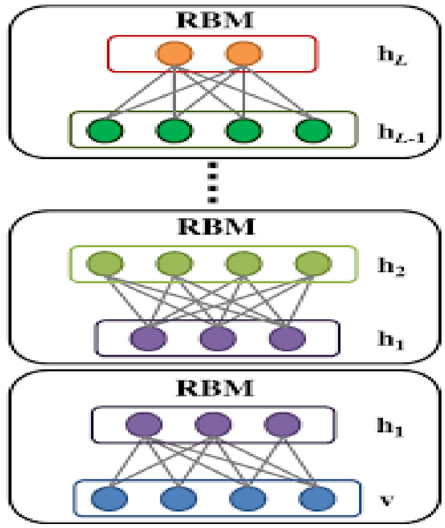

Deep Learning II

M2 Data Science

Project Report :
Restricted Boltzmann Machine (RBM)
Deep Belief Network (DBN)

Authors

Sajjad Mahdavi
Marvin Tordjman
Yanis Fellah

Professors

Yohan Petetin

March 25, 2023

Contents

1 Introduction

The objective of this project is to implement from scratch the Restricted Boltzmann Machine (**RBM**) and Deep Belief Network (**DBN**) models. In a first step, the performance of the two networks in generating alphabets (similar to the alphabets present in the AlphaDigits dataset) for different numbers of neurons, layers and number of characters to be learned will be presented. In a second step we will compare the classification performance on the MNIST dataset of a fully connected layer pre-trained with DBN and a fully connected layer initialized at random. Finally, we will look at the performance of RBM, DBN and VAE in generating digits.

2 Study on Binary AlphaDigit

In this section, we will look in more detail at the alphabets generated by RBM and DBN for different numbers of neurons, layers and number of characters to be learned.

