Fine-Tuning Temperatures in Restricted Boltzmann Machines Using Meta-Heuristic Optimization

Mateus Roder, Gustavo Henrique de Rosa, João Paulo Papa
Department of Computing
UNESP - São Paulo State University
Bauru - SP, Brazil
{mateus.roder, gustavo.rosa, joao.papa}@unesp.br

Fabricio Aparecido Breve
Department of Mathematics, Statistics and Computing
São Paulo State University
Rio Claro - SP, Brazil
fabricio.breve@unesp.br

***Abstract*—Restricted Boltzmann Machines (RBM) are stochastic neural networks mainly used for image reconstruction and unsupervised feature learning. An enhanced version, the temperature-based RBM (T-RBM), considers a new temperature parameter during the learning process that influences the neurons' activation. Nevertheless, the major vulnerability of such models concerns selecting an adequate system's temperature, which might lead them to inadequate training or even overfitting when wrongly set, thus limiting the network from predicting or working effectively over unseen data. This paper addresses the problem of selecting a suitable system's temperature through a meta-heuristic optimization process. Meta-heuristic-driven techniques, such as Particle Swarm Optimization, Bat Algorithm, and Artificial Bee Colony are employed to find proper values for the temperature parameter. Additionally, for comparison purposes, three standard temperature values and a random search are used as baselines. The results revealed that optimizing T-RBM is suitable for training purposes, primarily due to their complex fitness landscape, which makes fine-tuning temperatures a non-trivial task.**

***Index Terms*—Image Reconstruction, Restricted Boltzmann Machine, Temperature-based Systems, Meta-Heuristic Optimization**

I. Introduction

Even with the fostering of machine learning techniques, there are still many debatable problems on how to produce accurate representations of the real world, such as recognizing and classifying objects. These techniques have been widely researched in the last years, establishing several hallmarks in a broad range of applications, such as handwriting, face, emotion, and speech recognition, among others.

Most of the techniques usually cope with the parameter set up step, which may give some degree of freedom to work on different problems. In the context of parametrized machine learning techniques, one shall refer to two different denominations: (i) *parameters* and (ii) *hyperparameters*. Usually, the first term stands for low-level parameters that are not controlled by the user, such as the connection weights in neural networks. The latter term refers to high-level parameters that can be adjusted and chosen by the user, such as the temperature, learning rate, and the number of hidden neurons, among others. Both terms are of crucial importance to amend the performance of neural models.

The problem of hyperparameter fine-tuning in machine learning models as a meta-heuristic-driven optimization task has only received attention recently. In 2012, Fedorovici et al. [1] employed the Gravitational Search Algorithm [2] to fine-tune Convolutional Neural Networks (CNN) [3] in the context of optical character recognition. Papa et al. [4] employed Harmony Search in the context of meta-parameters fine-tuning concerning RBM, Discriminative RBM [5], and Deep Belief Networks [6]. Finally, Passos et al. [7] attempted to address this problem in the context of temperature-based Deep Boltzmann Machines.

Another technique has been a considerable spotlight in recent years due to its simplicity, high level of parallelism, and robust representation ability [8]. Restricted Boltzmann Machines (RBM) can be assumed as stochastic neural networks mainly used for image reconstruction and collaborative filtering through unsupervised learning [9]. Recently, an improved version has been proposed by Li et al. [10], the so-called "temperature-based RBM" (T-RBM), which employs a new temperature parameter during the learning process to manipulate the neurons' activation.

Nevertheless, RBM usually suffers from overfitting under the lack of data, which may cause premature convergence and thus poor generalization over unseen data. In order to overcome this issue, different approaches can be highlighted, such as regularization, data augmentation, and hyperparameter fine-tuning. In this paper, the problem of fine-tuning the temperature parameter in T-RBM is modeled as a meta-heuristic-driven optimization task, in which agents encode the values of the temperature in a search problem guided by the reconstruction error over training images. As far as we are concerned, this is the first work that attempted to address the problem of fine-tuning the temperature parameter in T-RBM. In order to validate the proposed approach, we opted to employ Particle Swarm Optimization (PSO) [11], Artificial Bee Colony (ABC) [12] and the Bat Algorithm (BA) [13], since these are well-known and consistent meta-heuristic optimization techniques in the literature. Therefore, the main contributions of this paper are twofold: (i) to introduce meta-

The authors would like to thank São Paulo Research Foundation (FAPESP) grants #2013/07375-0, #2014/12236-1, #2017/25908-6, #2019/02205-5 and #2019/07825-1, as well as National Council for Scientific and Technological Development (CNPq) grants #307066/2017-7 and #427968/2018-6.

heuristic techniques to the context of fine-tuning temperature in T-RBM, and (ii) to fill the lack of research regarding temperature-based RBM.

