2506 – A study on Hyper-parameters and Energy Structure in Restricted Boltzmann Machines

Riccardo Corte, Alessandro Miotto, and Lorenzo Rizzi
(Dated: April 6, 2025)

Restricted Boltzmann Machines (RBMs) are powerful tools for learning complex data distributions. In this work, we apply RBMs to the MNIST dataset, focusing on three specific digits. Starting from a basic architecture, we optimize key hyperparameters through likelihood-based analysis and evaluate performance under various training configurations.

A central part of our study involves a deep exploration of the hidden space, the internal representation learned by the model. By applying dimensionality reduction techniques such as PCA and t-SNE, we visualize and interpret how the RBM organizes data in this latent space, identifying meaningful clusters that correspond to digit classes. We further examine the energy landscape associated with these configurations, highlighting how energy minima relate to learned patterns.

INTRODUCTION

The central focus of this dissertation is the study of Restricted Boltzmann Machines (RBMs) [2], a class of generative neural networks capable of modeling complex probability distributions. RBMs are energy-based models whose structure is inspired by physical systems in statistical mechanics, such as the Ising model. In this framework, learning corresponds to adjusting the model's parameters in order to minimize a global energy function, thereby capturing the underlying structure of the data.

The architecture of an RBM consists of two layers: a visible layer, which represents the observed data, and a hidden layer, which captures latent features. The hidden units enable the model to learn high-order correlations among the visible variables, effectively encoding complex dependencies in a simplified and more interpretable manner. Unlike traditional neural networks, RBMs are characterized by a bipartite structure, where visible and hidden units are fully connected, but no intra-layer connections exist.

Learning in RBMs involves adjusting the weights between visible and hidden units to maximize the likelihood of the training data. These weights determine the strength of interaction between neurons and are updated through a training process, typically using algorithms such as Contrastive Divergence. Given an input vector, each neuron produces a Bernoulli output that contributes to the overall configuration's energy, guiding the model toward the most probable data patterns.

The interaction between visible and hidden units in an RBM is expressed by an energy function, designed so that one can model the probability of a microstate $(\boldsymbol{v}, \boldsymbol{h})$ using the Boltzmann distribution:

$$P(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{Z} \tag{1}$$

This formulation draws inspiration from the Ising model in statistical mechanics and specifically utilizes the Hubbard–Stratonovich transformation to decouple the interacting terms, effectively transferring the complexity of the interactions into the latent space represented by the hidden layer. The energy function can then be written as:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i W_{ij} h_j \tag{2}$$

where:

- $\boldsymbol{v} \in \{0,1\}^D$ are the visible units representing the observed data,
- $\boldsymbol{h} \in \{0,1\}^L$ are the hidden units related to latent features,
- $W_{ij} \in \mathbb{R}$ are the weight parameters encoding interactions between visible and hidden units,
- $\boldsymbol{a} \in \mathbb{R}^D$ and $\boldsymbol{b} \in \mathbb{R}^L$ are respectively the biases for the visible and hidden units.

and the partition function, given by the sum over all possible states:

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{3}$$

In order to accurately model the observed data, the weights and biases must be properly learned through a training process. This is achieved by maximizing the log-likelihood of the data under the model. Training, typically involves computing the gradient of the log-likelihood with respect to the weights, which guides the parameter updates toward configurations that better represent the underlying data distribution.

$$\frac{\partial \log P(\boldsymbol{v})}{\partial W_{i\mu}} = \langle v_i h_\mu \rangle_{\text{data}} - \langle v_i h_\mu \rangle_{\text{model}} \tag{4}$$

Direct computation of the gradient of the log-likelihood is intractable due to the presence of the partition function, which requires summing over all possible configurations of the system. To make this calculation feasible, an approximation method known as Contrastive Divergence (CD) is typically used. This approach performs a truncated version of Gibbs sampling by limiting the number of sampling steps rather than running the chain until full equilibrium is reached.

Gibbs sampling is particularly well-suited to the architecture of Restricted Boltzmann Machines, which consist of two layers with no intra-layer connections. This structure enables efficient alternating updates: given the visible layer, the hidden units can be sampled independently, and vice versa.

1. Sample h_j given v_i:

$$P(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_i W_{ij} v_i\right) \tag{5}$$

2. Sample v_i given h_j:

$$P(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_j W_{ij} h_j\right) \tag{6}$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function.

In this work, we focus on identifying the optimal configuration of hyperparameters that best suits the RBM. Specifically, we will tune key values such as the learning rate and the number of Gibbs sampling steps, as these have a significant impact on training efficiency and model accuracy.

Having established the foundational concepts underlying Restricted Boltzmann Machines, we now turn to a practical application using the MNIST dataset [3]. This benchmark will allow us to explore the capabilities and limitations of RBMs in a controlled setting.

METHODS

Initialization of weights and biases

From a practical standpoint, proper initialization of weights and biases is crucial when working with Restricted Boltzmann Machines (RBMs). In this context, it is essential to acknowledge the foundational contributions of Geoffrey Hinton, whose work on RBMs and deep learning has profoundly influenced the field. Among his research, the study presented in [1] offers valuable guidelines for initializing model parameters, which can significantly impact both the efficiency and convergence of the training process.

The weights in a Restricted Boltzmann Machine are typically initialized to small random values drawn from a normal distribution with a standard deviation of approximately 0.01. While using larger initial weights can accelerate the early stages of learning, it may lead to slightly worse final results due to increased instability. Interestingly, when using stochastic learning algorithms, it is also possible to initialize all weights to zero. In such cases, the inherent noise in the stochastic process introduces sufficient variability to differentiate the hidden units, even when they begin with identical connectivity patterns.

