# GROUP2306 – Restricted Boltzmann Machine Performance

Edoardo Renato Bucalo, Camilla Forza, Anna Garbo, and Federico Ruffini
(Dated: April 2, 2023)

A matrix of one-hot encoded vectors with well defined pattern is used to train a Restricted Boltzmann Machine (RBM) which preserves the one-hot encoding structure. In this paper its performance is discussed and analysed through statistical techniques such as Log-Likelihood and Energy gradient. The study is conducted using both bipolar and binary representations of the same dataset. Some techniques are implemented in order to test the machine and improve the results likes using the RMSprop gradient descent algorithm, using the backward contrastive divergence and also increasing the number of contrastive divergence steps. The performance analysis rewards the binary representation as the best choice for all machines.

## INTRODUCTION

The main purpose of this project is to evaluate the performance of a RBM, which is a neural network algorithm mainly used for unsupervised learning. The RBM consists in a set of visible neurons and hidden neurons linked between each other. A learning algorithm allows the machine to learn the distribution of the input data. Data processable by a RBM is generated with a specific pattern, the one-hot encoding, representing the polarity states of proteins. Hence, the RBM should learn this structure in its learning process. After implementing the RBM code, the performance of the algorithm is studied by varying some parameters and conditions:

- Binary $[0, 1]$ and polar $[-1, 1]$ representation of training data;
- Vanilla Gradient Descent Method and RMSProp as optimiser algorithms;
- Number of Contrastive Divergence (CD) steps;
- Introduction of a Backward Contrastive Divergence (BCD) step which preserves the one-hot encoding structure of the data;
- Variation of the number of the hidden layer to understand which is the best one for the machine to get the data pattern.

## METHODS

During the learning process, the RBM adjusts the weights of the connections between visible and hidden neurons. A joint configuration $(\vec{v}, \vec{h})$ of the visible and hidden units can be associated to an energy [1] given by:

$$E(\vec{v}, \vec{h}, \vec{a}, \vec{b}, \vec{w}) = -\sum_{i \in visible} a_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_i w_{ij} \tag{1}$$

where $v_i, h_j$ are the binary states of visible unit $i$ and hidden unit $j$, $a_i, b_j$ are their biases and $w_{ij}$ is the weight between them. This energy is used to compute the probability associated to each state. In order to start the learning algorithm, weights and biases are randomly initialised from the normal standard distribution. Fig.1 shows the RBM structure in its initial state:

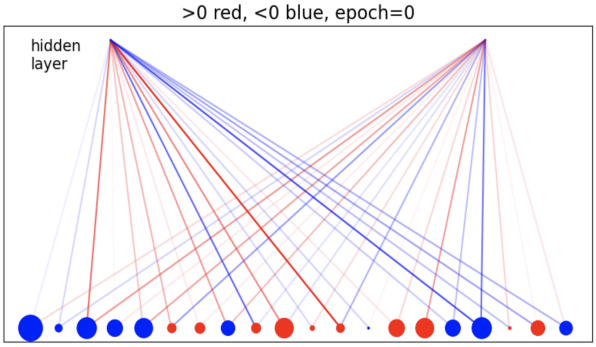



FIG. 1: Initial state of the RBM using 2 hidden units.

The RBM generates data from the model and updates the weights at each training step initially following the pure CD method. The RMSProp optimizer and more CD steps are respectively implemented to develop the second and third objectives. For the last purpose the BCD method is used. The latter method has a further backward step to generate visible units and in this work is built ensuring the preservation of the one-hot encoded structure of the input data. The code generates each one-hot encoded block sampling it from a distribution proportional to its Boltzmann weight. The two quality indicators computed to evaluated the RBM performances are:

- **Log-Likelihood** defined as:

$$\mathcal{L}(\vec{a}, \vec{b}, \vec{w}|\mathcal{D}) = < E >_d - ln(Z) \tag{2}$$

  where $< E >_d$ is the energy mean value on the training data, $Z$ is the partition function and $\mathcal{D}$ the training dataset [1];

- **Energy gradient** computed as :

$$\Delta E(\vec{a}, \vec{b}, \vec{w}|\mathcal{D}, \mathcal{M}) = < E >_d - < E >_{RBM} \tag{3}$$

where $< E >_{RBM}$ is the mean energy calculated over the data model by the RBM [1] and $\mathcal{M}$ is RBM dataset.

