M2 Data sciences

Deep Learning

Generative modeling project

Manuel NKEGOUM, Salim ABDOU DAOURA

Under the supersivion of:

Yohan PETETIN

Academic year: 2023/2024

Contents

List of Figures

Introduction

Within the framework of the "Deep Learning II" project at the Institut Polytechnique de Paris, we are undertaking an in-depth exploration of the capabilities and performances of deep neural networks in the task of handwritten digit classification. This project aims to comprehensively compare the performances between a pre-trained deep neural network and a randomly initialized network, both constructed on the basis of Restricted Boltzmann Machines (RBMs). Our study focuses on various key parameters such as the volume of training data, the number of network layers, and the number of neurons per layer, in order to determine their impact on the accuracy rate.We used two main datasets: a database of small-dimension images for initial tests and the MNIST database, a standard benchmark in machine learning, to assess the effectiveness of unsupervised pre-training.

This document presents an introduction to the data used, describes the elementary functions to be implemented, and guides the process of constructing and evaluating deep neural networks and Deep Belief Networks. Additionally, we present a comparative study of the generative capabilities of RBMs, DBNs, and Variational Autoencoders (VAEs).

Alpha-digits

Alpha-Digits is a collection of small-scale images that includes numbers and letters, providing a rapid testing framework for our models before moving on to more complex datasets such as MNIST.For the experiments on this database, our aim was to evaluate the ability of our models to learn and generate handwritten characters.

Restricted Boltzman Machine

A restricted Boltzmann machine RBM (also called a restricted Sherrington–Kirkpatrick model with external field or restricted stochastic Ising–Lenz–Little model) is a generative stochastic artificial neural network that can learn a probability distribution over its set of inputs. RBMs have found applications in dimensionality reduction, classification, collaborative filtering, feature learning, topic modelling and even many body quantum mechanics. They can be trained in either supervised or unsupervised ways, depending on the task.
Energy Based Model
The standard type of RBM has binary-valued (Boolean) hidden and visible units, and consists of a matrix of weights W of size $m \times n$. Each weight element $w_{i,j}$ of the matrix is associated with the connection between the visible unit v_i and the hidden unit h_j. In addition, there are bias weights a_i for v_i and b_j for h_j. Given the weights and biases, the energy of a configuration (pair of boolean vectors) (v,h) is defined as:

$$E(v,h) = E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j$$

Contrastive Divergence

Since RBMs are undirected, they don't adjust their weights through gradient descent and backpropagation. They adjust their weights through a process called contrastive divergence. At the start of this process, weights for the visible nodes are randomly generated and used to generate the hidden nodes. These hidden nodes then use the same weights to reconstruct visible nodes. The weights used to reconstruct the visible nodes are the same throughout. However, the generated nodes are not the same because they aren't connected to each other.

Experiments and Results

Implementation

We have designed and developed a RBM class in Python, using the PyTorch library. The aim of this class is to provide a basic structure for the creation, training and generation of data by an RBM. A RBM then consist of an Linear Layer with for visible (a) and hidden (b) units initialized at 0 and a weights's matrix. The class also implements methods like Gibbs sampling, training via contrast divergence, and new data generation.

Reconstruction

The reconstruction error of the RBM was evaluated for different numbers of classes used for training. Figure 1 shows the reconstructed images obtained using an RBM trained on a single class (the digit "0"). The RBM was able to accurately reconstruct the images, achieving a reconstruction error (Mean squared Error) of 2.6×10^{-4}.When the number of classes

used for training was increased to include all digits from 0 to 9, the RBM was still able to reconstruct images of the digit "0" with reasonable accuracy, as shown in Figure 2. However, the reconstruction error increased to 4.7×10^{-3}, indicating that the performance of the RBM is affected by the number of classes used for training.To improve the performance of the RBM, the number of hidden units was increased to 300 while keeping the number of training epochs constant. This resulted in a significant improvement in the reconstruction error, which decreased to 3.1×10^{-5}.
These results suggest that the performance of the RBM is dependent on various factors, including the number of hidden units, the number of training epochs, and the number of classes used for training. Further experiments were conducted to analyze the effects of these factors on the performance of the RBM, as described in the following sections.