2.1 RBM

2.1.1 Number of Neurons

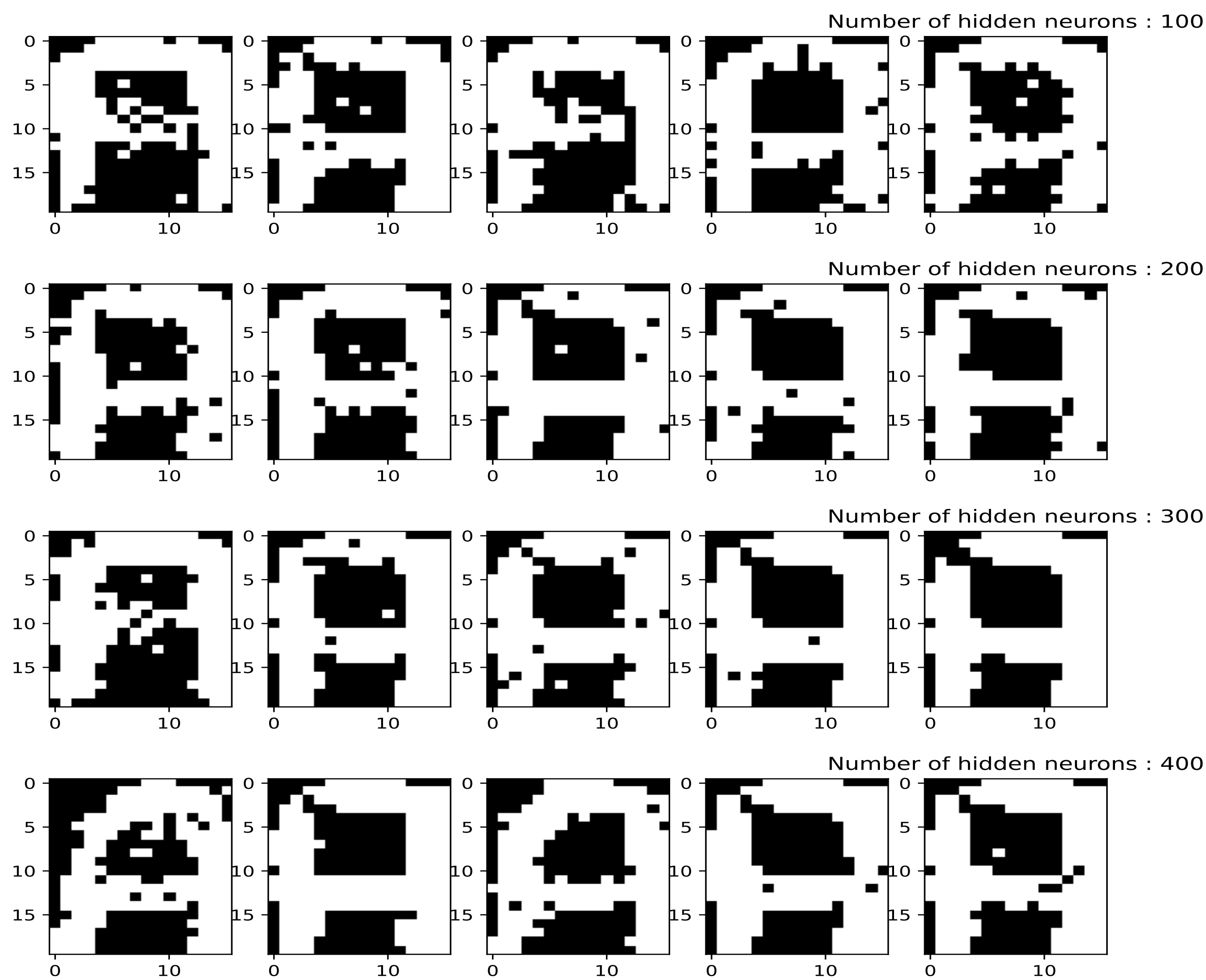

Figure 1: Generated AlphaDigits by RBM for different number of hidden neurons

Here we have trained our RBM on a single character ('a'). We can see (Fig 1) that with 200 neurons, we manage to generate a letter 'a' which is acceptable and increasing the number of neurons does not really help the cleanliness of the letter.

2.1.2 Number of Characters

In the following figure (Fig 2) we can see that if wee train the RBM on two characters you can almost see the two letters 'a' and 'b' (not very clear as well) but when we train on more than two letters it is almost impossible to generate any other letter than 'b' and it is not clear at all what kind of alphabets are generated. So we can say that when you train an RBM on more than two letters, the network can't understand the characteristics of each letter.