Proper initialization of the visible unit biases is particularly critical for training efficiency. Poorly chosen initial biases can result in early training iterations being wasted on adjusting the activation probabilities of the hidden units to align with the data distribution. To address this, Hinton proposed an initialization strategy that sets the initial bias values such that the activation probabilities of the hidden units reflect the average activation of the visible units. Mathematically, the goal is to set the initial activation probability of a hidden unit as:

$$p_i = \frac{x_{mean} + x_{min}}{x_{max} + x_{min}} \tag{7}$$

Since the activation probability is related to the sigmoid function this result is obtained by initializing the bias of the i-th visible unit as follows:

$$a_i = \log\left(\frac{p_i}{1 - p_i}\right) \tag{8}$$

where p_i is the proportion of training vectors in which the unit i is active. The sigmoid function therefore becomes:

$$\sigma(a_i) = \frac{1}{1 + e^{-log\left(\frac{p_i}{1-p_i}\right)}} = p_i \tag{9}$$

As for the hidden biases, it is generally acceptable to initialize them similarly to the visible biases; however, setting them to zero is often sufficient in practice. Alternatively, initializing hidden biases with large negative values can be used as a crude method for promoting sparsity, effectively discouraging hidden unit activation unless strongly supported by the input.

The rationale behind bias initialization is to align the initial activation probabilities with the training data distribution. If all biases are set to zero, the sigmoid function yields a default activation probability of 0.5, which may not reflect the actual data. Hinton's strategy addresses this by setting biases so that initial activations match the empirical averages, enabling faster and more stable convergence during training.

Log-likelihood

As already mentioned, it is our interest to find the hyperparameters that better fits the model, this is done by minimizing the following likelihood:

$$L = \frac{1}{M} \sum_{m \leq M} \ell_\theta(\boldsymbol{v}^{(m)}) \tag{10}$$

where

$$\ell_\theta(x) = \ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} - \ln \sum_{\boldsymbol{v}'} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}',\boldsymbol{h})} \tag{11}$$

The first element of Equation 11 depends on the energy function, and is therefore calculated from the weights and biases of the model. The second element of the equation corresponds to the partition function Z (Equation 3) which depends on all possible states. Although the partition function's general formulation is known, we need to express it in a practical, computable form.

$$\ln Z = D \ln q + \ln \left(\sum_{\boldsymbol{h}} G(\boldsymbol{h}) \prod_{i}^{D} \left(\frac{1 + e^{H_i(\boldsymbol{h})}}{q} \right) \right) \tag{12}$$

where we have defined $H_i(\boldsymbol{h}) = a_i + \sum_\mu W_{i\mu} h_\mu$ and $G(\boldsymbol{h}) = \prod_\mu e^{b_\mu h_\mu}$.

All the element for this calculation are known from the training, except for the parameter q. This parameter is introduced to compensate numerical instabilities related to the products of the terms $1 + e^{H_i(h)}$. Essentially, q constitutes a scaling factor defined as $\mathbb{E}\left[1 + e^{H_i(h)}\right]$.

Numerical Implementation

The RBM computations are implemented in Python, so it's essential to briefly discuss the practical programming methods used. While NumPy can compute the log-likelihood, performance degrades significantly as the number of hidden layers increases, due to the summation over 2^L possible hidden states. More Contrastive Divergence (CD) steps further increase computation time, making hyperparameter tuning impractical.

To address this, we leveraged CPU parallelization using the Python library *Numba*. By applying a simple decorator, we enable Just-In-Time (JIT) compilation, converting frequently-used functions into machine code at runtime. Additionally, we parallelized the code with CPU multithreading, reducing computation time from 16 minutes to under 30 seconds—achieving a 32.3x performance improvement.

RESULTS

In this section, we present the results, starting with the general form of the log-likelihood and then exploring the outcomes for various hyperparameter configurations.

Log-likelihood

As shown in Figure 1, the negative log-likelihood increases during training, which is expected. Conceptually, the visible and hidden units represent a system of interacting particles, with their state defined by the energy function. Training the RBM involves adjusting the parameters so that the real data corresponds to states of minimal free energy—i.e., high-probability states. From a physical perspective, maximizing the log-likelihood is equivalent to minimizing the system's free energy.

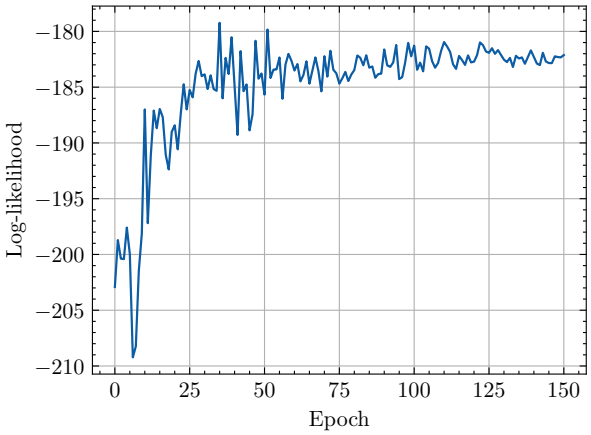

Figure 1: Log-likelihood for $L = 3$ using two CD steps. Stochastic Gradient Descent is employed as optimizer, with an adaptive learning rate from 1.0 to 0.25.

Hyperparameters

As mentioned earlier, selecting the right hyperparameters is essential for optimizing the performance of a Restricted Boltzmann Machine (RBM). Understanding the impact of each hyperparameter is key to identifying the optimal configuration. The main hyperparameters are presented in Table I.

Varying L

To understand how the number of hidden units impacts the log-likelihood, it's important to first contextualize their role. Hidden units serve as a latent representation of the data, capturing dependencies and patterns in the visible units. For example, a hidden unit may be shaped to represent a classification of the digits 0,1, or 2.