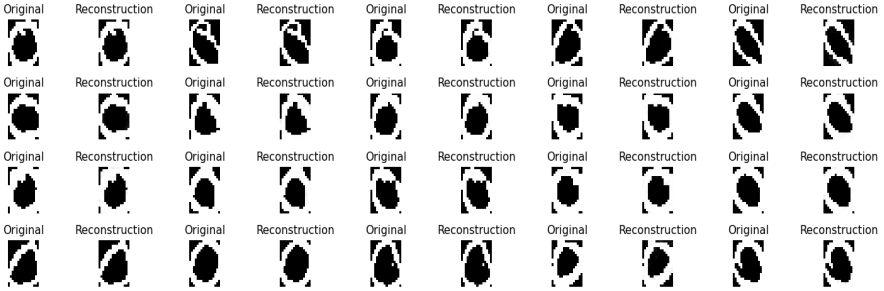

Figure 1 – Reconstructed images by a RBM - one class

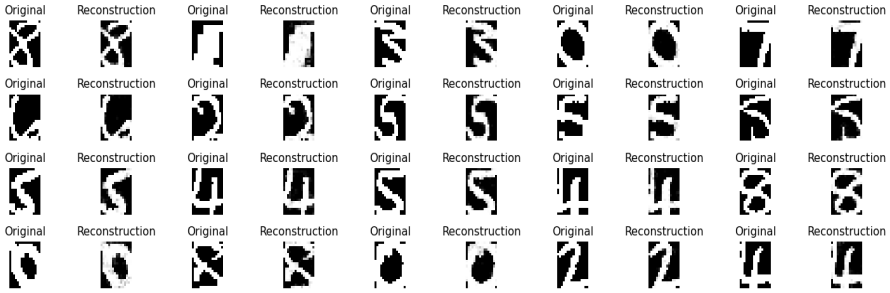

Figure 2 – Reconstructed images by a RBM - ten class

Generation

In this section, we present the results of the image generation experiment using the RBM, trained with varying numbers of classes. The generation is performed using 1000 Gibbs sampling iterations.Firstly, we trained the RBM with only one character (the digit "0") and evaluated its performance. We fixed the number of hidden units to 100 and the number of training epochs to 1000. Figure 3 shows the generated images by the RBM, and we can observe that the digit "0" is well-generated.Next, we increased the number of classes used for training to include all digits from 0 to 9. However, this time we set the number of hidden units to 300 and the number of training epochs to 5000. Figure 4 shows the generated images by the RBM, and we can see that they aren't of good quality. This suggests that increasing the number of hidden units and training epochs does not necessarily lead to better performance in terms of quality in the generated images.

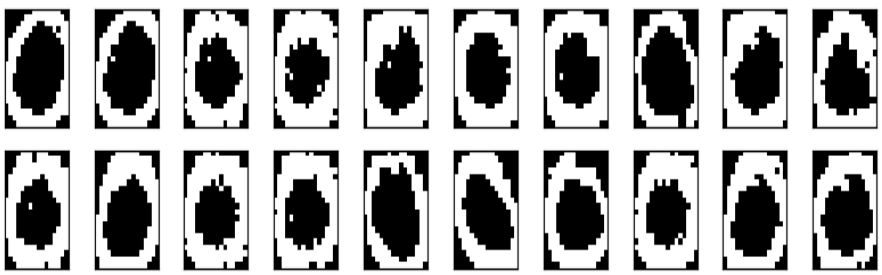

Figure 3 – Generated images by a RBM - one class

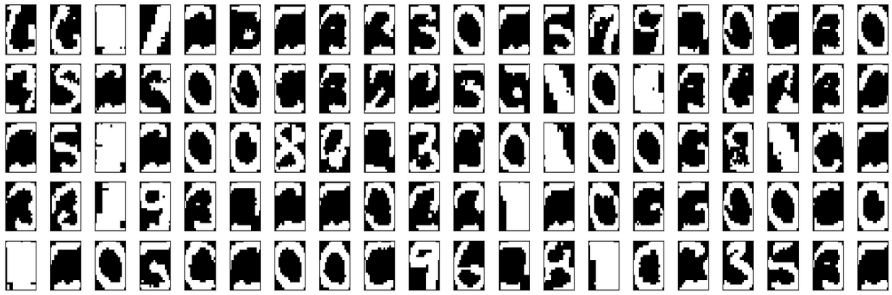

Figure 4 – Generated images by a RBM - ten class

Limits study

In order to assess the proficiency of our RBM model in accurately reconstructing input data and generating new data based on learned hidden representations, we carried out a sequence of experiments on the database. This evaluation process is essential in determining the effectiveness of our RBM model in capturing the underlying patterns in the data. To measure the quality of the reconstructions, we utilized a quantitative metric, namely the mean square error (MSE) between the original and reconstructed images. On the other hand, the quality of the generated data was evaluated through visual inspection.The tables 1 and 2 resume the hyperparameters of our studies.

— **Influence of the number of classes** To investigate the impact of the number of classes on the performance of the RBM, we conducted an experiment where we fixed the number of hidden units to 100 and increased the number of classes in our dataset. We then analyzed the reconstruction error of the RBM as a function of number of classes, as shown in Figure 5.As can be observed, the number of classes in our dataset significantly influences the performance of our model, with the error increasing as the number of classes is increased. Moreover, when using more than two classes, the generated images aren't of good quality , as depicted in Figure 6. This suggests that the RBM struggles to capture the variability in the data when dealing with a larger number of classes. Therefore, it is important to carefully consider the number of classes when training an RBM for a specific task.

