Unsupervised Learning and Log-likelihood Analysis with Restricted Boltzmann Machines

Elias Maria Bonasera, Alberto Casellato, Nicola Garbin, and Francesco Pazzocco
(-1Dated: April 1, 2025)

We present a comprehensive study on the use of restricted Boltzmann machines (RBMs) for unsupervised learning of handwritten digit features from the MNIST dataset. Through experiments with a custom Python implementation, we focus on the RBM architecture, the training via Contrastive Divergence, and extensive hyperparameter tuning. We detail the experimental setup, the log-likelihood estimation method, and evaluate different configurations. The results highlight both the strengths and limitations of the approach, suggesting directions for future improvements.

INTRODUCTION

Unsupervised learning is essential for extracting features from unlabeled data, and probabilistic graphical models (PGMs), such as Markov Random Fields (MRFs), provide a framework for this task. Restricted Boltzmann Machines (RBMs) are a specific type of MRF with a bipartite structure (a visible and a hidden layer with no intra-layer connections). Their ability to model complex distributions and learn efficient representations of intricate data relatively fast makes them interesting both theoretically and for practical applications [1–3].

In this work, we present a complete implementation of a Restricted Boltzmann Machine (RBM) and analyze its behavior through systematic experiments. The paper is organized as follows: Section 1 presents the theoretical foundation of RBMs, Section 2 details the experimental methodology and implementation specifics, Section 3 shows the experimental results accompanied by graphical analysis, and Section 4 discusses our findings, draws conclusions, and suggests avenues for future research. Supplementary equations and technical details are provided in the Appendix.

1. THEORY

Restricted Boltzmann Machines RBMs are generative models consisting of a visible layer, representing the input data, and a hidden layer that captures latent features.

Mathematically, RMBs consist in a set of D binary visible units i of state $\{v_i\}_{i=1,\dots,D}$ symmetrically connected to a set of L binary hidden units μ of state $\{h_\mu\}_{\mu=1,\dots,L}$. The continuos weight $w_{i\mu}$ quantifies the strength between unit i and unit μ (see Figure 1). [1, 2].

The energy of a configuration $(\mathbf{v},\mathbf{h})$ is defined as

$$E(\mathbf{v},\mathbf{h}) = -\sum_i a_i v_i - \sum_\mu b_\mu h_\mu - \sum_{i,\mu} v_i w_{i\mu} h_\mu \tag{1}$$

where a_i and b_μ are the biases for the visible and hidden units, respectively, and $w_{i\mu}$ denotes the weight between them. No intra-layer connections exist, which simplifies the energy function and learning dynamics.

The probability of a hidden unit being active is computed

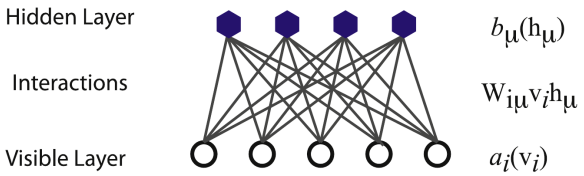

FIG. 1. Bipartite structure of a Restricted Boltzmann Machine. Visible units (bottom) are connected to hidden units (top) with no intra-layer connections.

using the logistic sigmoid for both visibile and hidden units:

$$p(h_\mu = 1|\mathbf{v}) = \sigma\left(b_\mu + \sum_i v_i w_{i\mu}\right) \tag{2}$$

$$p(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_\mu h_\mu w_{i\mu}\right) \tag{3}$$

The training consists in the minimization of the energy's $\boldsymbol{\theta}$ parameters; and is typically performed via Contrastive Divergence algorithm, which approximates the gradient of the log-likelihood by alternating Gibbs sampling between the visible and hidden layers [1–3].