The expected impact of varying the number of hidden units is as follows: with too few hidden units, the model

Hyperparameter	Description
Learning rate (η)	Controls the step size of weight updates during training. A small value leads to slow convergence, while a large value may cause instability.
Hidden units (L)	Determines the capacity of the RBM to capture complex representations. More hidden units allow for higher expressivity but increase the risk of overfitting.
Contrastive divergence steps (CD)	Specifies the number of Gibbs sampling steps used in Contrastive Divergence. A higher value improves convergence to thermodynamic equilibrium but increases computational cost.
Lasso L1 regularization parameter (γ)	A regularization term that penalizes large weights to prevent overfitting and encourage sparsity.

Table I: RBM Hyperparameters

lacks the capacity to capture the underlying data structure, resulting in limited representational power and a lower log-likelihood. On the other hand, a very large number of hidden units may lead to overfitting or cause the model to reach a performance plateau.

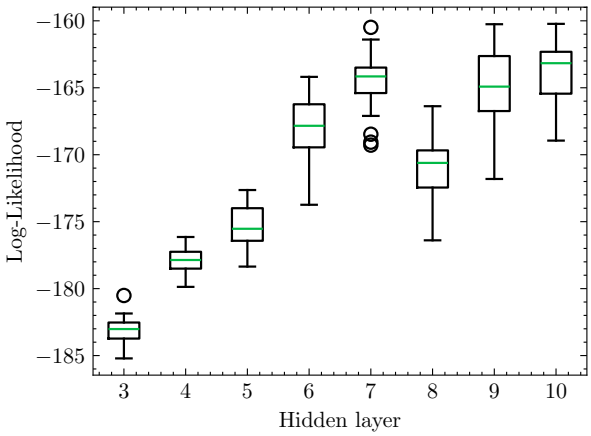

Figure 2: Average value of the last 40 epochs out of 150 for various hidden layer dimensions L, using RMSProp with a constant learning rate of 0.05.

This analysis aligns with the results showed in Figure 2, where the optimal configuration is achieved with $L = 6$. Further increases in the number of hidden units do not significantly improve the results.

Varying CD

We now analyze how the performance of the RBM changes with different numbers of Contrastive Divergence (CD) steps, keeping all other hyperparameters fixed.

The CD step count determines how many Gibbs sampling iterations are used to approximate the model's distribution during training. Since RBMs are trained via gradient descent, even a rough estimate of the gradient is often enough to guide the system toward an energy minimum. For this reason, a small number of CD steps, such as the two originally proposed by Hinton, is typically sufficient.

As shown in Figure 3, the results confirm this expectation: even with only a few Gibbs steps, the RBM achieves good performance. Just two CD steps are enough to steer the optimization along an effective path toward convergence.

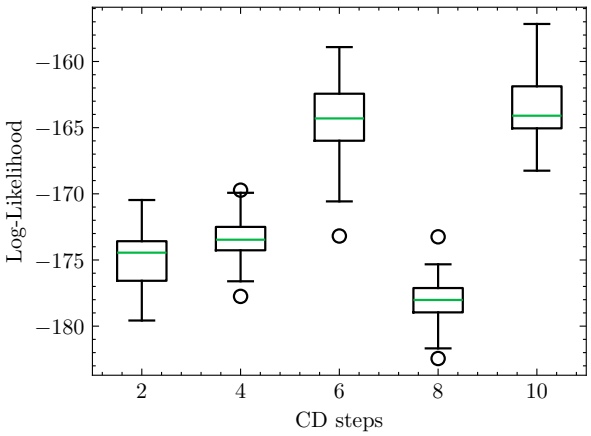

Figure 3: Average value of the last 40 epochs out of 150 for various number of CD steps. The number of hidden units is fixed at $L = 6$ as well as RMSprop optimizer with constant learning rate of 0.05.

Varying η

The learning rate determines the size of the steps the model takes during gradient descent. Selecting an inappropriate value can cause the model to diverge or converge too slowly, so careful tuning is essential. In our experiments, we employ adaptive learning rates, dynamically adjusting the learning rate during training to improve convergence and overall performance.

We compare two optimization methods: Stochastic Gradient Descent (SGD) and Root Mean Square Propagation (RMSprop).

SGD updates model parameters using gradients computed from randomly selected batches of training data. This stochasticity can help escape local minima but may also lead to oscillations and slower convergence.

RMSprop, in contrast, adapts the learning rate for each parameter using a moving average of past squared gradients. This helps stabilize training and often results in faster convergence.

Despite their methodological differences, our results show that both optimizers perform similarly overall, with strong sensitivity to the learning rate schedule. As shown in Figure 4, RMSprop yields better results with smaller learning rates, while SGD benefits from an intermediate value that balances stability and speed.

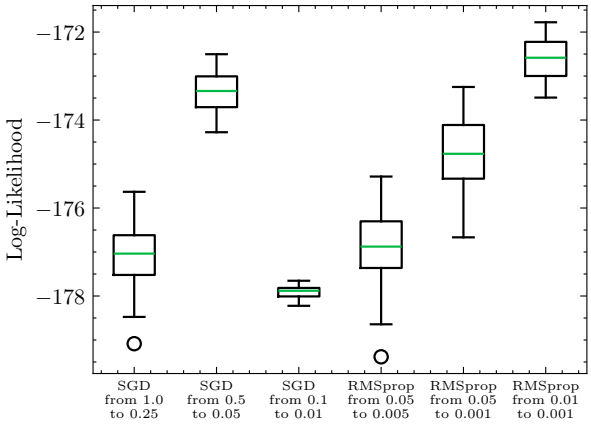

Figure 4: Average value of the last 40 epochs out of 150 for various learning rate schedules and optimizers. The number of hidden units is fixed at $L = 6$ and the number of CD steps at two.

Varying (γ)

We extend our analysis by introducing LASSO (L1) regularization, which promotes sparsity in the model by penalizing large weights. This helps prevent overfitting and encourages the emergence of simpler, more interpretable structures.

In the case of the MNIST dataset, which is already sparse, applying regularization does not necessarily improve performance. However, it plays a crucial role in producing cleaner and more interpretable weights, which can be valuable for understanding the underlying data structure.