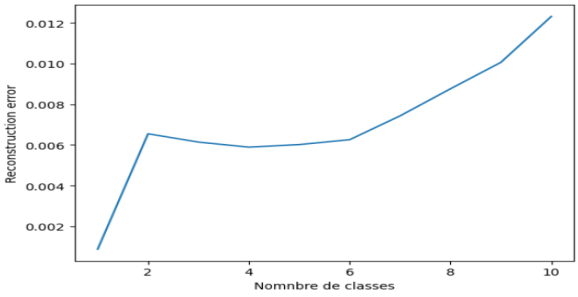

Figure 5 – reconstruction error of a RBM as a function of number of classes

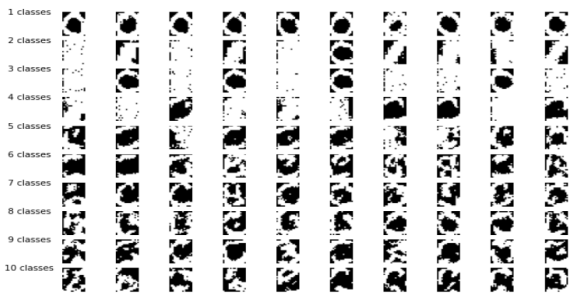

Figure 6 – Generated images of a RBM as a function of number of classes

— **Influence of the number of training epochs**
To investigate the influence of the number of training epochs on the performance of the RBM, we conducted an experiment where we fixed the number of hidden units and the number of classes in the dataset (20 classes). We then

varied the number of epochs used for the contrastive divergence algorithm, which is a method used to train RBMs.The results of the experiment are shown in Figure 7, where we can see the reconstruction error of the RBM as a function of the number of training epochs. As the number of epochs increases, the reconstruction error decreases, indicating that the RBM is learning better representations of the input data.
However, as shown in Figure 8, the generative performance of the RBM is still far from optimal, even with a larger number of training epochs. One possible solution to improve the generative performance of the RBM is to use multiple layers of RBMs, which can capture more complex dependencies in the data. Overall, the results suggest that the number of training epochs is an important factor to consider when training RBMs, but other factors such as the complexity of the data should also be taken into account.

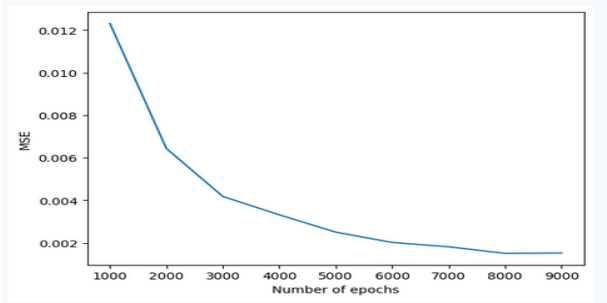

Figure 7 – reconstruction error of a RBM as a function of training epochs

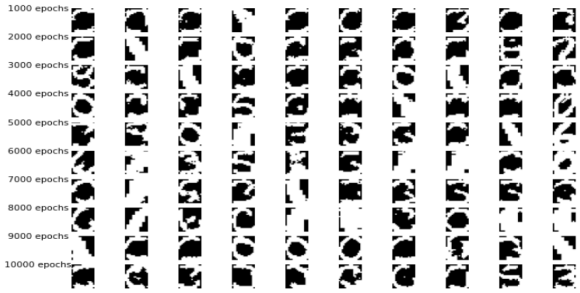

Figure 8 – generated images a RBM as a function of training epochs

Hyperparameter	**Value**
gibbs steps	1
hidden units	100
batch size	32
training epochs	1000
learning rate	0.1

Table 1 – RBM Experiment 1

Hyperparameter	**Value**
gibbs steps	1
hidden units	100
batch size	32
number of classes	10
learning rate	0.1

Table 2 – RBM Experiment 2

Deep Belief Network

Deep Belief Networks (DBN) are sophisticated artificial neural networks used in the field of deep learning, a subset of machine learning. They are designed to discover and learn patterns within large sets of data automatically.DBN are composed of multiple layers of RBM . Each layer in a DBN aims to extract different features from the input data, with lower layers identifying basic patterns and higher layers recognizing more abstract concepts. This structure allows DBNs to effectively learn complex representations of data, which makes them particularly useful for tasks like image and speech recognition, where the input data is high-dimensional and requires a deep level of understanding.The tables 3,4 and 5 resume the hyperparameters of our studies.

Greedy algorithm

A DBN is a stack of RBMs is trained in a greedy and sequential manner to capture the representation of hierarchy of relationships within the training data. A model of distribution between observed vector x and l hidden layer hk is as follows:

$$P(x, h_1, \ldots, h_l) = \left(\prod_{k=0}^{l-2} P(h_k | h_{k+1}) \right) P(h_{l-1}, h_l)$$

where the distribution for visible units conditioned on hidden units of a RBM block at level k is represented by $P(h_{k-1}, h_k)$, and the visible-hidden joint distribution of top-level RBMs is represented by $P(h_{l-1}|h_l)$.This layer-by-layer training is a greedy approach, meaning each RBM learns the best representation of its input for the current level, without considering the overall goal of the entire network.

Experiments and Results

For the deep belief networks (DBNs) part of our project, we shifted our focus from the reconstruction aspect of the model to the generative aspect.The implementation was made in pytorch and consisted of a list of RBMs. By training each layer of the DBN greedily and using the output of one layer as input to the next, the DBN can learn a hierarchical representation of the input data.In our experiments, we aimed to evaluate the generative capabilities of DBNs in modeling complex data distributions. We used a similar setup as in our RBM experiments.We conducted a series of experiments to evaluate the influence of various factors on the generative performance of the DBN, such as the number of layers, the number of hidden units per layer, and the number of classes (characters) .Our results showed that the DBN was able to generate more diverse and realistic samples compared to the RBM, demonstrating the effectiveness of using a deep generative model for complex data distributions.