The remainder of this paper is organized as follows. Sections II and III present some theoretical background concerning RBM, T-RBM, and PSO, respectively, while Section IV discusses the methodology employed in this work. Section V presents the experimental results and Section VI states conclusions and future works.

II. Restricted Boltzmann Machines

Restricted Boltzmann Machines are stochastic neural networks based on energy principles guided by physical laws and characterized by energy, entropy, and temperature factors. Most of the time, these networks learn in an unsupervised fashion and are applicable to a wide variety of problems that range from image reconstruction, collaborative filtering, and feature extraction to pre-training deeper networks.

In terms of architecture, RBM contain a visible layer $\boldsymbol{v}$ with m units and a hidden layer $\boldsymbol{h}$ with n units. Additionally, a real-valued matrix $\boldsymbol{W}_{m\times n}$ models the weights between the visible and hidden neurons, where w_{ij} represents the connection between the visible unit v_i and the hidden unit h_j. Figure 1 describes the well-know vanilla RBM architecture.

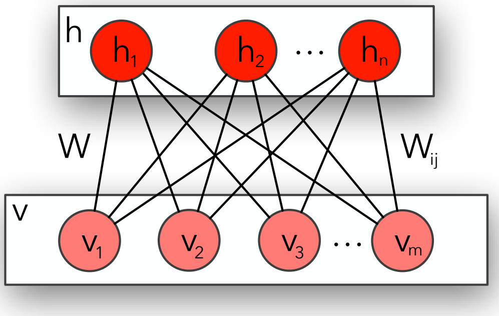

Fig. 1. The vanilla RBM architecture.

Mainly, the visible layer receives the data for processing, while the hidden layer learns its pattern and probabilistic distribution. Additionally, assume all units from layers $\boldsymbol{v}$ and $\boldsymbol{h}$ are binary and derived from a Bernoulli distribution [8], i.e., $\boldsymbol{v}\in\{0,1\}^m$ and $\boldsymbol{h}\in\{0,1\}^n$. Equation 1 models the energy function of an RBM:

$$E(\boldsymbol{v},\boldsymbol{h}) = -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j - \sum_{i=1}^{m}\sum_{j=1}^{n} v_i h_j w_{ij}, \quad (1)$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ represent the biases of visible and hidden units, respectively.

Furthermore, Equation 2 models the joint probability of a given configuration $(\boldsymbol{v},\boldsymbol{h})$:

$$P(\boldsymbol{v},\boldsymbol{h}) = \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{Z}, \quad (2)$$

where Z is the partition function, which normalizes the probability over all possible configurations, considering visible and hidden units. Moreover, Equation 3 represents the marginal probability of an input (visible units) vector:

$$P(\boldsymbol{v}) = \frac{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}}{Z}. \quad (3)$$

Note that the RBM is a bipartite graph, allowing the information to flow in a non-directional manner, from visible to hidden neurons and vice versa. With this property, it is possible to formulate mutually independent activations for both units. Equations 4 and 5 describe their conditional probabilities:

$$P(\boldsymbol{v}|\boldsymbol{h}) = \prod_{i=1}^{m} P(v_i|\boldsymbol{h}) \quad (4)$$

and

$$P(\boldsymbol{h}|\boldsymbol{v}) = \prod_{j=1}^{n} P(h_j|\boldsymbol{v}), \quad (5)$$

where $P(\boldsymbol{v}|\boldsymbol{h})$ and $P(\boldsymbol{h}|\boldsymbol{v})$ stand for the probability of the visible layer given the hidden states and probability of the hidden layer given the visible states, respectively.