As shown in Figure 5, excessive regularization can degrade performance by overly constraining the model and limiting its ability to learn useful patterns. In contrast, smaller values of γ strike a better balance, preserving model expressiveness while promoting sparsity and stability.

Potts Model

Although not strictly a hyperparameter, it is relevant to discuss the implementation and results of the Potts model. This model generalizes the Ising model by

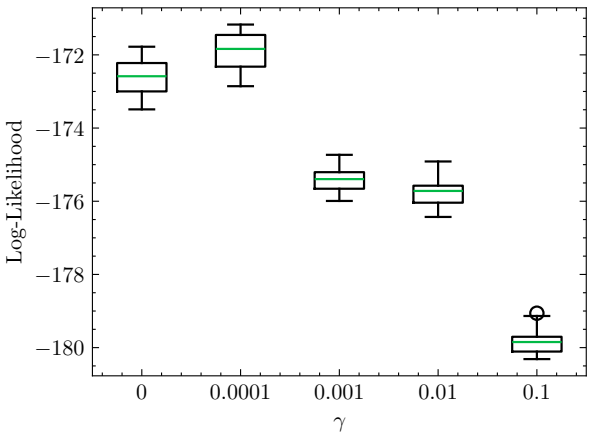

Figure 5: Average value of the last 40 epochs out of 150 for various L1 regularization parameters γ. $L = 6$ hidden units and two CD steps. The optimizer is RMSprop with $0.01 \to 0.001$ learning rate schedule.

allowing each hidden unit to take on q discrete states, represented as categorical variables using one-hot encoding. Unlike binary hidden units, these categorical units offer a different form of representation, with the hidden layer spanning q^L possible configurations.

As shown in Figure 6, the Potts model yields suboptimal log-likelihood results on MNIST compared to standard RBMs. The added representational flexibility of the Potts model does not translate into better performance in this case and may even introduce unnecessary overhead.

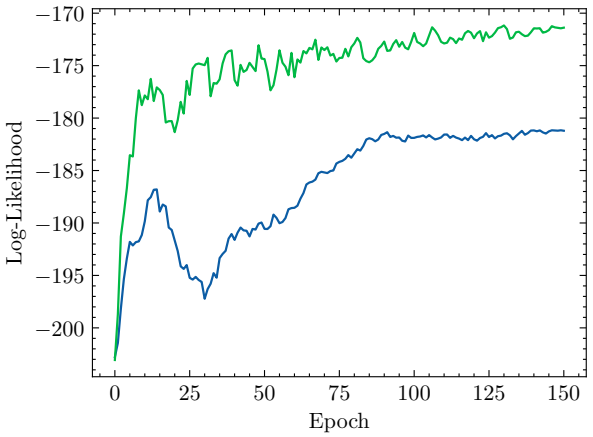

Figure 6: Comparison of standard binary RBMs (green) and Potts models with one-hot encoding (blue). Hidden units fixed at $L = 6$, using 2 CD steps and RMSprop with a $0.01 \to 0.001$ learning rate schedule.

RESTRICTED BOLTZMANN MACHINES TOMOGRAPHY

In this section, we aim to gain a deeper understanding of the training process of a restricted Boltzmann machine. By exploiting the RBM's dimension reduction capabilities, we map the complex 784-dimensional space of handwritten digits from the MNIST dataset into a lower-dimensional hidden space.

In the subsequent sections, we extend this analysis by applying additional dimensionality reduction techniques, such as Principal Component Analysis (PCA) or t-SNE, to project the learned hidden representations into even lower-dimensional spaces. This process effectively acts as a form of high-dimensional tomography, where we progressively reduce the number of dimensions to visualize and interpret the underlying structure of the learned features.

Principal component analysis

In order to gain insights into the structure of the hidden layer and better understand how it encodes the data, we employ dimension reduction techniques. These methods allow us to project the high-dimensional hidden unit (a L-dimensional hypercube) into lower-dimensional spaces, making it easier to visualize and interpret how the model learns to represent complex patterns, such as specific digits in the MNIST dataset.

Our first approach to understanding the structure of the hidden layer is using Principal Component Analysis (PCA), which reduces the dimensionality of the data by projecting it onto the directions of maximum variance. The second approach is t-Distributed Stochastic Neighbor Embedding (t-SNE), a technique that visualizes high-dimensional data by modeling pairwise similarities in a lower-dimensional space. However, t-SNE's stochastic nature makes it impractical for directly returning to the original space.

PCA relies on the assumption that the data follows a normal distribution in the hidden units h_i, as it explains variability based on variance. Since our variables are binary, standard PCA may lead to poor performance. To address this, we used a surrogate for the covariance matrix that better captures variability in the dataset. Specifically, we employed the Hamming distance, which measures the number of differing bits between binary vectors, as a way to quantify the dissimilarity between hidden unit activations.

Figure 7 shows the performance of different dimensionality reduction techniques. We start with an RBM with 24 hidden units to avoid point concentration in the same coordinates, which can occur with fewer units due to the limited number of configurations in the 2^L space. Both PCA and t-SNE perform well despite the non-normality of the dataset. While t-SNE better separates the digits, its inability to easily map back to the original 24D space makes standard PCA our preferred choice. Hamming PCA, however, yields inconclusive results as the algorithm struggles to separate the digits properly.

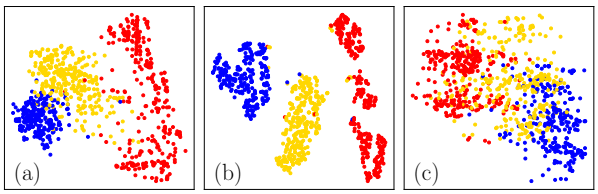

Figure 7: First two principal components using standard PCA (a), t-SNE (b), and Hamming PCA (c). The blue, red, and yellow points represent digits 0, 1, and 2, respectively.