Preliminary results

The generation is performed using 1000 Gibbs sampling iterations.We trained the RBM with only five characters (digits ”0-4”) and evaluated its performance. We fixed the number of hidden units to 300,the number of layers to 3 and the number of training epochs to 5000. Figure 9 shows the generated images by the DBN. We can see that the quality of the images improved greatly compared with those obtained with a single RBM.

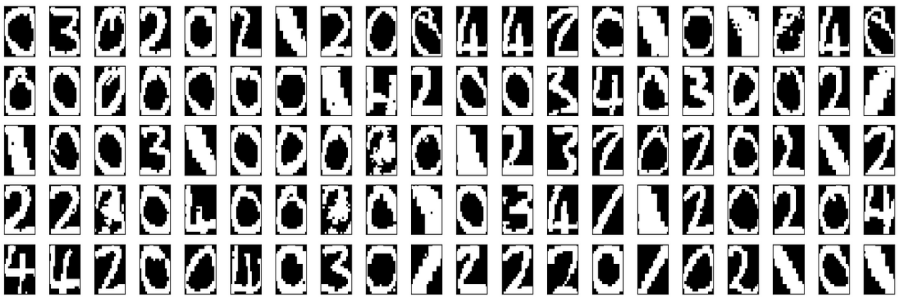

Figure 9 – Generated images by a DBN

Limits study

To evaluate the DBN's reconstruction and generation capabilities, we conducted the same series of experiments on the database as with the RBM except that we focus on the generative part of the model.

— **Influence of the number of classes** To investigate the influence of the number of classes on the generative performance of a DBN, we conducted an experiment where we varied the number of classes in the dataset while keeping other factors constant. We trained the DBN on subsets of the MNIST dataset with varying numbers of classes, ranging from 5 to 15 classes.

Our results showed that as the number of classes increased, the generative performance of the DBN decreased. This is because the model has to learn a more complex data distribution with more classes, which makes it harder to generate realistic samples.

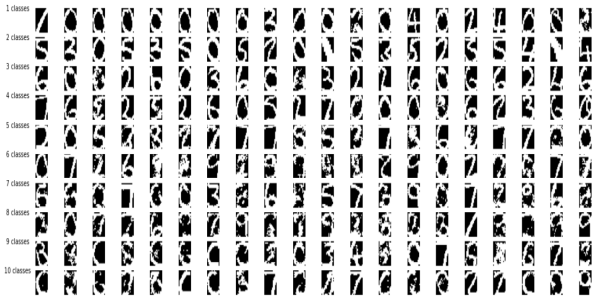

Figure 10 – Generated images of a DBN as a function of the number of classes

— **Influence of the number of layers** The influence of the number of layers in a DBN is a crucial factor in achieving good generative performance. By stacking multiple layers of RBMs, a DBN can learn a hierarchical representation of the input data, where each layer captures increasingly complex and abstract features.To investigate the effect of the number of layers on the generative performance of a DBN, we conducted experiments where we varied the number of layers while keeping other parameters fixed. We found that as the number of layers increased, the quality of the generated samples slightly improved that's because we have a small dataset so 3 or 5 layers at most should be sufficient.

Figure 11 – Generated images of a DBN as a function of the number of layers

— **Influence of the number of hidden units per layer** Next, we studied the influence of the number of neurons per layer on the quality of the generated images. We trained DBNs with a fixed number of layers and varied the number of neurons per layer. We found that increasing the number of neurons per layer led to an improvement in the quality of the generated images, up to a certain point. However, beyond a certain number of neurons, the improvement in image quality became negligible. This suggests that there is an optimal number of neurons per layer for generating high-quality images.

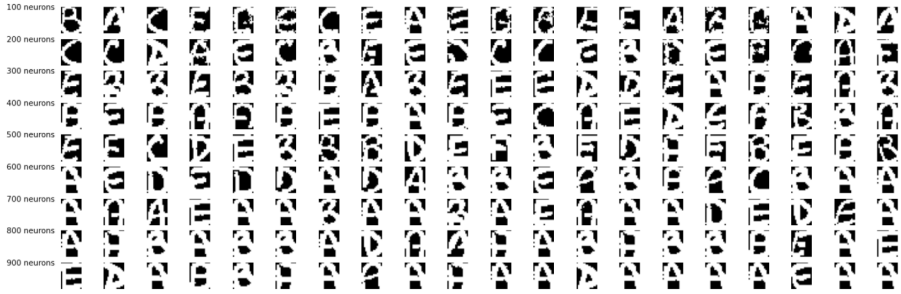

Figure 12 – Generated images of a DBN as a function of the number of hidden units per layer

Hyperparameter	**Value**
gibbs steps	1
hidden units	300
batch size	32
training epochs	5000
learning rate	0.1
number of layers	3

Table 3 – DBN Experiment 1

Hyperparameter	**Value**
gibbs steps	1
hidden units	100
batch size	32
training epochs	5000
learning rate	0.1
number of classes	10