A well trained RBM is expected to saturate the Log-Likelihood trend at the maximised value and to decrease the Energy gradient to zero. The same study is performed varying the number of hidden units from 2 to 6 and both with the bipolar and the binary representations. Here in Fig.2 an example of a trained RBM machine scheme after 100 epochs.

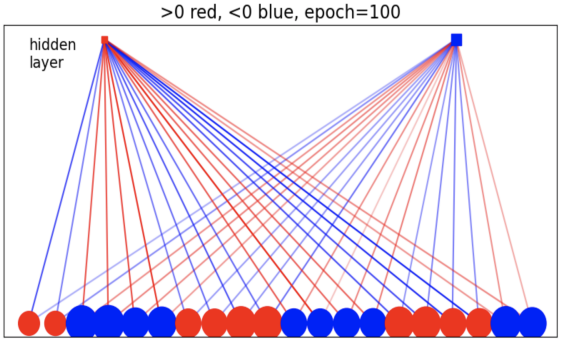



FIG. 2: Final state of a RBM machine with 2 hidden units trained with RMSprop, 2 more CD steps and 100 epochs.

## RESULTS

Firstly, the results of the RBM learning process with two hidden layers using bipolar data representation are presented in the following plots:

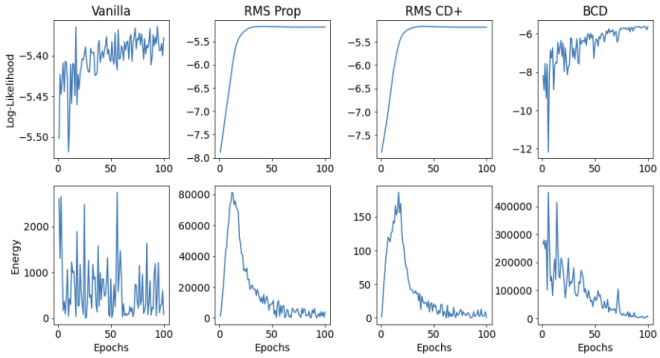



FIG. 3: Plot of different algorithm used to implement RBM machine with Bipolar Representation of dataset. For each algorithm Log-Likelihood and Energy Gradient are shown .

Some consideration on the goodness of the results can be done looking to the Likelihood and to the energy gradient. In the Vanilla and BCD algorithm the energy and

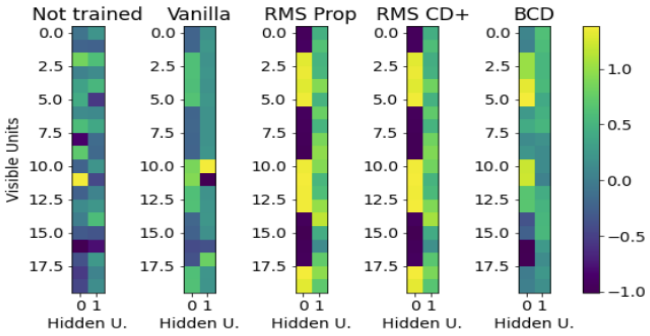



FIG. 4: Colour Plot of weights matrix for bipolar representation of dataset. the weights matrix has been represented after every different kind of algorithm that anlaysed the data.

the likelihood seem to oscillate more than the other two cases, even though at the end they are close to the saturation values. In the first case, the Likelihood appears to be almost stable and fluctuations are around 1%, but the energy gradient presents a lot of peaks not stabilising at zero and the matrix of the weights does not show convergence. The BCD case, instead, presents deep fluctuations in the Likelihood plot and the corresponding matrix of weights is irregular. However, this behaviour could be caused by the training of a RBM that uses a stochastic gradient descent algorithm, which can be a very noisy process subject to random fluctuations. For the second and third case both the energy gradient and log-likelihood plots present a faster and more regular convergence. This is confirmed by the matrices of weights (Fig.4) being easier to read since a partial convergence towards the original data pattern is visible. The process is then repeated using the binary representation. These are the results:

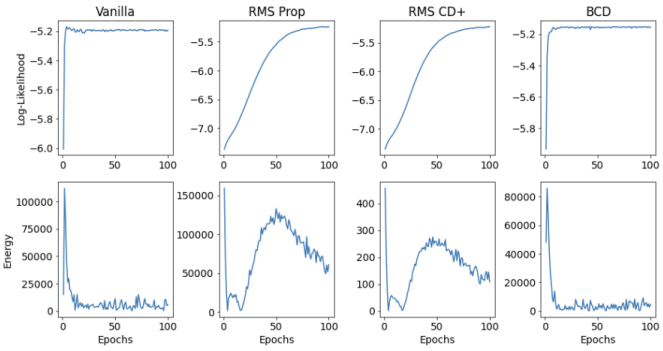



FIG. 5: Plot of different algorithm used to implement RBM machine with Binary Representation of dataset. For each algorithm are shown Log-Likelihood and Energy Gradient.

From the plots can be deduced that the convergence

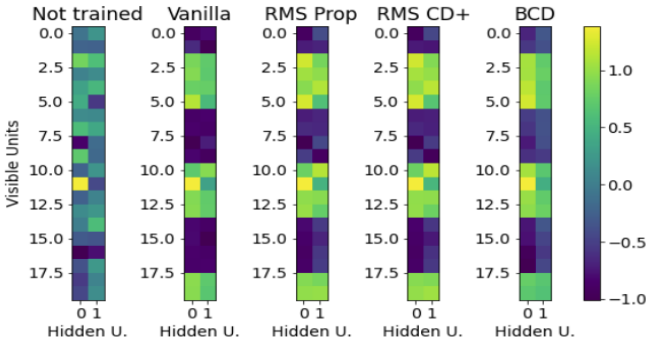



FIG. 6: Colour Plot of weights matrix for binary representation of dataset. the weights matrix has been represented after every different kind of algorithm that analysed the data

both in log-likelihood and in energy is much faster and regular with respect to the case of bipolar representation, especially for the 'Vanilla' and 'BCD' cases (Fig.5). This is also more evident looking at the matrices of weights which are easier to read, in the sense that they reveal the structure underlying the input data used for the training (Fig.6). It is considerable that the Energy in the second and third graph of Fig.5 has a peak in the middle of the epochs range. It could be due to the presence of an overfitting condition during the training of the RBM. Overfitting occurs when the RBM fits too well to the training data and does not generalise well to new data. In such cases, the RBM may try to memorise the training data with too much precision, leading to an increase in energy halfway through training. The same behaviour can also be seen in the bipolar data representation in the same graphs.

As a last step, the number of hidden units is varied in a range from 2 to 6 and all the results can be find in the Appendix. Different behaviours appears for the two representations both in the Likelihood and in the Energy gradient plots. More precisely, it is possible to note that the bipolar configuration preserves approximately the same shape except for the case of the 4 hidden units (Fig.8), where the likelihood presents a decrease and considerable oscillations. In the binary configuration instead is observed the likelihood getting worse increasing the hidden units (from Fig.11 to Fig.14). Considering the structure of our input data, the expected and confirmed result is that 2 hidden units are enough for the machine to understand the underlying structure, because the machine has to reach only two different configurations. The weight matrix is easier to read when we have less hidden units as can be seen in Fig.6 compared to Fig.14, where easier to read means that can see the underlying structure from the colour plot of the the weights matrix.

## CONCLUSION

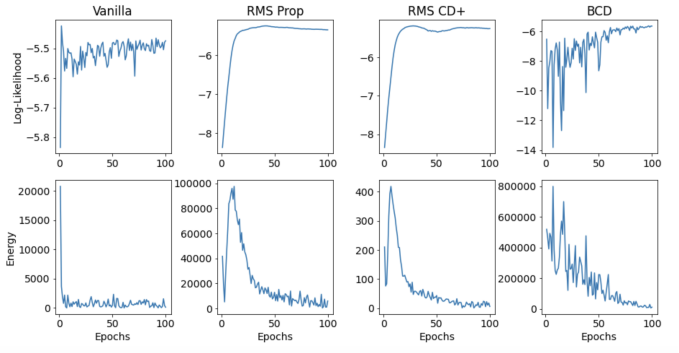
From the previous considerations conclusions are made. the binary representation of the data allows the machine to better learn the structure of the input data even in the case of the hidden layer composed of 2 hidden units or in the others. The choice of using binary or bipolar representation of data for training a Restricted Boltzmann Machine (RBM) depends on the type of data to use and on the characteristics of the model. In general, RBMs are designed to work with binary data [1], indeed the study shows that using bipolar representation the system does not reach the convergence. Another noticeable result is that RBM training becomes more complex and time-consuming [2] when implemented with bipolar data. Additionally, the use of bipolar data slows down the convergence of the learning algorithm which leads to suboptimal solutions.