Figure 8 displays the first three principal components of the PCA, along with the energy calculated for each point in the 3D space, which is then projected back into the original 24D hidden space. Using Gibbs sampling, we obtain the visible layer in 784D and compute the corresponding energy. The figure shows 1,000 data points projected in 3D, with their respective energy values. A deeper analysis on the energy landscape is reported in the following sections.

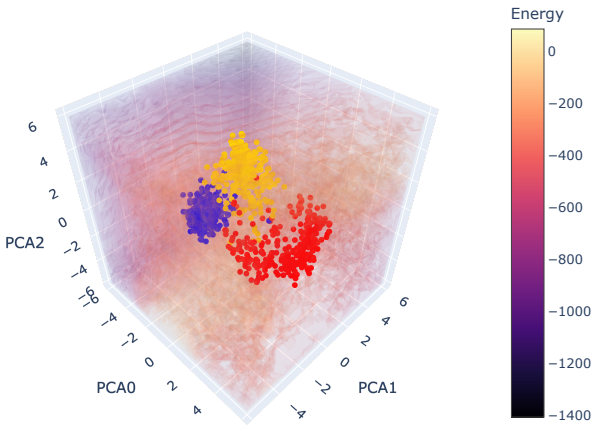

Figure 8: First three principal components using standard PCA. The blue, red, and yellow points represent digits 0, 1, and 2, respectively. The energy is calculated averaging 10 separated Gibbs sampling with 100 CD steps each.

Restricted Boltzmann Machines as classificators

Building on the satisfactory results of the previous sections, we can now attempt to use the trained RBM as a digit classifier, so a type of *supervised learning* technique, since the MNIST dataset comes equipped with labels. In fact, starting from all the digits in an initial set (*training set*), we were able to identify three different clusters in the reduced 3D hidden space, each one of them corresponding to a different digit. Now, given a previously unseen digit, we perform multiple rounds of Gibbs sampling using the machine already trained to obtain a hidden representation in 24D (which should converge towards a local minimum). This representation is then mapped to a 3D vector using the PCA matrix that was produced according to the training set. Finally, we classify the digit based on its closest cluster by selecting its first k neighbors (in the training set) and determining the digit to which they belong.

We used a subset of the MNIST dataset as the training set, specifically the first 1000 elements. The RBM was trained and the PCA transformation matrix was constructed using these points. To evaluate its classification performance, we selected 400 additional points from the same MNIST dataset for validation. Since the dataset includes associated labels, we can assess the effectiveness of the model as a classifier by measuring its validation accuracy. The value of k was set to be equal to 10. We found that, on average:

$$V.A. = (90.3 \pm 1.0)\% \tag{13}$$

We can observe that:

- **Performance**: The RBM performs reasonably well on this task; however, its performance is generally lower when compared to more sophisticated methods, such as Convolutional Neural Networks (CNNs), where accuracies can exceed those achieved by the RBM
- **Stochasticity**: The uncertainty introduced in Equation 13 reflects the inherent stochastic nature of the RBM classifier's final prediction. The Gibbs sampling procedure is a stochastic algorithm, meaning that different runs on the same validation digit may yield different results. Moreover, the training of the RBM itself, conducted via the contrastive divergence method, is also stochastic, making the final prediction necessarily unstable and inconsistent, even when hyperparameters such as k remain fixed. To balance this effect, we repeated the classification 10 times and computed the averaged performance
- **Time**: The Gibbs alternating sampling process can be time-consuming, negatively impacting the model's efficiency. As a result, the RBM classifier takes significantly longer to classify the test set compared to an ordinary neural network.

We can thus conclude that the RBM-classifier performs reasonably well, achieving around 90% accuracy. However, its main drawbacks are its stochastic nature and the relatively slow execution time. Consequently, more traditional methods, such as CNNs are preferable due to their superior performance and efficiency in handling image classification tasks.

Energy Analysis

As previously mentioned, we now examine the energy landscape resulting from the RBM's training. Ideally, we expect energy minima to correspond to learned digit representations. However, as shown in the 2D PCA projection in Figure 9, the energy distribution deviates from this expectation: low-energy regions appear at the edges, and an unexpected energy maximum occurs beneath the cluster of 2s.

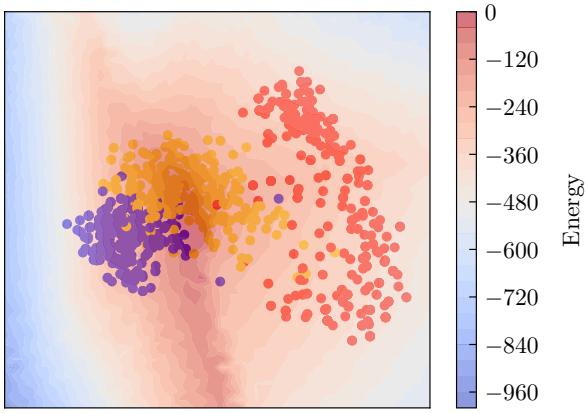

Figure 9: First two principal components using standard PCA. The blue, red, and yellow points represent digits 0, 1, and 2, respectively. The energy is calculated averaging 10 separated Gibbs sampling with 100 CD steps each.

This discrepancy arises from the sampling procedure. Energy values are computed by projecting 2D PCA coordinates back into the original 24D hidden space. However, since the hidden units are binary ($h_i \in \{0, 1\}$), only points within the 24D unit hypercube represent valid configurations. Many of the projected points fall outside this admissible space, resulting in unphysical, unreachable states with artificially low energy.

Additionally, PCA distorts scale to highlight clustering, which further limits interpretability in terms of energy. For a more accurate analysis, we therefore turn to examining the energy landscape directly in the original 24-dimensional hidden space in the following section.