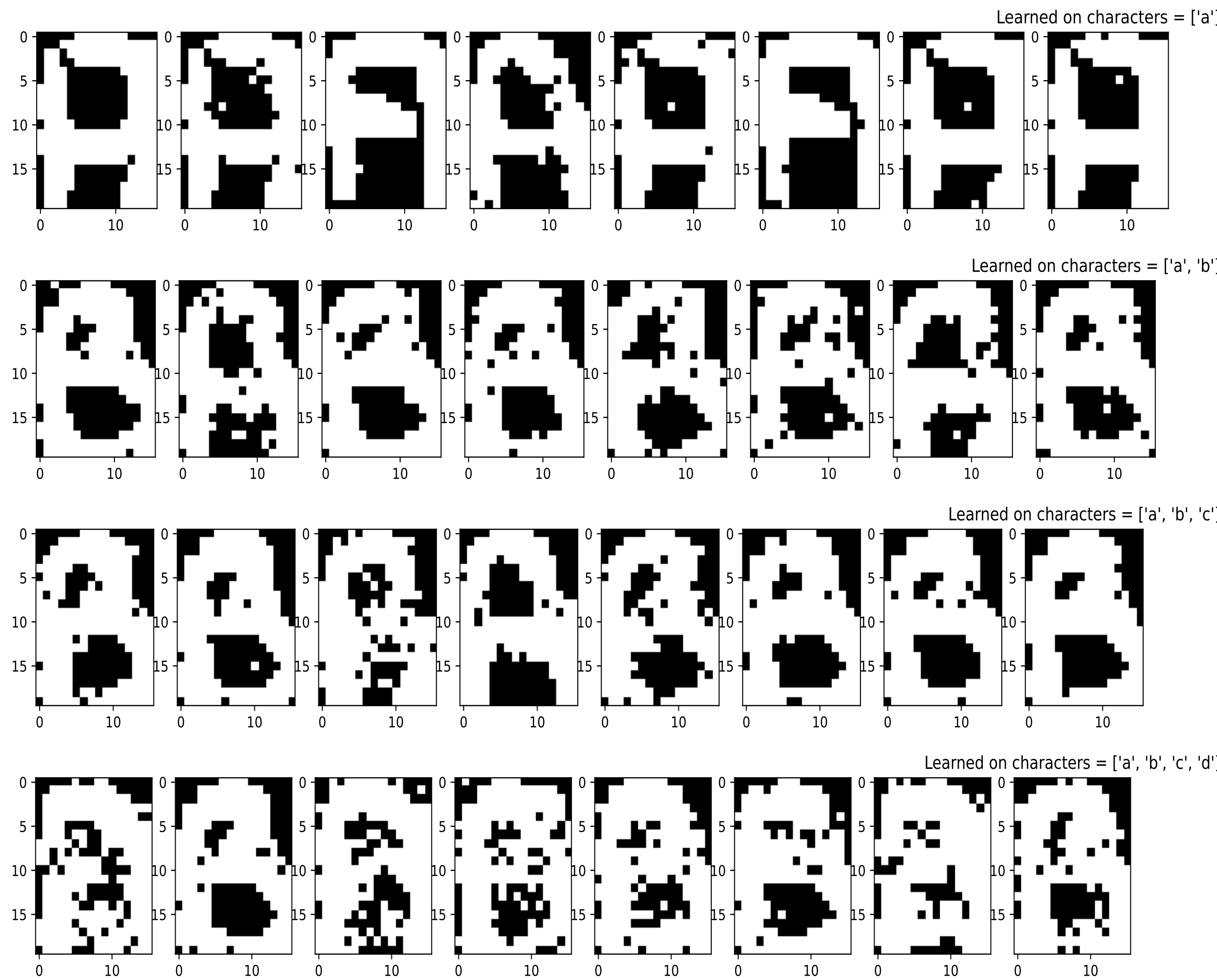

Figure 2: Generated AlphaDigits by RBM for different number of character

2.2 DBN

2.2.1 Number of Neurons

Here we trained our DBN on a single character ('a'). We can see (Fig 3) that with 200 neurons, we manage to generate an acceptable letter "a" and that increasing the number of neurons does not really improve the cleanliness of the letter. Comparing to the images generated by RBM, here we can see that for the same number of neurons the images generated by DBN are sharper than those generated by RBM.