Table 4 – DBN Experiment 2

Hyperparameter	**Value**
gibbs steps	1
number of layers	2
batch size	32
training epochs	5000
number of classes	10
learning rate	0.1

Table 5 – DBN Experiment 3

MNIST

Deep Neural Networks

In this section of the project, we delve into the application of Deep Neural Networks (DNN) on the MNIST dataset. The MNIST dataset, a benchmark for machine learning algorithms, consists of 70,000 images of handwritten digits (0-9), where each image is 28x28 pixels. Two networks were instantiated for this purpose. The first network was pretrained in an unsupervised manner using the greedy algorithm of Deep Belief Networks (DBNs). The greedy layer-wise unsupervised training strategy of DBNs allows for more efficient training and better initial weights for the subsequent supervised fine-tuning phase.The second network, on the other hand, was a standard DNN, initialized with random weights. This network served as a baseline for comparison, highlighting the benefits of unsupervised pretraining.

Following the unsupervised pretraining phase, both networks were trained in a supervised manner. The pretrained DBN was fine-tuned using backpropagation, while the standard DNN was trained from scratch using the same algorithm. The goal of this supervised training was to classify the handwritten digits correctly.The performance of the two networks was then evaluated on a test set, which consisted of unseen images from the MNIST dataset. The error rate was used as the performance metric. The error rate is the proportion of the test set samples that were incorrectly classified by the network. The results of this experiment provided insights into the effectiveness of unsupervised pretraining in DNNs. It was expected that the DBN, which was initialized with pretrained weights, would outperform the standard DNN, demonstrating the benefits of unsupervised pretraining for deep learning models.The tables 11,7 and 8 resume the hyperparameters of our studies.

Experiments and results

— **Influence of the number of layers**

The first experiment aimed to study the effect of the number of layers on the performance of the networks.In this experiment, we varied the number of hidden layers in both networks and observed the corresponding changes in their error rates on the test set. The goal was to understand how the depth of a network affects its ability to learn complex representations of the data. It was expected that deeper networks would be able to learn more abstract and hierarchical representations, leading to better performance on the classification task.For each configuration of the number of layers, both networks were trained and evaluated in the same manner as described in the previous section. The error rates of the two networks were then compared to assess the effect of the number of layers on their performance.

The results of the first experiment provided in 13 revealed an interesting pattern in the performance of the two networks as the number of layers was varied. As expected, the error rate of the pretrained network generally diminished with the addition of more layers. This can be attributed to the fact that deeper networks have a higher capacity to learn complex and hierarchical representations of the data. The unsupervised pretraining phase of the pretrained network also likely provided it with a better initialization, making it easier to train and less prone to overfitting.In contrast, the error rate of the standard network increased with the addition of more layers. This can be explained by the fact that deeper networks are more difficult to train from scratch, due to issues such as vanishing gradients and the increased risk of overfitting. The random initialization of the standard network also likely put it at a disadvantage compared to the pretrained network, which had a more informed initialization.These results highlight the benefits of

unsupervised pretraining for deep learning models. By providing a good initialization and allowing for more efficient training, unsupervised pretraining can help to overcome the challenges associated with training deep networks. The results also underscore the importance of careful network design and regularization techniques, such as dropout and weight decay, to prevent overfitting in deep networks.

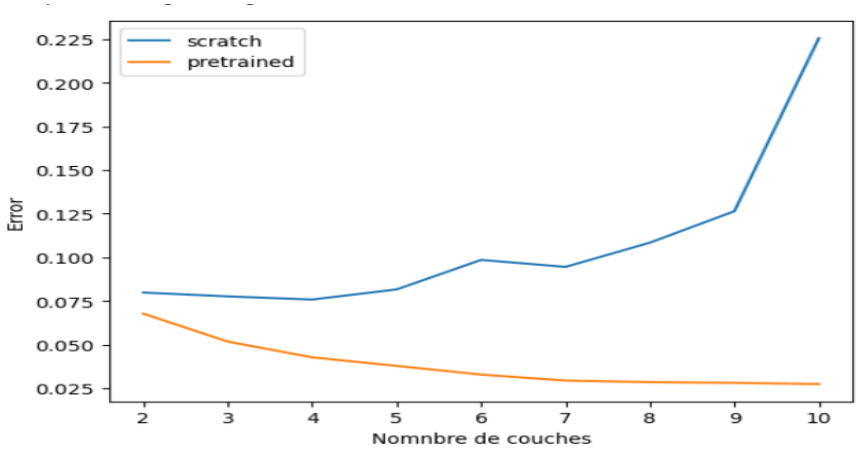

Figure 13 – Error rate of two models as a function of the number of layers

— **Influence of the number of neurons**

In the second experiment, we studied the effect of the number of neurons in each hidden layer on the performance of the two networks. We varied the number of neurons in each hidden layer while keeping the number of layers fixed, and observed the corresponding changes in the error rates on the test set.The goal of this experiment was to understand how the width of a network affects its ability to learn complex representations of the data. It was expected that wider networks would have a higher capacity to learn complex representations, but might also be more prone to overfitting.For each configuration of the number of neurons, both networks were trained and evaluated in the same manner as described in the previous section. The error rates of the two networks were then compared to assess the effect of the number of neurons on their performance.