Hamming path

We begin by randomly selecting two digits from the MNIST dataset, represented in the 24-dimensional hidden space as $\mathbf{h}_1$ and $\mathbf{h}_2$. Since the components $h_{1,i}$ and $h_{2,i}$ are binary-valued (either 0 or 1), these vectors can be interpreted as vertices of a unit hypercube in 24 dimensions. We now construct a discrete path between $\mathbf{h}_1$ and $\mathbf{h}_2$ that moves through adjacent vertices of the 24-dimensional hypercube (i.e. a *Hamming path*, since every element of the path is different from the previous one by just a bit). Finally, for each intermediate hidden vector, we compute the energy with Equation 2. Results for the energy behaviour along a Hamming path connecting a 0 to a 2 are shown in Figure 10. At each step of the path, we also represent the visible counterpart of the hidden vector (obtained with a round of Gibbs sampling).

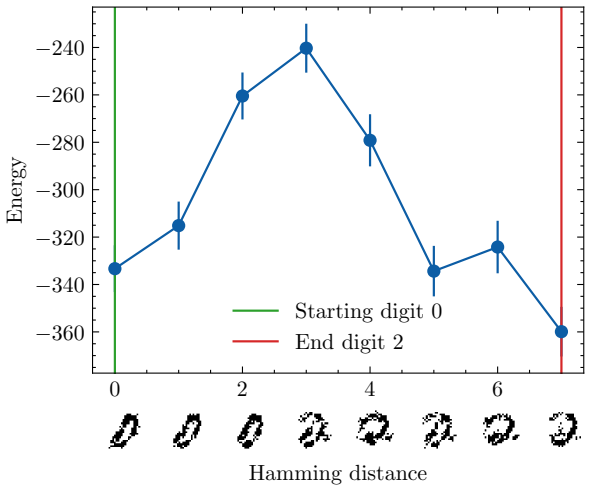

Figure 10: Value of the energy function E along the vertices of the 24D hypercube connecting two digits, in this case from 0 to 2.

The resulting energy profile exhibits a very interesting pattern: it begins at what appears to be a local minimum and increases as we move away from the initial digit, reaching a peak at some point along the Hamming path. The corresponding visible representations begin with a clear image of the selected starting digit and gradually morph into the target digit (2). Indeed, the energy peak aligns with a highly distorted and ambiguous version of digit 2. Beyond this point, the energy decreases again, approaching a new minimum where the final digit is fully formed. This behavior is exactly the one we expected for the RBM: the model has learned the data distribution of the input dataset and has assigned higher probabilities to meaningful and recurring samples by assigning them lower energy values. Indeed, the highly distorted image of the digit 2 was associated with a relatively high energy value, meaning that this specific visible pattern has a low chance of being generated. In contrast, the hidden vectors corresponding to the three digits were assigned lower energy values, indicating that they are much more likely to be generated (and, in that sense, we say that the machine has learned something).

Euclidean path

We now attempt to repeat the same procedure as the one illustrated in the previous paragraph, but this time choosing an euclidean path connecting the initial and final chosen digits. This means that we no longer jump from one vertex to another within the 24-dimensional hypercube (representing the possible values of $\boldsymbol{h}$) and compute the energy along the way. Instead, we embed the hypercube into a 24-dimensional Euclidean metric space and define the energy function even for points inside the hypercube, where it is no longer strictly true that $h_i \in \{0, 1\}$. However, this relaxation does not pose any issues for the algebraic computation of $E(\boldsymbol{h}, \boldsymbol{v})$.

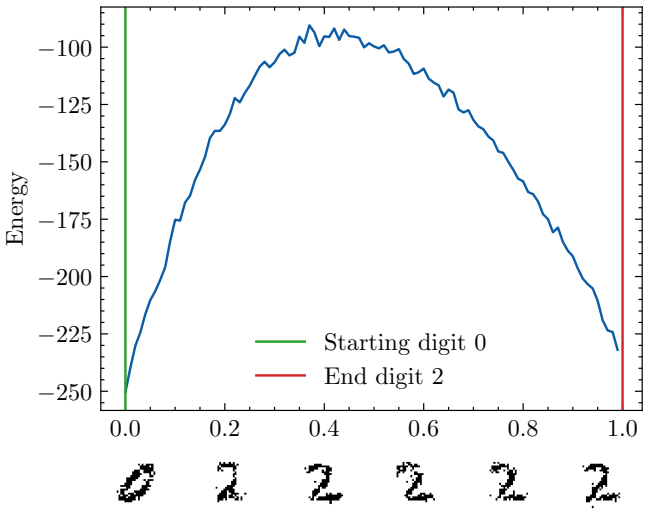

Figure 11: Value of the energy function E along an euclidean line in the 24D hidden space connecting two digits, in this case from 0 to 2.

As before, we select a pair of points and connect them with a straight line in the Euclidean sense. By parameterizing this line with $0 \leq t \leq 1$, we can compute the energy along this path. This allows us to investigate the analytically continued energy function, even for those values of $\boldsymbol{h}$ that are not in a binarized form. Results for a linear path in 24D between a specific pair of points are shown in Figure 11. The energy value increases as we move away from digit 0, reaching a maximum somewhere along the path, after which it decreases as we approach digit 2. This behavior aligns perfectly with our understanding of how a RBM

operates and with the results of the discrete Hamming path (Figure 10). However, by embedding the hypercube into an Euclidean space and considering a continuous version of the problem, we were able to obtain a much smoother curve, which will be particularly useful for later applications. In this continuous representation, it becomes evident that the local minima learned by the machine, corresponding to the digits, are distinctly separated by an energy barrier in the hidden space.

It is now very interesting to compare the result just obtained performing a similar operation in the reduced 3D PCA hidden space. This direct confront will allow us to study how the model performs over different dimensionalities since it is not trivial to obtain the same results. First, we project our sample points into their 3D representations using the PCA transformation. We then connect these projected points with a straight line, which now lies within a three-dimensional space. As in the 24D case, we compute and visualize the energy along this path to analyze the behavior of the energy landscape in the reduced space. This is done by reprojecting the 3D points coordinates over the 24 dimensions using the inverse PCA, calculating the corresponding energy values. The key concept is that the corresponding 24D points over which we perform the energy calculation are not connected in a straight line, hence we could obtain unexpected results. The 3D visualization of the path is reported in Figure 12, while the values of the energies over the line are reported in Figure 13.