[1] Decelle, Aurélien, Cyril Furtlehner, and Beatriz Seoane. "Equilibrium and non-equilibrium regimes in the learning of restricted Boltzmann machines." Advances in Neural Information Processing Systems 34 (2021): 5345-5359

[2] Hinton, Geoffrey E. "A practical guide to training restricted Boltzmann machines." Neural Networks: Tricks of the Trade: Second Edition (2012): 599-619.

## APPENDIX

In this section the graphs of the Log-Likelihood and of the Energy gradient are provided together with the matrices of the weights for each choice of hidden units. Figures 7,8, 9 and 10 are related to bipolar representation while 11, 12, 13 and 14 show the results in the binary case.



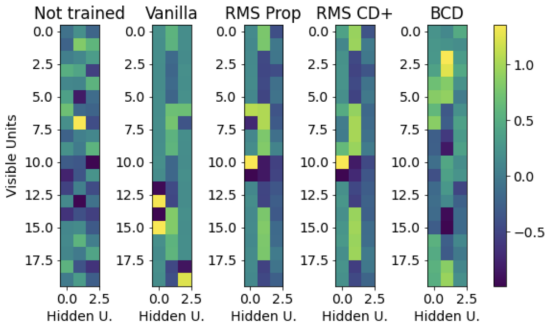
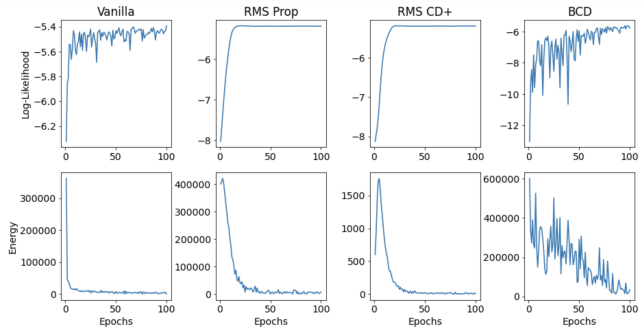




FIG. 7: 3 HIDDEN UNITS Bipolar



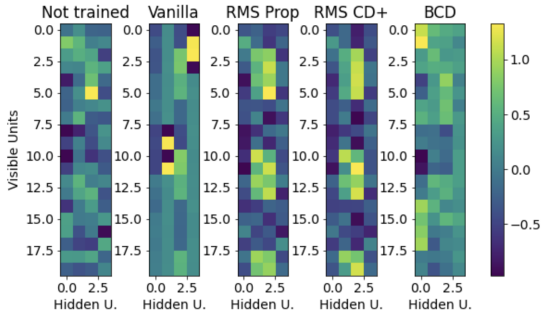
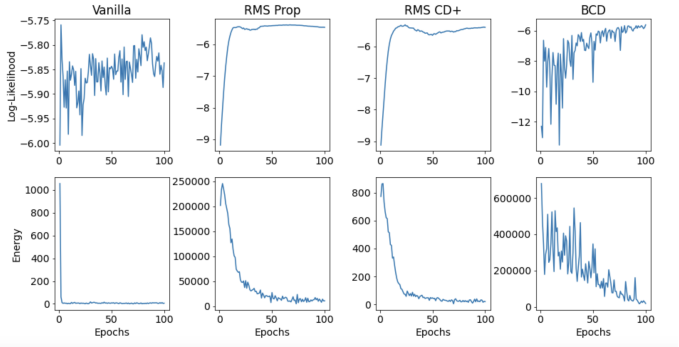