The results of the second experiment provided in 14 revealed a clear advantage of the pretrained network over the standard network in terms of the effect of the number of neurons on performance. As expected, the error rates of both networks generally diminished with the addition of more neurons in each hidden layer. However, the pretrained network showed a more significant improvement in performance, reaching an error rate of less than 2% on the test set, while the standard network did not manage to reach an error rate below 3%.The superior performance of the pretrained network can be attributed to its unsupervised pretraining phase, which provided it with a better initialization and allowed for more efficient training. The pretrained network was able to take advantage of the increased capacity provided by the wider layers to learn more complex and abstract representations of the data. In contrast, the standard network likely suffered from the challenges associated with training wide networks from scratch, such as vanishing gradients and the increased risk of overfitting.

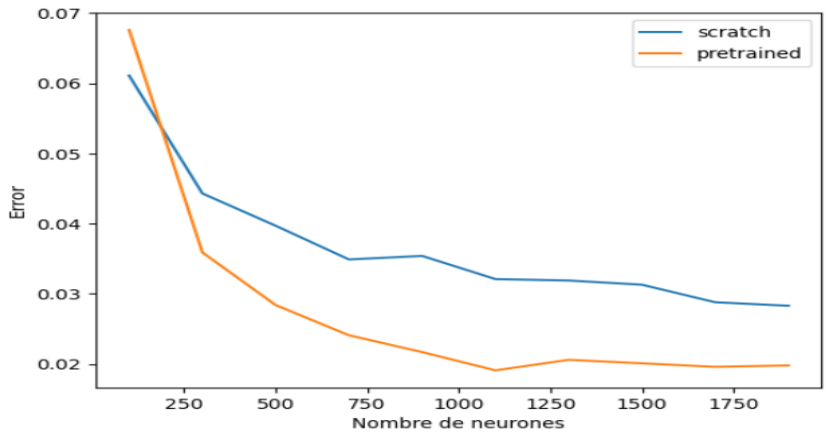

Figure 14 – Error rate of two models as a function of the number of neurons

— **Influence of the number of training examples**

In the third and final experiment, we studied the influence of the number of training examples on the performance of the two networks. We varied the size of the training set while keeping the number of layers and neurons fixed, and observed the corresponding changes in the error rates on the test set.The goal of this experiment was to understand how the amount of training data affects the performance of deep learning models. It was expected that larger training sets would lead to better performance, as they provide more information for the network to learn from. However, it was also expected that the improvement in performance would diminish as the size of the training set increases, due to the law of diminishing returns.For each configuration of the training set size, both networks were trained and evaluated in the same manner as described in the previous section. The error rates of the two networks were then compared to assess the effect of the training set size on their performance.

The results of this experiment provided in 15 showed that the error rates of both networks generally decreased as the size of the training set increased. However, the rate of decrease diminished as the size of the training set increased, indicating that there is a point of diminishing returns beyond which additional training data does not significantly improve performance. The pretrained network showed a more significant improvement in performance compared to the standard network, reaching an error rate of less than 2% on the test set with a moderate-sized training set, while the standard network did not manage to reach an error rate below 3% even with a larger training set.

These results highlight the importance of having a sufficient amount of training data for deep learning models to achieve good performance. However, they also suggest that there is a limit to the benefits of simply adding more training data, and that other factors such as network architecture and regularization techniques should also be considered to achieve optimal performance. The results also reinforce the advantages of unsupervised pretraining for deep learning models, particularly in scenarios where labeled training data is scarce.

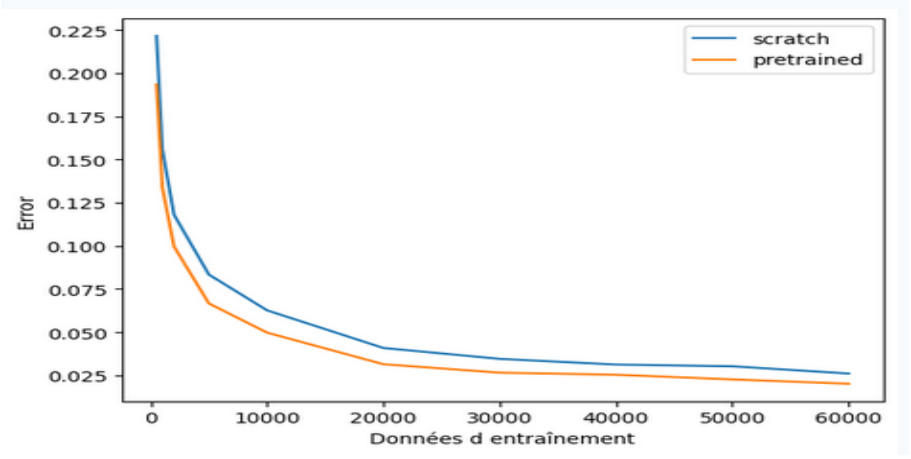

Figure 15 – Error rate of two models as a function of the number of training data

Hyperparameter	Value
gibbs steps	1
hidden units	300
pretraining batch size	128
pretraining epochs	100
pretraining learning rate	0.1
Training epochs	30
Adam's learning rate	$1e^{-4}$
Dropout	0.2
Activation	ReLU

