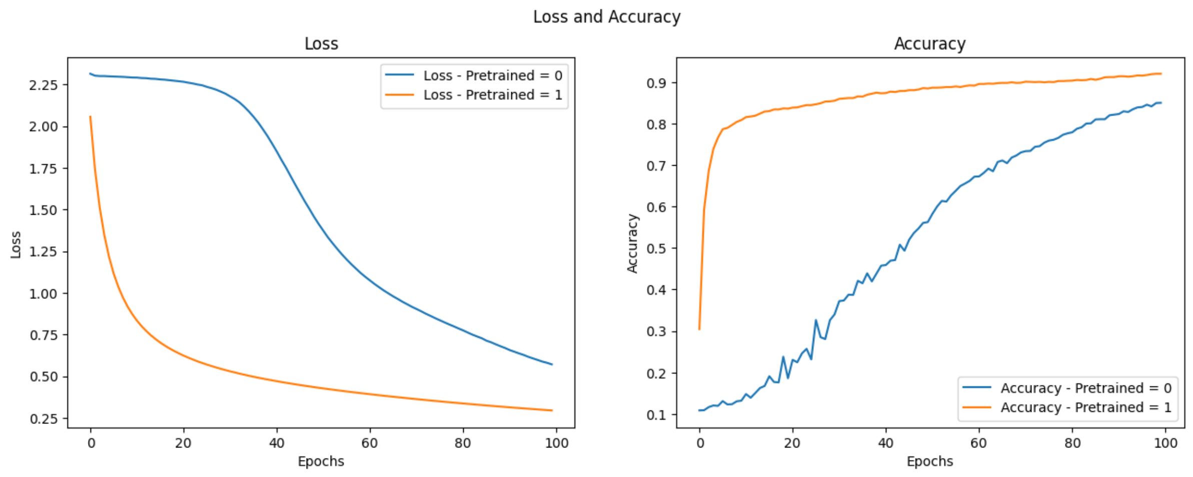


Figure 7: Loss and accuracy for pretrained and untrained DNN

We see a clear difference of evolution while the end result for 100 epochs and a network size of [784, 200, 200, 30, 10]. While the pretrained gains accuracy in an exponential form, the untrained learns more linearly.

4.2 Effect of the number of hidden layer

Like the DBN, we want to check the impart of the number of hidden layers on the classification task. The hidden dimension is fixed at 200 and we add one layer each time to get the figure below.

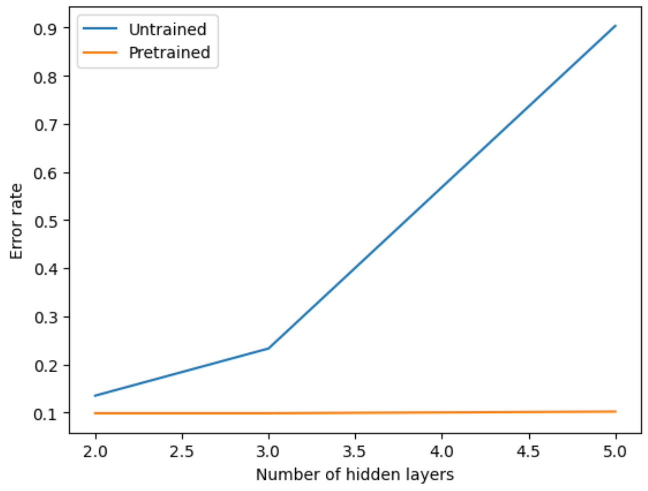


Figure 8: Error rate for diffrent numbers of hidden layers

We see that while the pretrained network is robust to the number of hidden layers, the untrained network falls to having the same performance as a random guess for 5 hidden layers.

4.3 Effect of the number of hidden units

Here we are fixing the number of hidden layers to to and vary the hidden dimension:

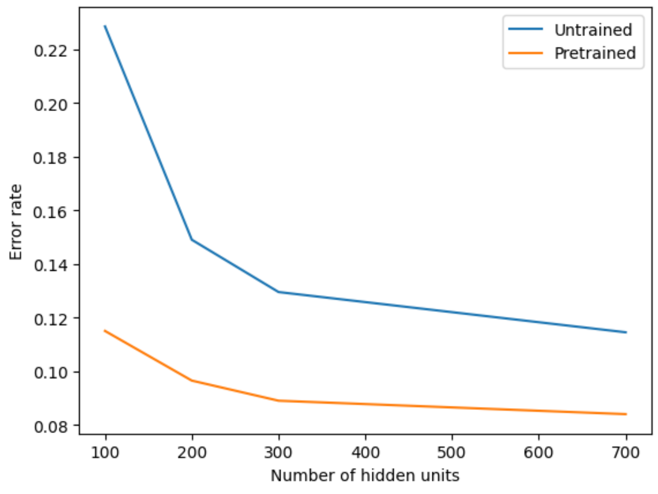


Figure 9: Error rate for different hidden units

Here the untrained network obtains relatively good performance, especially in higher hidden dimension, but the pretrained network is far ahead. We can still note that performances seem to converge to a minimum from 400 units

4.4 Effect of the number of training data

Finally, we vary the number of data used in training to see how it affects the performances of both networks

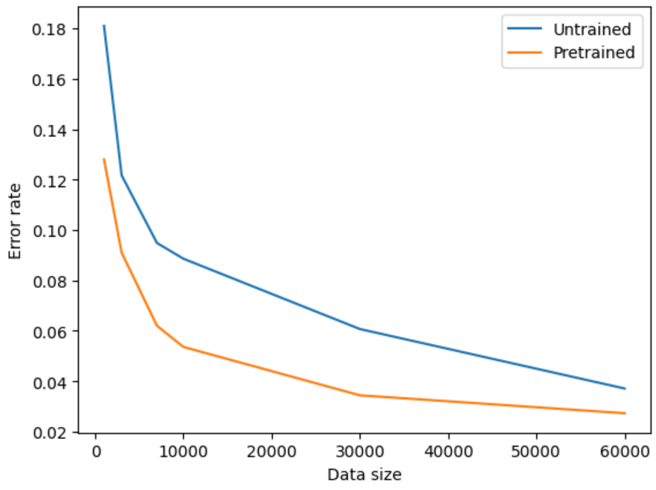


Figure 10: Error rate for different number of training data

While we see great performances on display, and that the untrained network is as close as possible to the pretrained one for all dataset in training, we must observe that the time taken by the training using this much data is very high, especially compared to pretraining the network and using less data.

In addition, we see like in the previous subsection that the improvement in performances are decreasing for the pretrained network from 30,000 data samples.

5 Conclusion

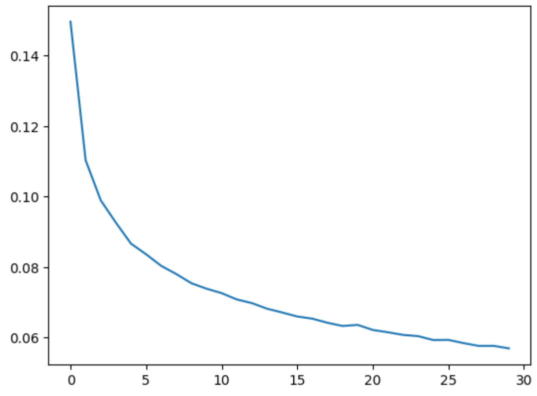
In the end, we see that everywhere, the pretrained network has better performances than the untrained one. However, these results are to put in perspective because pretraining, especially with high dimension hidden layers is costly in time.

By the curves we obtained, we can deduce an architecture that might be the best compromise: a pretrained DNN with 4 hidden layers of 400 neurons trained on 30,000 data samples.

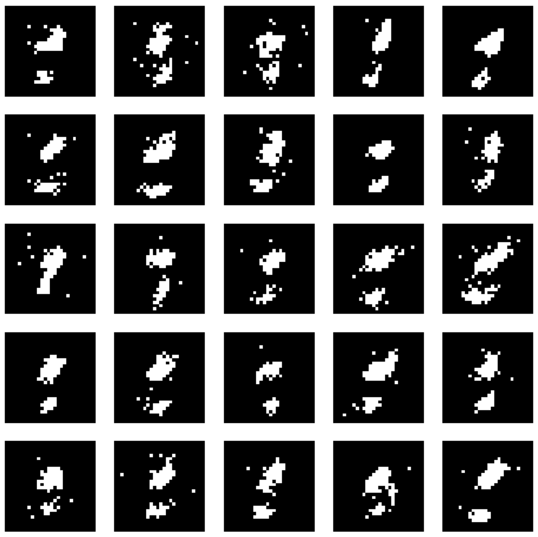
6 VAE comparison

6.1 RBM on MNIST

First we trained our RBM with our MNIST data:



(a) Evolution of recreation loss

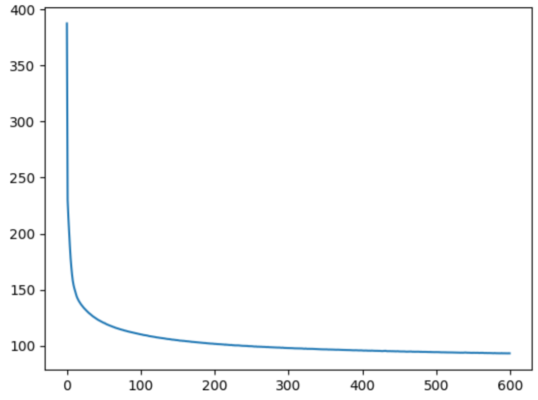


(b) Sample generation

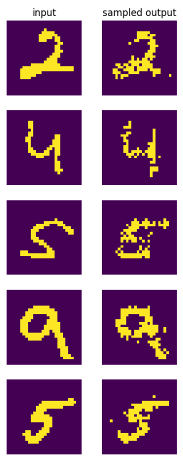
Figure 11: Training RBM on MNIST

6.2 VAE training and sampling

After defining our VAE architecture, the dimension of our decoder, etc..., we can train it and compare the samples to input data



(a) Evolution of the loss



(b) Sample generation from input

Figure 12: Training of the VAE

6.3 Sampling from random

We can then sample from random to se the generative capability of the VAE and compare it to our RBM:

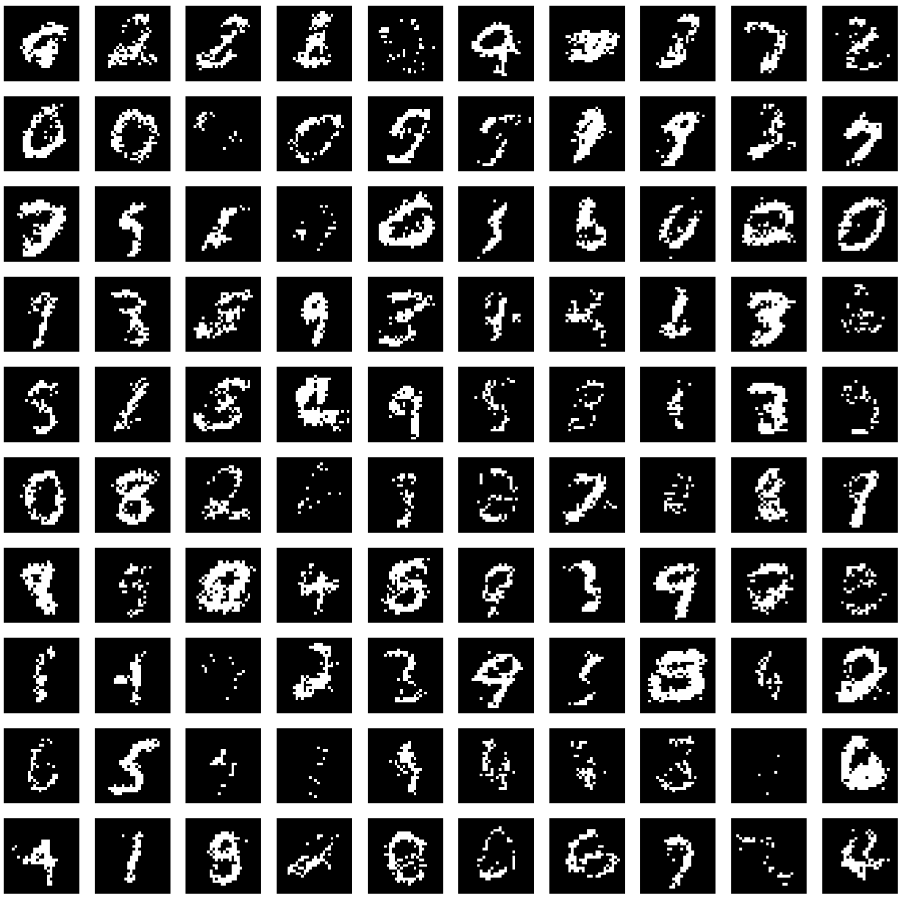


Figure 13: Sampling from random with VAE

We get really good results for 200,000 parameters in the VAE while our RBM (784,1000) with hence more than 785,000 produces less quality reconstruction