2. METHODS

Experimental Setup and Preprocessing The MNIST dataset is used, consisting of 60,000 training images and 10,000 test images of handwritten digits. Each image is flattened into a 784-dimensional vector and binarized by thresholding at 0.5. Two encoding schemes are implemented:

- **Bernoulli encoding:** Pixel values in $\{0,1\}$.
- **Spin encoding:** Pixel values in $\{-1,+1\}$.

The RBM is implemented in a Python class (`RBM`) with configurable hyperparameters.

Hyperparameter Tuning Key hyperparameters were systematically varied to assess their effect on model performance [1–3]. These include:

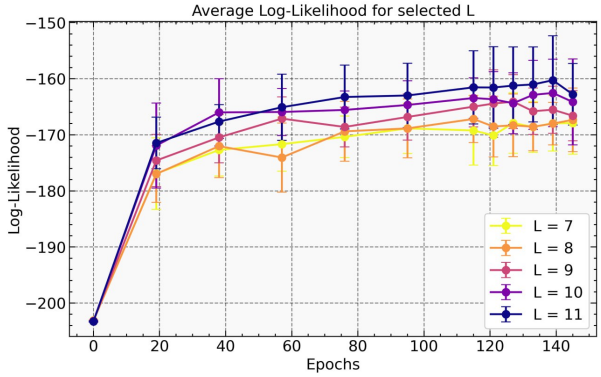

FIG. 2. Evolution of the log-likelihood over training epochs. Results are averaged over multiple runs.

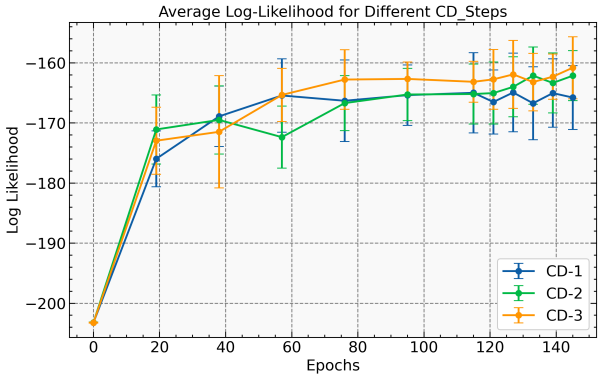

FIG. 3. Log-likelihood evolution for different numbers of Contrastive Divergence steps.

- *Number of hidden units (e.g., 3 to 13):* Determines the model's capacity. Too few may lead to underfitting, while too many could cause overfitting.

- *Contrastive Divergence (CD) steps (k=1,2,3):* Affects the accuracy of gradient estimation; lower k may introduce bias, whereas higher k could improve approximation but slows training.

- *Learning rate and regularization coefficients:* Control the speed and stability of convergence; improper values may result in divergence or overly slow learning.

- *Optimizer type (SGD versus RMSprop):* Influences how weights are updated and can impact convergence speed and the ability to escape local minima.

- *Data encoding (Bernoulli vs. Spin):* Alters the energy function and conditional probabilities, affecting how well the model captures data characteristics.

For each configuration, training is repeated over multiple runs (each executed 10 times) to ensure statistically robust results.

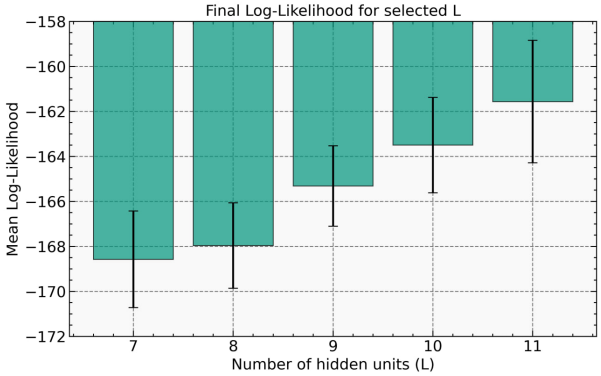

FIG. 4. Bar chart displaying the weighted mean of the log-likelihood over the last epochs for different values of L.