FIG. 8: 4 HIDDEN UNITS Bipolar



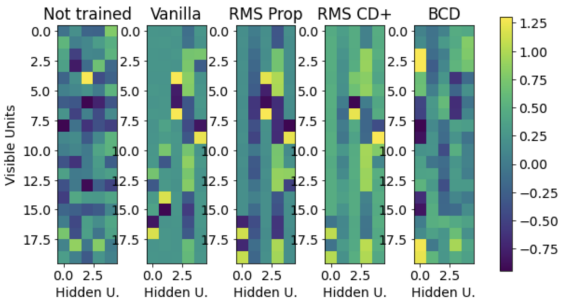
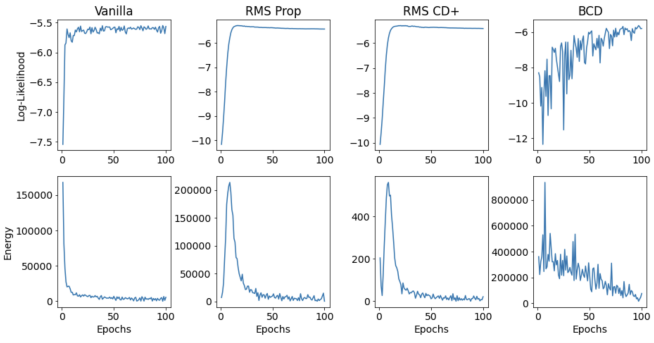