From Equations 4 and 5, it is possible to obtain the probability of activating a single visible neuron i given the hidden states, and the probability of activating a single hidden neuron j given the visible states. Equations 6 and 7 describe these activations:

$$P(v_i = 1|\boldsymbol{h}) = \sigma\left(\sum_{j=1}^{n} w_{ij} h_j + a_i\right) \quad (6)$$

and

$$P(h_j = 1|\boldsymbol{v}) = \sigma\left(\sum_{i=1}^{m} w_{ij} v_i + b_j\right), \quad (7)$$

where $\sigma(\cdot)$ stands for the logistic-sigmoid function.

Fundamentally, an RBM needs to learn a set of parameters $\theta = (\boldsymbol{W},\boldsymbol{a},\boldsymbol{b})$ through a training algorithm. One can observe this approach as an optimization problem, which aims to maximize the product of data probabilities for all the training set $\mathcal{V}$, as stated below:

$$\arg\max_{\Theta} \prod_{\boldsymbol{v}\in\mathcal{V}} P(\boldsymbol{v}). \quad (8)$$

An interesting approach to model this problem is applying the negative of the logarithm function, represented by the Negative Log-Likelihood (NLL), which represents the distribution approximation of the reconstructed data over the original data. Therefore, one can employ the partial derivatives of $\boldsymbol{W}$, $\boldsymbol{a}$ and $\boldsymbol{b}$ at iteration t to cope with this problem. Equations 9, 10 and 11 describe the parameters update rules:

$$\boldsymbol{W}^{t+1} = \boldsymbol{W}^{t} + \eta(\boldsymbol{v}P(\boldsymbol{h}|\boldsymbol{v}) - \tilde{\boldsymbol{v}}P(\tilde{\boldsymbol{h}}|\tilde{\boldsymbol{v}})), \quad (9)$$

$$\boldsymbol{a}^{t+1} = \boldsymbol{a}^{t} + (\boldsymbol{v} - \tilde{\boldsymbol{v}}) \tag{10}$$

and

$$\boldsymbol{b}^{t+1} = \boldsymbol{b}^{t} + (P(\boldsymbol{h}|\boldsymbol{v}) - P(\tilde{\boldsymbol{h}}|\tilde{\boldsymbol{v}})), \tag{11}$$

where η stands for the learning rate, $\tilde{\boldsymbol{v}}$ stands for the reconstruction of the visible layer given $\boldsymbol{h}$, and $\tilde{\boldsymbol{h}}$ denotes an estimation of the hidden vector $\boldsymbol{h}$ given $\tilde{\boldsymbol{v}}$.

One interesting proposition by Hinton et al. [14] is to use the training data as the initial visible units, also known as the Contrastive Divergence (CD), which uses the Gibbs sampling method to infer the hidden and visible layers.

A. *Temperature-based Restricted Boltzmann Machines*

An RBM is an approximation of a physical system that interacts with the environment (data). As the temperature factor plays a significant role in influencing the environment, it is also remarkable that the temperate factor affects how the data distribution is modeled.

One can rewrite Equations 6 and 7 into Equations 12 and 13, respectively, and include the temperature factor as follows:

$$P(v_i = 1|\boldsymbol{h}) = \sigma\left(\frac{\sum_{j=1}^{n} w_{ij}h_j + a_i}{T}\right) \tag{12}$$

and

$$P(h_j = 1|\boldsymbol{v}) = \sigma\left(\frac{\sum_{i=1}^{m} w_{ij}v_i + b_j}{T}\right). \tag{13}$$

Considering the equations mentioned above, T is responsible for scaling the system energy and changing the shape of the probability distribution. Also, it is a new hyperparameter that is empirically set or optimized. In other words, it is possible to explore the parameter influence in the learning process, searching for different values that may help the system to learn better features and reduce the reconstruction error.

III. Meta-Heuristic Optimization

A. *Artificial Bee Colony*

Artificial Bee Colony is a nature-inspired algorithm based on honey bee swarms, which is composed of three distinct groups of bees: employees, onlookers, and scouts. Each group has particular importance and function to the swarm, such as choosing a food source, going to the food source, and randomly searching food in new areas [12]. Additionally, the whole bee colony is split in half into employees[1] and onlookers bees. Moreover, when the employee bee exhausts its food source, it becomes a scout bee.