Table 6 – DNN Experiment 1

Hyperparameter	Value
gibbs steps	1
Rbm layers	2
pretraining batch size	128
pretraining epochs	100
pretraining learning rate	0.1
Training epochs	30
Adam's learning rate	$1e^{-4}$
Dropout	0.2
Activation	ReLU

Table 7 – DNN Experiment 2

Hyperparameter	Value
gibbs steps	1
hidden units	200
pretraining batch size	128
pretraining epochs	500
pretraining learning rate	0.1
Training epochs	30
Adam's learning rate	$1e^{-4}$
Dropout	0.2
Activation	ReLU

Table 8 – DNN Experiment 3

Bonus : visual comparison

In this section, we conduct a visual comparison of images generated by a Variational Autoencoder (VAE), a Restricted Boltzmann Machine (RBM), and a Deep Belief Network (DBN). Our networks were trained on the MNIST dataset, comprising 60,000 images of handwritten digits (0-9), each sized 28x28 pixels.
To ensure a meaningful baseline for comparison, we endeavored to maintain a comparable number of parameters, approximately one million, across our various network architectures. The training parameters utilized are summarized in the tables below.

Hyperparameter	Value
gibbs steps	1
Number of parameters	942784
hidden size	1200
Training epochs	1000
Adam's learning rate	$1e^{-1}$
Batch size	512
Activation	ReLU

Table 9 – RBM parameters

Hyperparameter	Value
gibbs steps	1
Number of parameters	1413034
hidden size	[1000, 500, 250]
Training epochs	5000
Adam's learning rate	$1e^{-1}$
Batch size	128
Activation	ReLU

Table 10 – DBN parameters

Hyperparameter	Value
Number of parameters	1074980
hidden size	[512, 256, 2]
Training epochs	30
Adam's learning rate	$1e^{-3}$
Batch size	128
Activation	ReLU

Table 11 – VAE parameters

Restricted Boltzman Machine

The images generated by the RBM (Figure 16) exhibit low quality and do not allow for easy recognition of the different classes. The RBM lacks sufficient representation power to effectively handle the ten distinct digits.

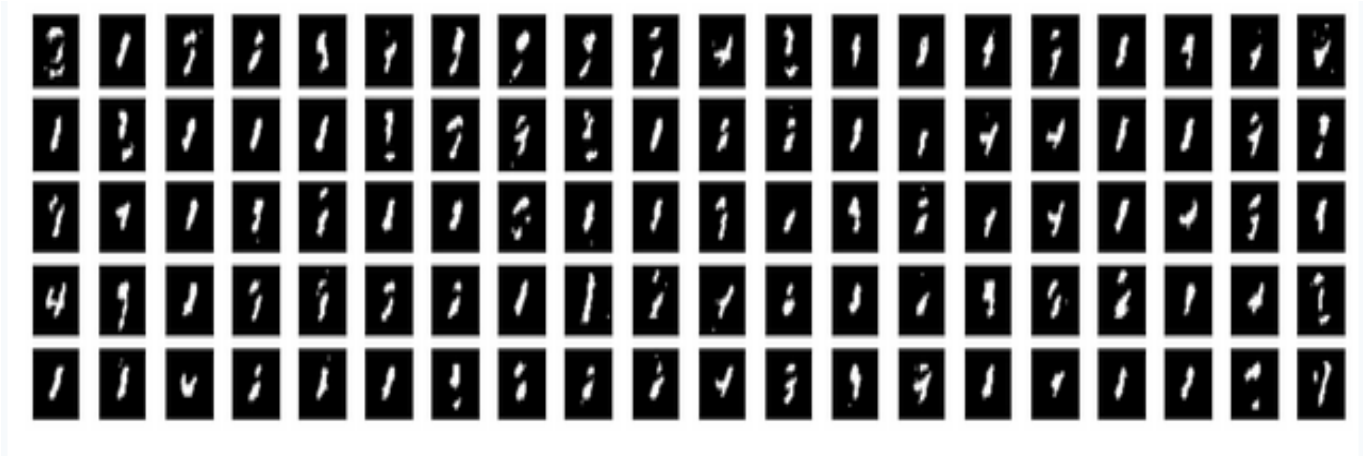

Figure 16 – Generated digits by RBM on the MNIST benchmark

Deep Belief Network

Images generated by the DBN (Figure 17) display better quality, enabling clear differentiation between the various digits.