FIG. 9: 5 HIDDEN UNITS Bipolar



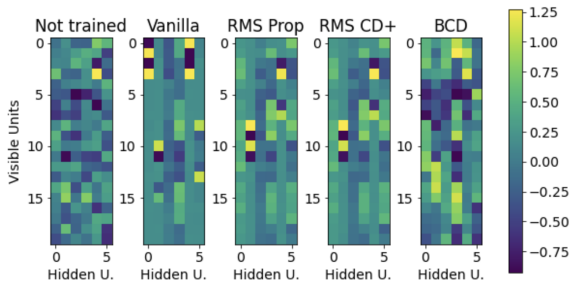




FIG. 10: 6 HIDDEN UNITS Bipolar

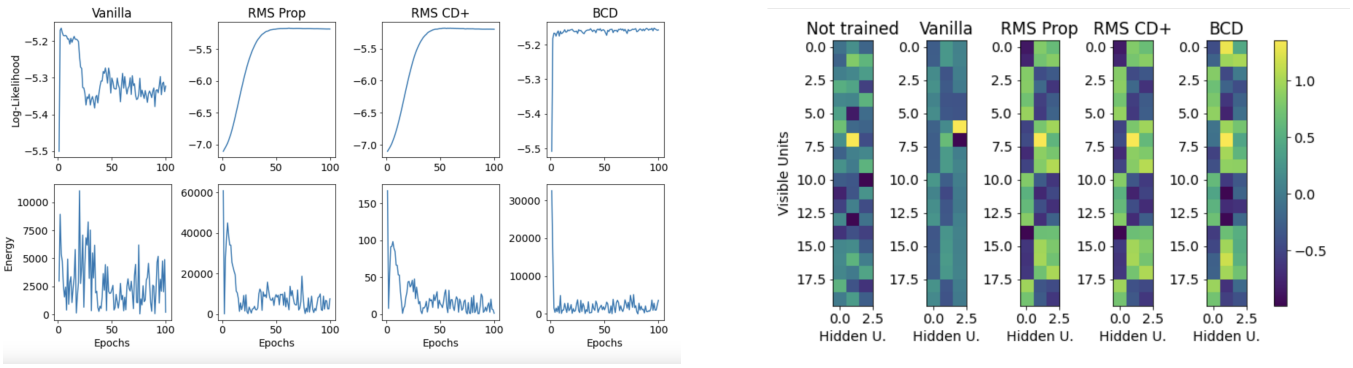



FIG. 11: 3 HIDDEN UNITS Binary

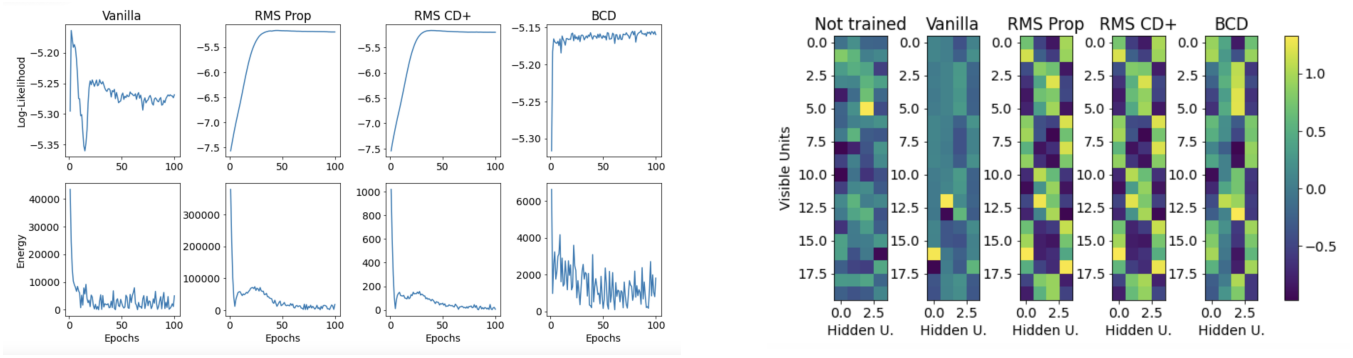



FIG. 12: 4 HIDDEN UNITS Binary

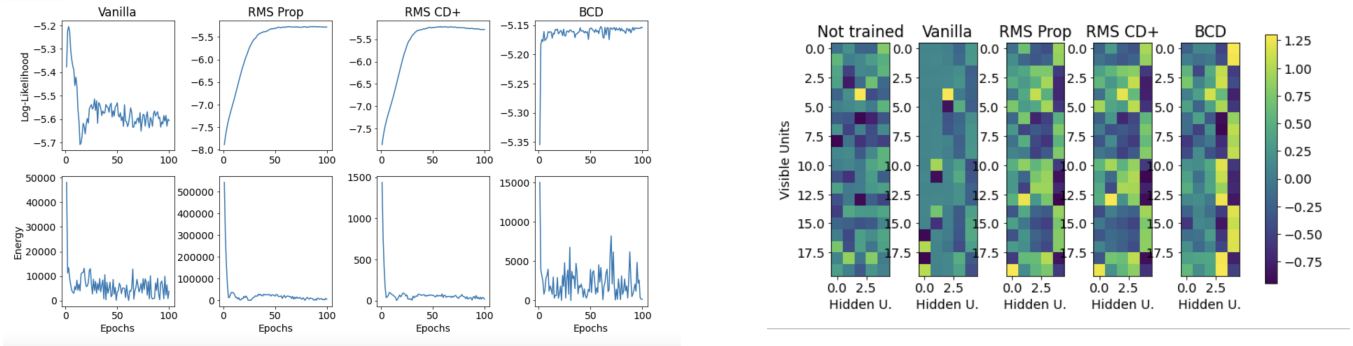



FIG. 13: 5 HIDDEN UNITS Binary

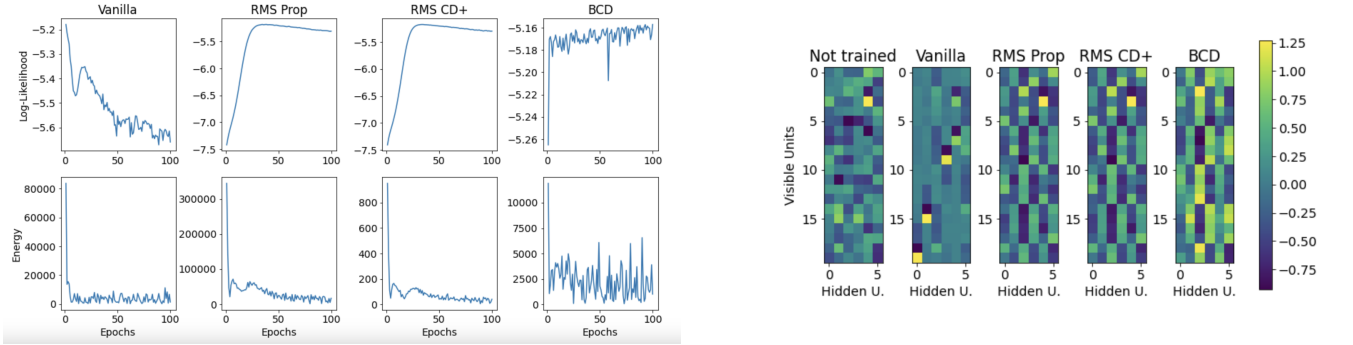



FIG. 14: 6 HIDDEN UNITS Binary