Let $\mathcal{S} = (\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_M)$ be a set of food sources such that $\boldsymbol{s}_i \in \Re^N$ stands for the position of food source i. Also, let $\mathcal{T} = (t_1, t_2, \ldots, t_M)$ be the number of cycles for each food source, known as the "food source trials", which is regulated by the `n_trials` parameter. After exploring a food source or discovering a newer one, bees share their discovered information about the nectar (food). Hence, an onlooker bee chooses a nectar source based on a probability associated with its achieved fitness, as formulated below:

$$p_i = \frac{F_i}{\sum_{k=1}^{M} F_k}, \tag{14}$$

where F_i is the fitness value of food source i.

Finally, one can use Equation 15 to formulate a new food source position, as follows:

$$\boldsymbol{s}_i = \boldsymbol{s}_i + \phi(\boldsymbol{s}_i - \boldsymbol{s}_k), \tag{15}$$

where $i \neq k$ and $\phi \in [-1, 1]$ denotes a random value that controls the bee visualization of other food sources.

B. *Bat Algorithm*

Bat Algorithm is a biological-inspired algorithm proposed by Yang et al. [13] primarily used for solving engineering optimization tasks. It takes into account the advanced capability of the bats' echolocation system, where they have a sonar-like mechanism that enables them to detect food, avoid obstacles, and communicate among themselves.

Mathematically speaking, let $\mathcal{B} = (b_1, b_2, \ldots, b_M)$ be a set of bats that compose the swarm, such that $b_i = (\boldsymbol{x}_i, \boldsymbol{z}_i)$, where $\boldsymbol{x}_i \in \Re^N$ and $\boldsymbol{z}_i \in \Re^N$ stand for the position and velocity of bat i, respectively. Additionally, each bat is associated with a frequency value of $f \in [f_{min}, f_{max}]$, a loudness value of A and a pulse rate of r. Each bat is initialized with random values for its position, velocity, and frequency. During each iteration, Equations 16, 17 and 18 are responsible for updating their frequency, velocity and position values, respectively:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{16}$$

and

$$\boldsymbol{z}_i^{t+1} = \boldsymbol{z}_i^{t} + (\boldsymbol{x}_i^{t} - \hat{\boldsymbol{x}})f_i, \tag{17}$$

and

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^{t} + \boldsymbol{z}_i^{t+1}, \tag{18}$$

where β is a uniform random number between $[0, 1]$, and $\hat{\boldsymbol{x}}$ denotes the current best global position (solution).

Furthermore, if a bat's pulse rate is smaller than a sampled probability p, e.g., $r_i < p$, a new solution is generated around the current best solution. Equation 19 formulates this procedure, as follows:

$$\boldsymbol{x}_i^{t+1} = \hat{\boldsymbol{x}} + \epsilon\bar{A}, \tag{19}$$

where ϵ is a random value between $[-1, 1]$, and $\bar{A}$ is the mean loudness of all bats in the swarm.

[1]An employee bee is only responsible for a single food source.

C. Particle Swarm Optimization

Particle Swarm Optimization is a swarm-based intelligence algorithm inspired by social behavior dynamics [11]. The idea behind employing social behavior learning is to allow each possible solution to move onto the search space, combining details from its previous and current locations with the ones provided by other swarm particles. One can understand this process as a simulation of the social interaction of birds looking for food or even humans trying to achieve a common objective.

Let $\mathcal{D} = (d_1, d_2, \ldots, d_M)$ be a set of particles that compose the swarm, such that $d_i = (\boldsymbol{\psi}_i, \boldsymbol{\rho}_i)$, where $\boldsymbol{\psi}_i \in \Re^N$ and $\boldsymbol{\rho}_i \in \Re^N$ stand for the position and velocity of particle i, respectively. Also, for each particle, we know its best local solution $\hat{\boldsymbol{\psi}}$, as well as the best solution (global) of the entire swarm $\boldsymbol{g}$. Each particle is initialized with random values for both velocity and position. Hence, each individual is evaluated with respect to a given fitness function, thus having its local minimum updated. At the end, the global minimum for each decision variable is updated with the value of the particle that achieved the best position in the swarm. This process is repeated until a convergence criterion is satisfied. Equations 20 and 21 present the update formulation concerning the velocity and position of particle i at time step t, respectively:

$$\boldsymbol{\rho}_i^{t+1} = \mu\boldsymbol{\rho}_i^t + c_1 r_1(\hat{\boldsymbol{\psi}}_i - \boldsymbol{\psi}_i^t) + c_2 r_2(\boldsymbol{g} - \boldsymbol{\psi}_i^t) \tag{20}$$

and

$$\boldsymbol{\psi}_i^{t+1} = \boldsymbol{\psi}_i^t + \boldsymbol{\rho}_i^{t+1}, \tag{21}$$

where μ is the inertia weight that controls the interaction among particles, and $r_1, r_2 \in [0, 1]$ are random variables that give a stochastic trait to PSO. Additionally, variables c_1 and c_2 are constants that conduct the swarm's members onto the search space.