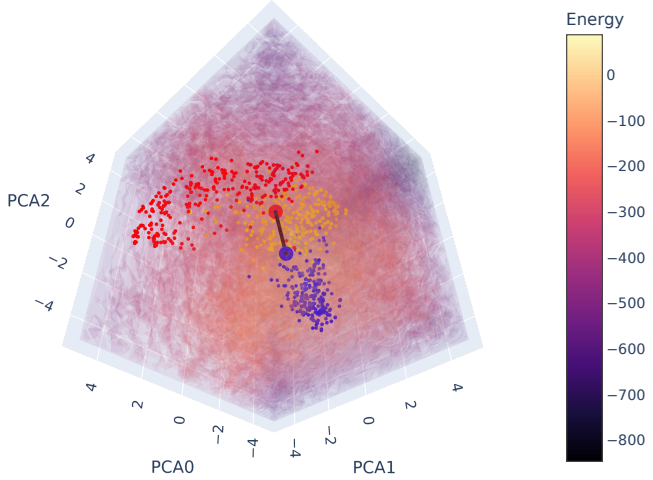

Figure 12: 3D visualization of the Hidden space. The black line indicates the path from a digit 0 to a digit 2 of 3d coordinates over which the Energy have been explicitly calculated.

We can see that the value of the energy starting from a 3D sampling of the hidden units returns results which are basically identical to the 24D case. The consistency between the two measures confirms our expectations allowing for a different approach to analyze the system.

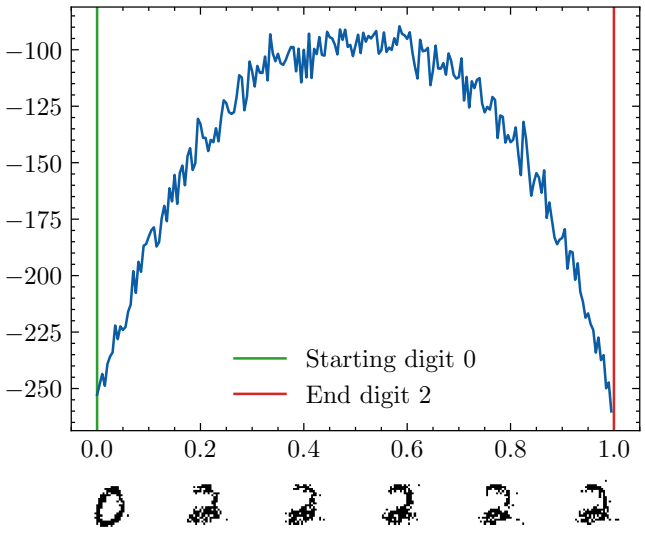

Figure 13: Value of the energy function E along an euclidean line in the 3D hidden space connecting two digits, in this case from 0 to 2.

We observe that the energy values obtained from a 3D sampling of the hidden units are very similar to those computed in the full 24-dimensional space. This consistency between the two approaches confirms our expectations and validates the use of a reduced-dimensional representation as a viable alternative for analyzing the system.

Band gap

In the previous paragraphs, we illustrated the energy trend of the model along a path (either continuous or discrete) connecting two digits in the dataset. As expected, this energy profile exhibits a maximum that energetically separates the two local minima corresponding to the starting and ending points. This occurs because the RBM aims to minimize the energy of the points corresponding to the samples seen during training.

Now, we repeat the same process described in the previous subsection, this time using a large number of sample pairs from the dataset. The obtained results are shown in Figure 14 and confirm the trend observed with a single pair of points in Figure 11. Notably, we observe that the "valley" of 1s is associated with a lower energy value compared to 0s and 2s, indicating that the RBM has learned those digits a bit more effectively. Additional insights into the energy landscape in the hidden space can be inferred from the same plot. For instance, the bandwidth at $t = 0$ or $t = 1$ (corresponding to the sample points) is related to how uniform the energy is

within a single cluster (either that of the starting digit or the ending digit). By visually inspecting the plot, we observe that the bandwidth associated with digit 1 is smaller than those associated with digits 0 and 2, meaning that the 1s in the dataset were probably more similar to each other than the 0s or 2s.

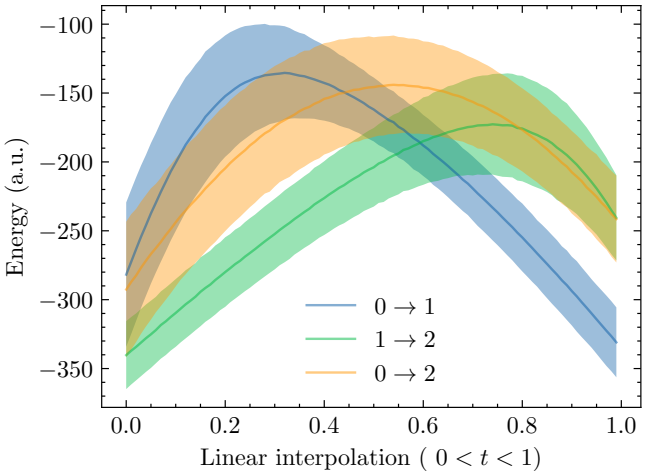

Figure 14: Representation of multiple energy paths along a 24D line connecting different digit pairs. The continuous line represents the mean energy value at a specific value of the parameter t. The bandwidth corresponds to the standard deviation of the energy values.