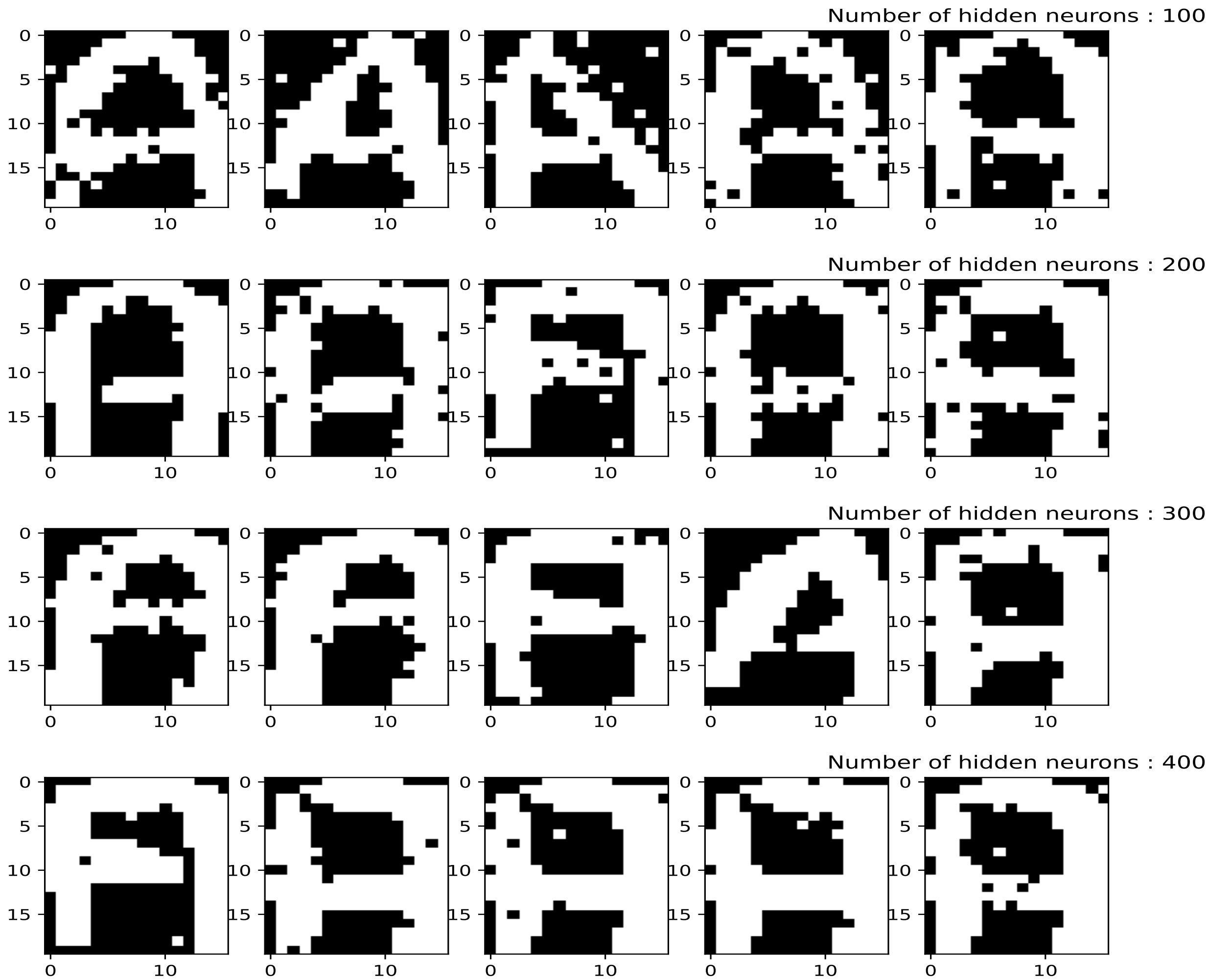

Figure 3: Generated AlphaDigits by DBN for different number of hidden neurons

2.2.2 Number of layers

Here we set the number of hidden neurons to 200 and we train our network on a single character and we change the number of layers. Here we put the images generated by networks with different numbers of layers. We can see (Fig 4) that for 5 layers, we have the best result but we have the impression that the more we increase the number of layers, the more the generated images become unreadable.

