Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 4

Restricted Boltzmann Machines and Deep Belief nets

Côme Lassarat, Victor Sanchez, Marie-Ange Stefanos

March 4, 2022

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- to understand the key concepts of Restricted Boltzmann Machines (RBMs) and implement a RBM using a contrastive divergence algorithm to learn data representations
- given the done work on RBMs, to implement a deeper architecture called Deep Beliefs Network (DBN) and its greedy layer-wise pretraining and study its performance.

2 Methods

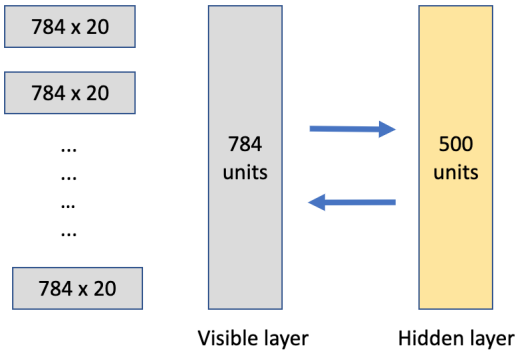
In order to reach the intended goals, the different methods and algorithms were coded in Python, a wide-spread programming language. Several libraries were also imported and used:

- **Numpy** to handle multi-dimensional arrays
- **Matplotlib**: Pyplot to plot graphs and Animation to animate figures
- **Random** to handle with randomisation of introduce special distribution of data
- **MNSIT** as a dataset of handwritten digits to train and validate the neural network

3 Results and discussion

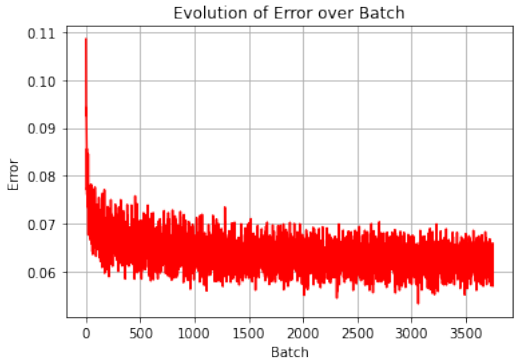
3.1 RBM for recognising MNIST images

The goal is to train an RBM composed of a 784 units visible layer (which corresponds to the input image) and of a 500 units hidden layer (Figure 1). The MNIST dataset used is divided in minibatches of 20 units for computational purposes.



Figur 1: Architecture of the RBM

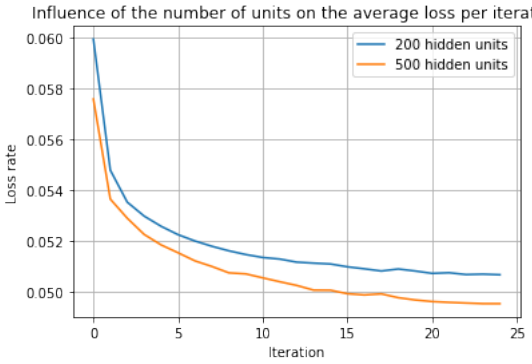
To measure the stability of the units and the convergence of the training, the error rate between the input minibatch of images and its reconstructed images is plotted (Figure 2)



Figur 2: Error rate between the input images and the reconstructed images over batch

- The error rate of Figure 2 converges to 6%: testifies accuracy, stability and convergence of the learning.

Then, the influence of the number of hidden units is analyzed on Figure 3.

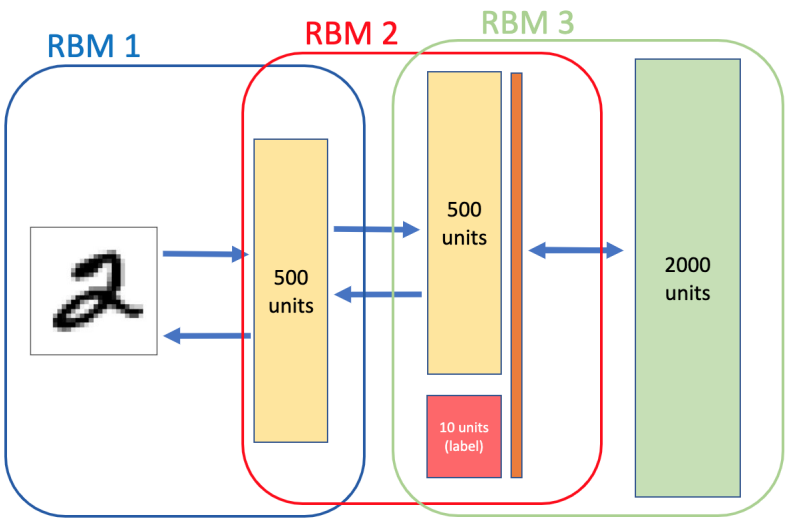


Figur 3: Error rate between the input images and the reconstructed images over the epochs, for 200 hidden units and 500 hidden units

- Having 500 hidden units appears to lead to better accuracy than 200 units.

3.2 Deep Belief Nets - greedy layer-wise pretraining

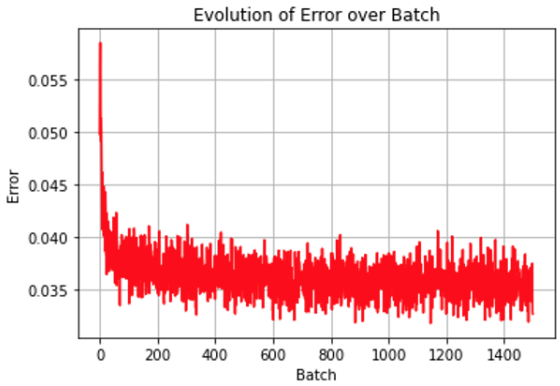
Given the work previously done on RBMs, a deeper network architecture is then built: Deep Belief Nets, which consists in a pile of different RBMs (Figure 4).