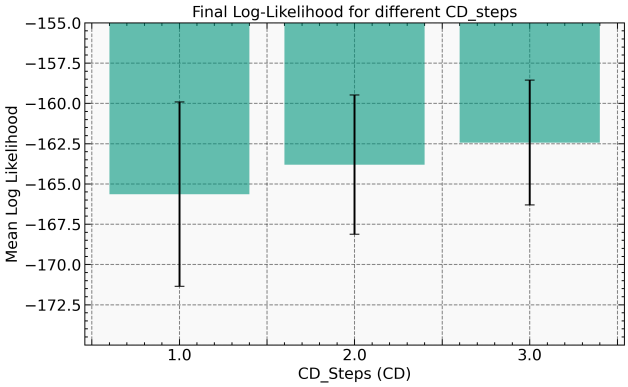

FIG. 5. Bar chart displaying the weighted mean and standard deviation of the log-likelihood over the last epochs for different numbers of CD steps.

Log-Likelihood Estimation The performance of the RBM is evaluated using the log-likelihood metric:

$$\mathcal{L} = \frac{1}{M}\sum_{m=1}^{M} \log\left(\sum_{\mathbf{h}} e^{-E(\mathbf{v}^{(m)},\mathbf{h})}\right) - \log Z \tag{4}$$

where Z is the partition function. For details control Section 5).

This metric serves both to monitor the training progress and to ensure the consistency of the model's performance across different hyperparameter settings [1, 2].

3. RESULTS

Experimental results are presented through several graphs. IN our analysis we observed the following trends:

- **Steady improvement of log-likelihood:** As depicted in Figure 2, the log-likelihood steadily increases during training and saturates after a sufficient number of epochs around epoch=100, confirming that the RBM is effectively learning to model the data [2, 3].

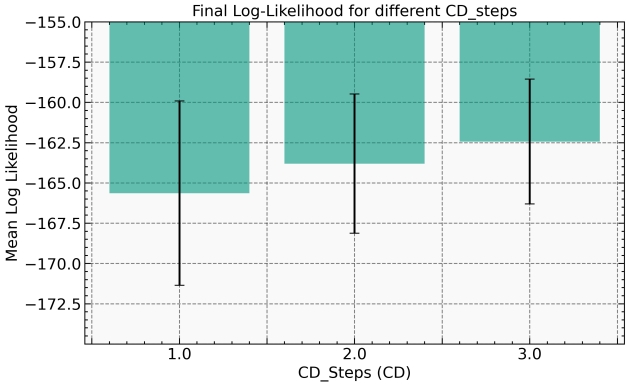

FIG. 6. Bar chart displaying the weighted mean and standard deviation of the log-likelihood over the last epochs for different numbers of CD steps.

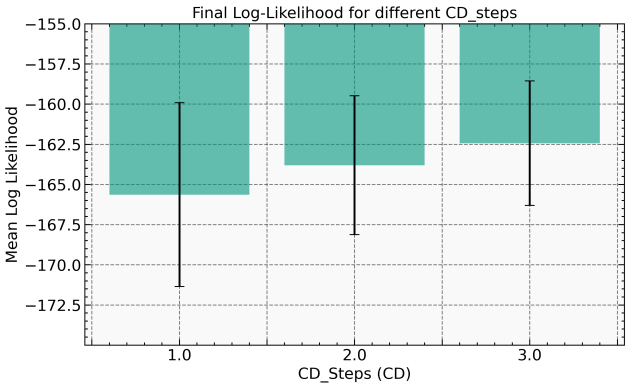

FIG. 7. Bar chart displaying the weighted mean and standard deviation of the log-likelihood over the last epochs for different numbers of CD steps.

- **Effects of CD steps:** Figure 3 indicates that increasing the number of CD steps from 1 to 3 yields only marginal gains. Moreover, the measures shows, via the analysis of the t-test after the 100 epochs, great compatibility; This suggests that CD-1 provides a good balance between computational efficiency and gradient approximation quality; however, too few steps may introduce bias and risk divergence [2, 3]. For this last reason in the following tuning we kept CD-3.