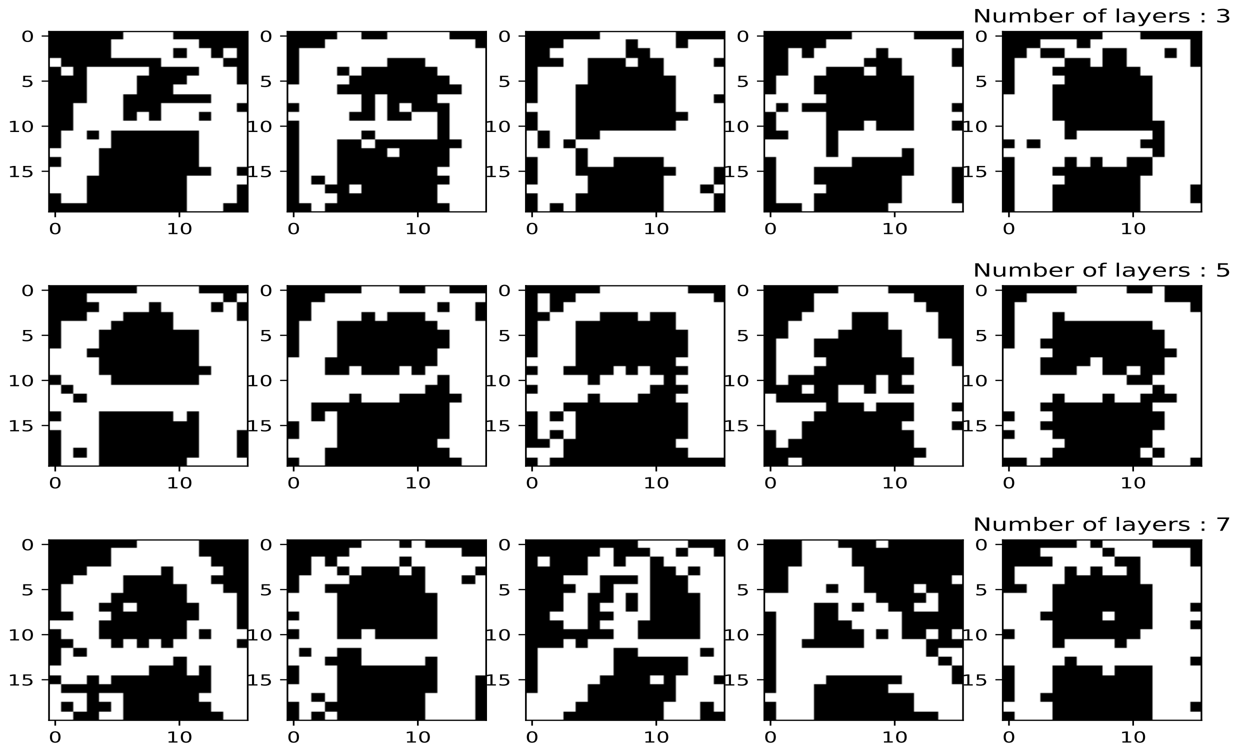

Figure 4: Generated AlphaDigits by DBN for different number of layers

2.2.3 Number of training images

One can see below (Fig 5) the images generated by DBN which has been trained on more than one letter are much sharper than a same RBM trained on several characters. For example, for the DBN network trained on 4 characters we can see the three letters ('a', 'b' and 'd') out of four.