Figur 4: Architecture of the DBN used

Just like before, the convergence and stability of the network is monitored by plotting the reconstruction loss of the top RBM in the stack after its training. (5).

- The reconstruction loss converges to 3.5% which indicates convergence of

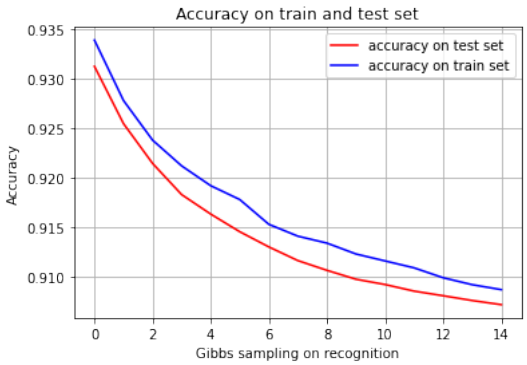


Figur 5: Reconstruction loss of the top RBM

the learning process.

3.2.1 Image recognition

A first purpose of such an architecture is image recognition. After the training of the network, the training accuracy and the test accuracy is plotted according to the Gibbs sampling advancement (Figure 6), to measure the generalization ability of the network. One can see that the accuracy of the image recognition (from the samples of the MNIST dataset) converges to approximately 90% for both testing and training set, demonstrating the great performance of such an architecture.

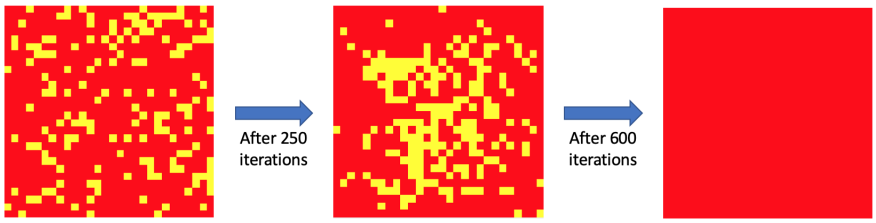


Figur 6: Training and testing error over the Gibbs sampling advancement

3.2.2 Image generation

A second relevance of a DBN is the generation of images: given the trained network with digits, it is possible to äskthe network to generate different digits.

A first try was to ask the network to generate a ≪0≫digit and giving it a random image as input. The result can be seen on Figure 7: the network doesn't manage to build a ≪0≫after 600 iterations, even if one can see that the pixels tend to gather themselves at the center of the image. After a high number of iterations, the totality of the pixels are even turned off.



Figur 7: Attempt to generate a ≪0≫digit from a random input image

A second try was to as an input image another digit which was previously learned by the network. On Figure 8, the network is asked to generate a ≪2≫digit with a ≪0≫digit as input: after only few iterations, the DBN manages to do so. All digits were able to be built thanks to this method.

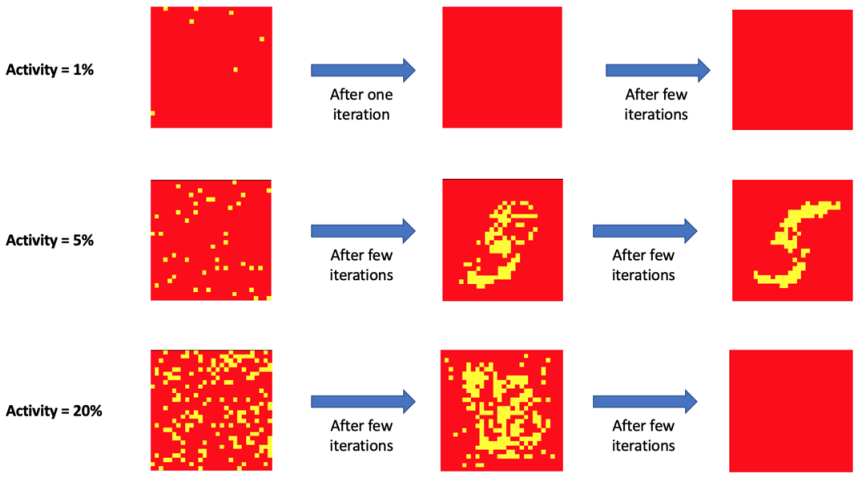
Figur 8: Attempt to generate a ≪2≫digit from a ≪0≫digit input image

Finally, the influence of the activity of the input image (rate of activated pixels) has been studied (Figure 9).

- A very low activity leads too the pixels being all desactivated after one iteration
- A 5-% activity allows to build the expected digit. It allowed during the lab to generate all digits asked accurately.
- A high activity leads to the non-convergence of the network

Note: when trying a 10% activity, the network were able to build few digits but not all of them.

As a conclusion, it seems that random images with controlled activity or samples from the dataset (different from the one aimed to be built) as an input to generate images seem to be the right strategy.



Figur 9: Influence of the activity of a random input image to generate a ≪5≫digit

4 Final remarks

To conclude, this work allowed to build RBMs and DBNs networks. Their learning and accuracy were monitored qualitatively and quantitatively and once these architectures were trained thanks to the MNIST dataset and implemented, it enabled to conduct image recognition and image generation with success.