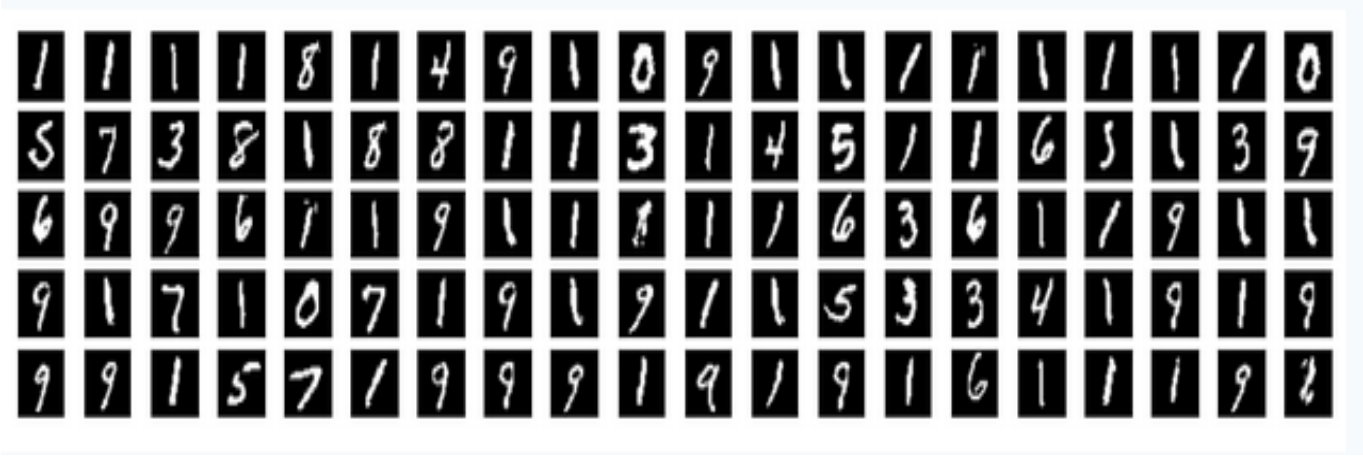

Figure 17 – Generated digits by DBN on the MNIST benchmark

Variational Auto Encoder

Images generated using the Variational Autoencoder (Figure 18) exhibit good quality overall and facilitate recognition of the different digits. However, some images may appear blurry, attributable to inherent limitations within the VAE architecture.

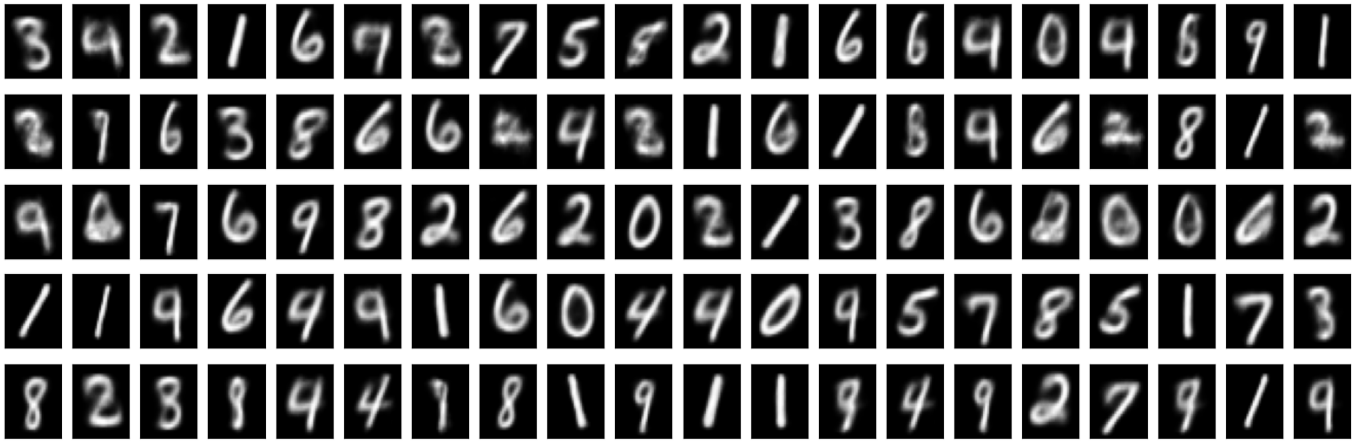

Figure 18 – Generated digits by VAE on the MNIST benchmark

Conclusion

n conclusion, this project has provided a comprehensive exploration of the capabilities and performances of deep neural networks in the task of handwritten digit classification. The study focused on comparing the performances between a pre-trained deep neural network and a randomly initialized network, both constructed on the basis of Restricted Boltzmann Machines (RBMs). The impact of various key parameters such as the volume of training data, the number of network layers, and the number of neurons per layer on the accuracy rate was also investigated.

The project utilized two main datasets: a database of small-dimension images for initial tests and the MNIST database, a standard benchmark in machine learning, to assess the effectiveness of unsupervised pre-training. The results demonstrated that unsupervised pre-training significantly improves the performance of deep neural networks, particularly when the amount of labeled training data is limited.

Furthermore, a comparative study of the generative capabilities of RBMs, DBNs, and Variational Autoencoders (VAEs) was conducted. The study revealed that while RBMs struggle to generate high-quality images, DBNs and VAEs are capable of producing more realistic and diverse samples.

In future work, it would be interesting to explore other types of deep generative models, such as Generative Adversarial Networks (GANs), and compare their performance to that of RBMs, DBNs, and VAEs. Additionally, the impact of different training strategies, such as transfer learning and multi-task learning, on the performance of deep neural networks could be investigated.

Overall, this project has provided valuable insights into the capabilities and limitations of deep neural networks in the task of handwritten digit classification and has laid the groundwork for further research in this area.

Glossary

DBN Deep Belief Networks.

DNN Deep Neural Networks.

MSE Mean squared error.

RBM Restricted Boltzmann Machine.