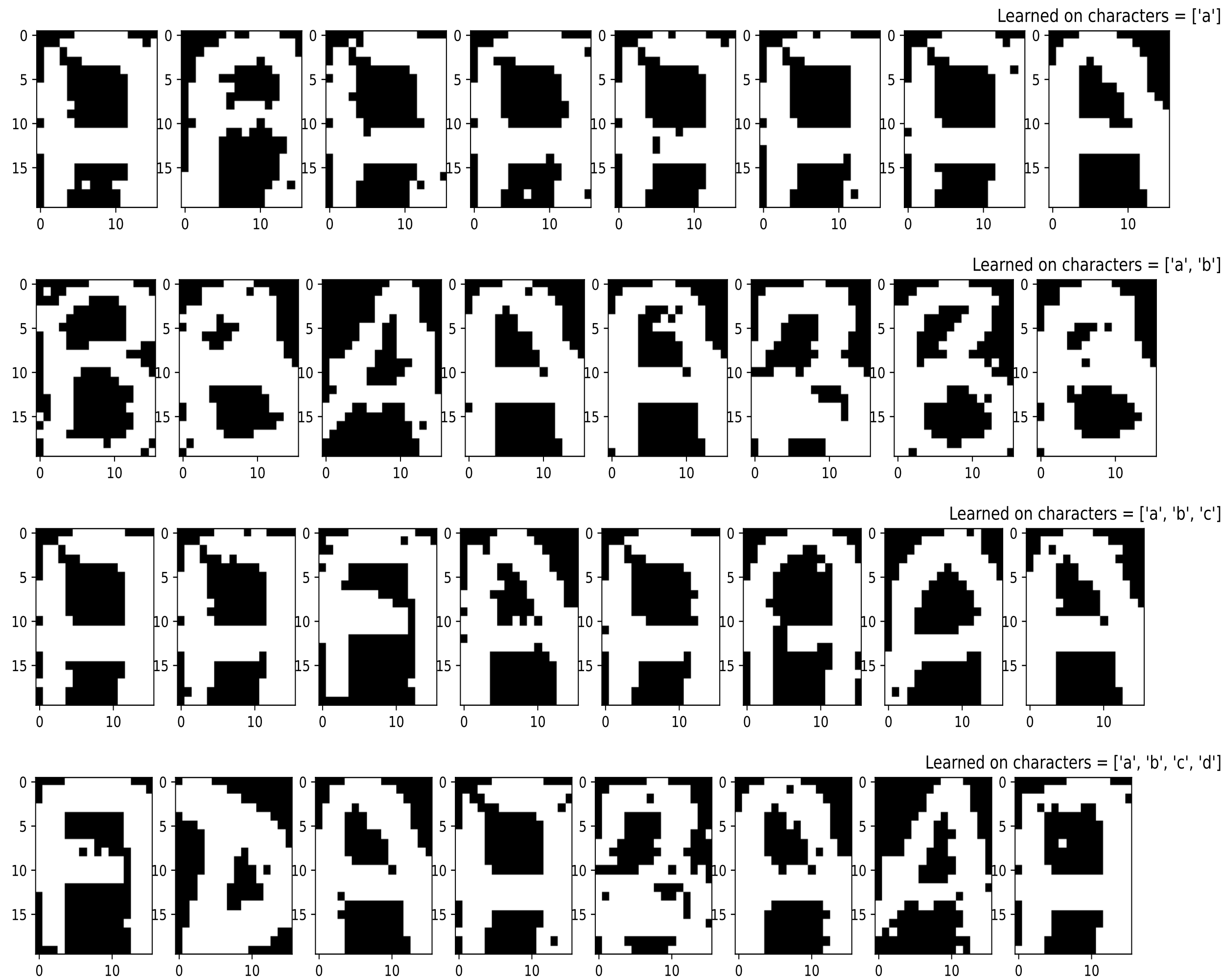

Figure 5: Generated AlphaDigits by DBN for different number of character

3 Study on MNIST

In this section we will compare the classification performance of a fully connected neural network pre-trained with a DBN and one where the weights are randomly initiated.

3.1 Number of Layers

First, we keep the number of neurons (200 neurons for each layer) and the number of training images (60,000 training samples) unchanged. After we change the number of layers to different values (2, 3 and 5 layers). We plot on Fig 6 the error rate on the training data and the test data. We can see that the higher the number of layers, the lower the error rate for the pre-trained Network. On the other hand, we can see that on the test data, the network pre-trained with DBN achieves a lower error rate than the untrained network.

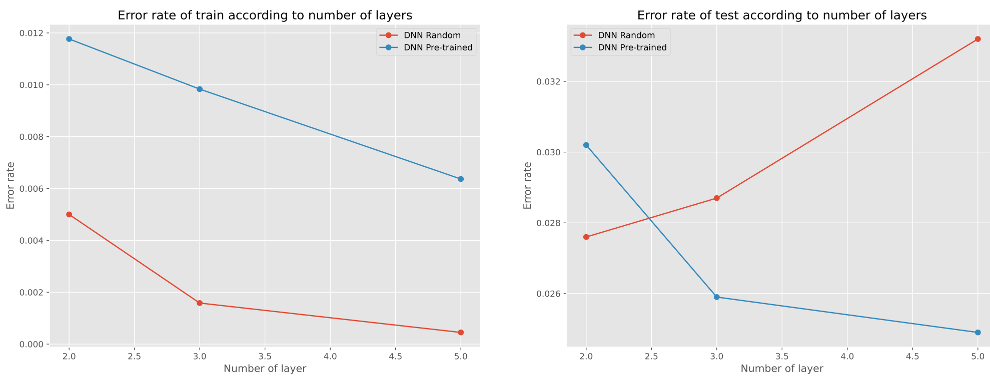

Figure 6: Error rate of train according to number of layers

3.2 Number of hidden neurons

In the next step, the number of layers (2 layers) and the number of training images (60,000 training samples) remain unchanged, and the number of hidden neurons is fixed at different values (100, 300 and 700). We observe (Fig 7) in the case of pre-trained with DBN, the more we increase the number of neurons the lower the error is (this is not the case for the randomly initialized network where the error stagnates). In general the pre-trained network easily beats the randomly initialized network from 300 neurons.

Figure 7: Error rate of train according to number of hidden neurons

3.3 Number of training images

Finally, we keep the number of layers (2 layers) and the number of neurons (200 neurons for each layer) unchanged, and we change the number of images on which we train the neural network. It is clear (Fig 8) that the larger the number of images in the training data, the lower the error rate. On the other hand, we can point out that the error rate is less important for the neural network pre-trained by DBN.