CD-1 vs CD-2	CD-2 vs CD-3	CD-3 vs CD-1
$t=0.256$	$t=0.236$	$t=0.464$

TABLE I. t-values for each pair of chosen CD steps.

- **Impact of hidden unit count:** As shown in Figure 4, increasing the number of hidden units improves performance up to a point, after which the benefits plateau. This reflects the model's ability to capture data complexity without incurring overfitting when the capacity is appropriately chosen [2, 3].

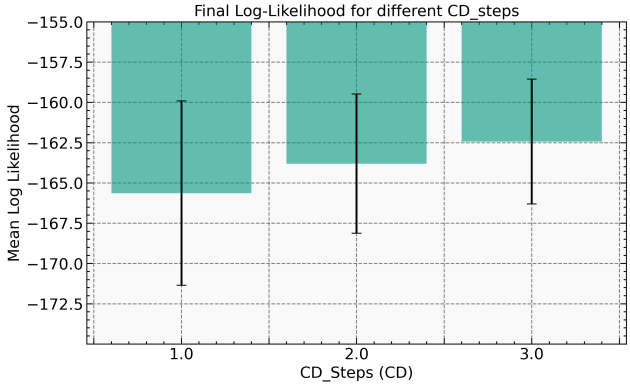

FIG. 8. Bar chart displaying the weighted mean and standard deviation of the log-likelihood over the last epochs for different numbers of CD steps.

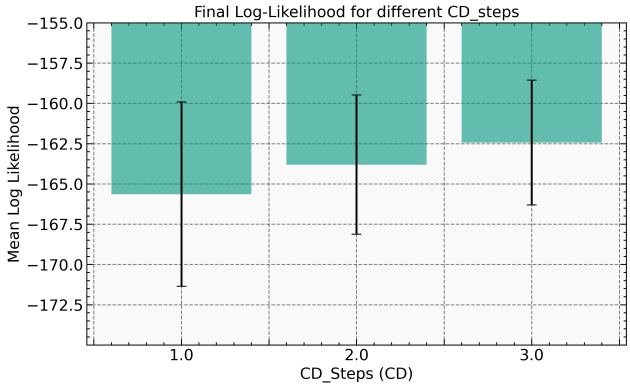

FIG. 9. Bar chart displaying the weighted mean and standard deviation of the log-likelihood over the last epochs for different numbers of CD steps.

- **Influence of activation and encoding schemes:** Our experiments also reveal that one-hot hidden activations (POTS) and spin encoding can lead to instability or degraded performance, highlighting the importance of selecting appropriate unit types and data representations for stable learning [2, 3].

4. DISCUSSION AND CONCLUSION

Our experiments confirm that RBMs are capable of learning useful representations of handwritten digits in an unsupervised fashion. The analysis shows that:

- Proper initialization and regularization are critical for stable training.
- While increasing hidden units improves performance, the benefit saturates beyond a certain network capacity.
- The choice of optimizer and data encoding significantly affects convergence dynamics.

These findings provide a baseline for further research, which might include adaptive scheduling of CD steps, deeper architectures (e.g., stacked RBMs), or alternative estimation techniques (such as AIS). Overall, the work not only demonstrates the practical viability of RBMs but also highlights their limitations and areas for future improvement.