Given a pair of samples belonging to two different digits, we define the *energy gap* as the amount of energy required to transition from the minimum of the starting digit to that of the ending digit (that is, the maximum energy value along the path minus the initial energy value). This quantity provides insights into how easily two digits can be confused: if the average energy gap between two digits is low, the sampling algorithm is more likely to fall into the incorrect minimum and generate an unintended image. Actually, the value of $\Delta E_{a\to b}$ only tells us how easy it is for the Boltzmann machine to confuse the digit a with the digit b. In fact, looking at Figure 11, we see that the energy path are not always symmetric, so we expect that, generally, $\Delta E_{a\to b} \neq \Delta E_{b\to a}$.

To compute those energy gaps, as we did with Figure 14, we fix the starting and ending digits and we compute ΔE for a large number of pairs in the dataset. The energy gaps distribution is shown in Figure 15, and their average value (in arbitrary units) is:

$$\Delta E_{0\to1} = 153.3 \pm 0.4 \qquad \Delta E_{1\to0} = 205.6 \pm 0.5$$
$$\Delta E_{0\to2} = 147.0 \pm 0.4 \qquad \Delta E_{2\to0} = 100.4 \pm 0.5$$
$$\Delta E_{1\to2} = 170.9 \pm 0.5 \qquad \Delta E_{2\to1} = 71.0 \pm 0.3$$

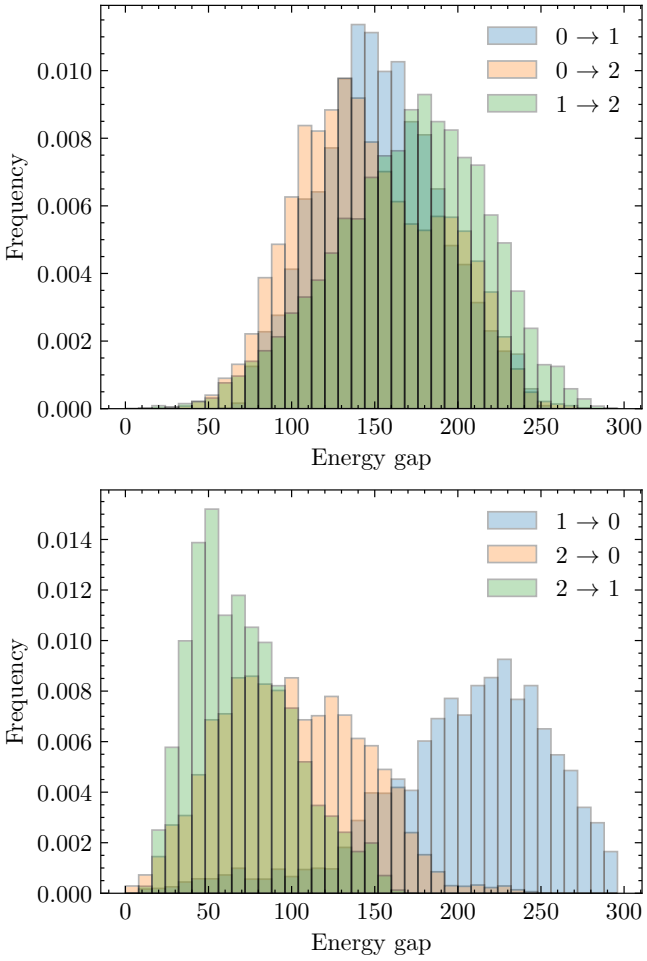

Figure 15: Distribution of the energy gaps between digits for three different configuration of starting-ending points.

The smallest value of ΔE occurs when the transition is made from digit 2 to digit 1 (in agreement with Figure 14). This can be interpreted as indicating that the model has a higher probability of misclassifying a 2 as a 1 compared to misclassifications involving other digit pairs. Notably, since $\Delta E_{1\to2} > \Delta E_{2\to1}$, it is more likely that the model mistakes a 2 for a 1 than a 1 for a 2. The value of $\Delta E_{a\to b}$ may therefore serve as a metric for assessing the confusion properties of the RBM.

CONCLUSIONS

The submitted work proved the capabilities of Restricted Boltzmann Machines (RBMs) by exploiting their full potential application while also extending beyond conventional implementations. Through a detailed analysis of various configurations and a complete log-likelihood study across a range of hyperparameter settings, it provided a deeper understanding of the model's behavior and its limitations. The work also aimed to replicate and enhance the RBM model following the methodologies and improvements introduced by the Nobel prize winner, Geoffrey Hinton.

Beyond reproducing known results, we ventured into original paths. Inspired by clustering techniques, we implemented a 3D visualization of the hidden unit activations, which revealed meaningful structural properties of the latent space. The energy landscape analysis proved insightful; the bi-dimensional and three-dimensional visualization helped understand how the local minima of the energy function behave. These minima correspond to the actual locations of the learned hidden representations, aligning well with our theoretical expectations. The clusters identified in the low-dimensional space were also used to construct a classifier based on the RBM, with mediocre results.

Finally, the analysis of the energy bands and gaps through euclidean embedding offered additional insights into the internal mechanisms of the RBM, shedding light on the model's energetic structure and providing a quantitative framework for defining a confusion metric.

In summary, this project provided an in-depth exploration of the inner functioning of a Restricted Boltzmann Machine. Moreover, it allowed to explore their capabilities through both theoretical analysis and innovative experimentation. The project also opened prospects for further investigation: for example, the development of a PCA variant more suitable for binary model is an interesting topic to explore in the optic of future improvements.

[1] G. E. Hinton. *A Practical Guide to Training Restricted Boltzmann Machines*. Springer, 2012.

[2] P. Mehta, M. Bukov, C. Wang, A. Day, C. Richardson, C. K. Fisher, D. J. Schwab, *A high-bias, low-variance introduction to Machine Learning for physicists*. Physics Reports, Volume 810, 2019.

[3] E. Alpaydin and C. Kaynak. *Optical Recognition of Handwritten Digits* UCI Machine Learning Repository, 1998.