Figure 8: Error rate of train according to number of training images

4 Comparing Generating power of RBM, DBN and VAEs

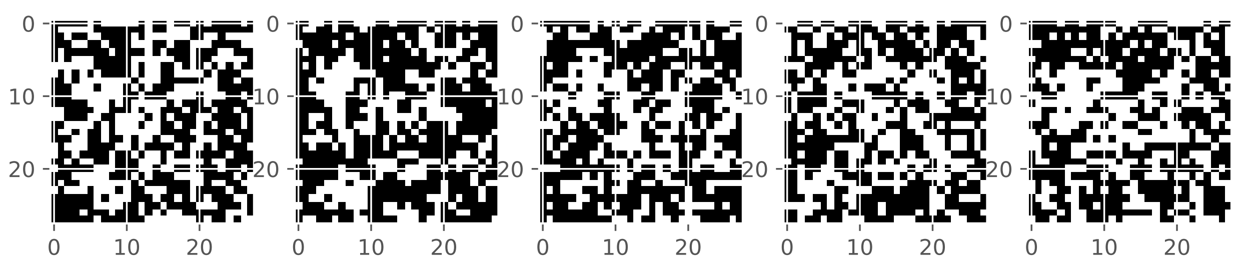

Figure 9: Generating images by *RBM*

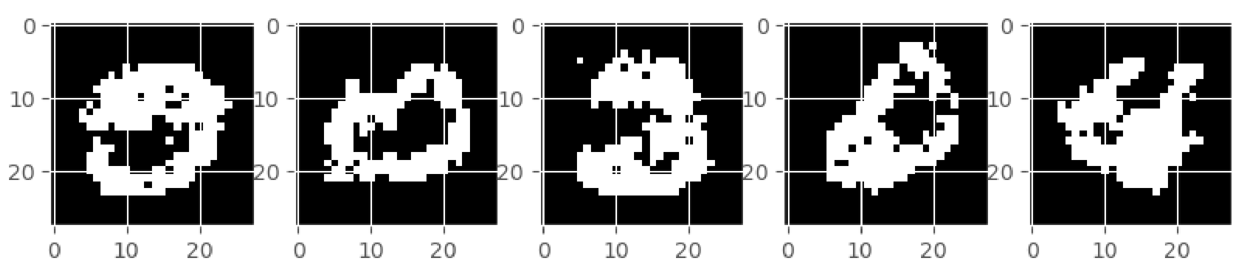

Figure 10: Generating images by *DBN*

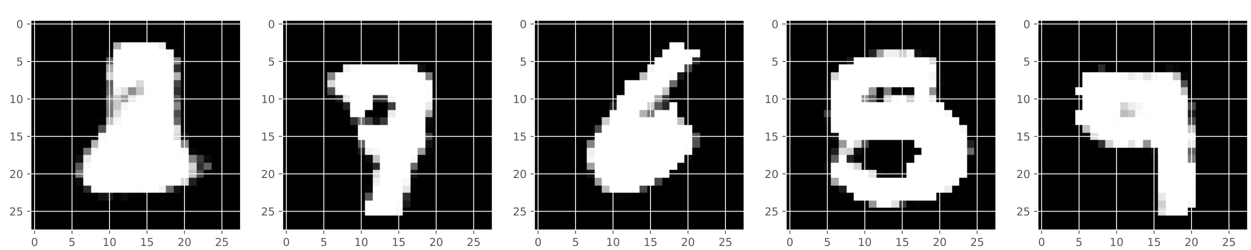

Figure 11: Generating images by *VAE*

In this part, we presented (Fig 11) 5 digits generated by a RBM, DBN and VAEs. We can see clearly that the images generated by RBM and DBN are not readable at all. This confirms the confirmation we had before on the AlphaDigits dataset when we trained the networks on several characters. However we can see that the images generated by VAEs are quite good and quite sharp. This shows the power of VAEs compared to the other two architectures.

5 Conclusion

In conclusion, we saw that the performance of neural networks with a generative pre-training process are improved in most cases. However, the pre-training can require a significant amount of time, especially when the network is complex.

That's why, even though the performances are improved, we have to choose a trade-off between running time and accuracy.