5. APPENDIX

Gradient Update Equations and Regularization

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma \cdot \eta_t \cdot \text{sign}(\boldsymbol{\theta}) \tag{5}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t \pm \eta_t \cdot \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}) \tag{6}$$

$$\begin{cases} \boldsymbol{g}_t = \nabla_{\boldsymbol{\theta}} E(\boldsymbol{\theta}) \\ \boldsymbol{s}_t = \beta \boldsymbol{s}_{t-1} + (1-\beta)\boldsymbol{g}_t^2 \\ \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t \pm \eta_t \frac{\boldsymbol{g}_t}{\sqrt{\boldsymbol{s}_t + \epsilon}} \end{cases} \tag{7}$$

Log-Likelihood and Partition Function Approximations The log-likelihood $\mathcal{L}$ of the data is defined as

$$\mathcal{L} = \frac{1}{M} \sum_{m=1}^{M} \ell_\theta(v^{(m)}) \tag{8}$$

with

$$\ell_\theta(v^{(m)}) = \log \sum_h e^{-E(v^{(m)},h)} - \log Z \tag{9}$$

Here, M is the number of data points and Z denotes the partition function. The energy function $E(v,h)$ is expressed through

$$e^{-E(v,h)} = G(h) \prod_i e^{H_i(h) v_i} \tag{10}$$

where

$$G(h) = \prod_\mu e^{b_\mu h_\mu} \quad \text{and} \quad H_i(h) = a_i + \sum_\mu w_{i\mu} h_\mu \tag{11}$$

In Equation 9, the summation over h runs over all possible states of the hidden layer.

Since directly computing the partition function Z is computationally intractable, we apply approximations and normalization techniques. A more in-depth mathematical analysis about this procedure can be found in [4].

For the Bernoulli encoding, the partition function is manipulated as:

$$Z = q^D \sum_h G(h) \prod_{i=1}^{D} \frac{1 + e^{H_i(h)}}{q} \tag{12}$$

where q is a scaling factor introduced to avoid numerical overflow. For spin encoding, where the units assume values in $\{-1,1\}$, the partition function is approximated by

$$Z = \sum_h G(h) \prod_{i=1}^{D} 2 \cdot \cosh(H_i(h)) \tag{13}$$

In eq. 13, to ensure numerical stability, we relaborated the product introducing a logaritm to avoid overflow. The product becomes via *logarithmic identity*:

$$\log\left(\prod_{i=1}^{N} 2 \cdot \cosh(x_i)\right) = \underbrace{N \cdot \log(2)}_{\text{from } 2^N} + \sum_{i=1}^{N} \log(\cosh(x_i)) \tag{14}$$

we compute this sum instead of multiplying large values directly.

Where the log(cosh) is manipulated using the identity

$$\log(\cosh(x)) = |x| - \log(2) + \log\left(1 + e^{-2|x|}\right) \tag{15}$$

which avoids overflow and underflow. In Python we used NUMPY.LOG1P for precision. The sum over $\log(2\cosh(x_i))$ becomes:

$$\begin{aligned} &\sum_{i=1}^{N} \log(2\cosh(x_i)) = \sum_{i=1}^{N} [\log(2) + \log(\cosh(x_i))] \\ &= \underbrace{N \log(2)}_{\text{explicit } \log(2)} + \sum_{i=1}^{N} \left[|x_i| - \log(2) + \log\left(1 + e^{-2|x_i|}\right) \right] \end{aligned} \tag{16}$$

The $N\log(2)$ term cancels out, leaving:

$$\sum_{i=1}^{N} \left(|x_i| + \log\left(1 + e^{-2|x_i|}\right) \right) \tag{17}$$

That we used in our algorithm to ensure numerical stability.

[1] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, Physics Reports **810**, 1 (2019).

[2] M. Bortoletto, *Study of performances for Restricted Boltzmann Machines*, Final dissertation, Università degli Studi di Padova, Dipartimento di Fisica e Astronomia "Galileo Galilei" (2021).

[3] G. E. Hinton, in *Neural Networks: Tricks of the Trade: Second Edition*, edited by G. Montavon, G. B. Orr, and K.-R. Müller (Springer, 2012) pp. 599–619.

[4] M. Baiesi, "Log-likelihood computation for restricted boltzmann machines (rbms)," (2025), unpublished manuscript.