IV. Methodology

In this section, we present the proposed approach to fine-tune the temperature parameter concerning T-RBM, as well as describe the employed datasets and the experimental setup.

A. Modeling T-RBM Temperature Optimization

We propose to model the problem of selecting a suitable temperature parameter considering T-RBM in the task of binary image reconstruction. As aforementioned in Section II, the learning step has three parameters: the learning rate η, the number of hidden units n, and the system's temperature T. As we are interested in fine-tuning the temperature only, we fixed the tuple (η, n) and played with parameter T in order to minimize the mean squared error (MSE) of the reconstructed training images.

For that purpose, we applied T on the probabilities sampled during the contrastive divergence process, once the aim is to support the model to converge. After that, the selected temperature parameter is applied to reconstruct the unseen images of the test set. One can observe that temperature may act as a regularizer since it can modulate the weights during the learning process, thus allowing the model to adjust to data better.

B. Datasets

We considered three datasets in the experimental section, as follows:

- MNIST[2] [3]: set of 28×28 grayscale images of handwritten digits. The original version contains a training set with $60,000$ images from digits '0'-'9', as well as a test set with $10,000$ images.
- Fashion-MNIST[3] [15]: set of 28×28 grayscale images of clothing objects. The original version contains a training set with $60,000$ images from 10 distinct objects (t-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot), as well as a test set with $10,000$ images.
- Kuzushiji-MNIST[4] [16]: set of 28×28 grayscale images of hiragana characters. The original version contains a training set with $60,000$ images from 10 previously selected hiragana characters, as well as a test set with $10,000$ images.

Furthermore, Figure 2 illustrates mosaics of 100 random training samples for every dataset.

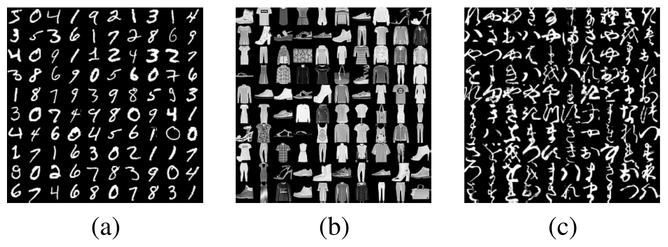

Fig. 2. Random training samples from (a) MNIST, (b) Fashion-MNIST and (c) Kuzushiji-MNIST datasets.

C. Experimental Setup

We fixed each T-RBM[5] architecture parameters according to Table I. Concerning the temperature rates, we set $T \in [0.1, 1.5]$, which means we used such ranges to initialize the optimization techniques. Such a range of values was chosen by previous experimental analysis.

Note that the baseline experiments also use the very same configurations. The main difference lies in the value of T, where the 'low-temperature' T-RBM uses $T = 0.1$ (R_{low}), the 'no-temperature' (R_{no}) T-RBM[6] uses $T = 1$, and the 'high-temperature' version uses $T = 1.5$ (R_{high}). We also have employed 5 epochs for the T-RBM learning weights procedure with mini-batches of size 128. This number of epochs was chosen considering our interest in evaluating

[2]http://yann.lecun.com/exdb/mnist

[3]https://github.com/zalandoresearch/fashion-mnist

[4]https://github.com/rois-codh/kmnist

[5]The T-RBM is available in Learnergy: https://github.com/gugarosa/learnergy

[6]Such a version stands for the traditional RBM.

TABLE I
T-RBM parameters configuration.

Parameter	Value
n (number of hidden neurons)	128, 256, 512
η (learning rate)	0.1
T (temperature)	0.1, 1, 1.5

the meta-heuristics performance, instead of the "long term" RBM learning. All T-RBMs were trained with the Contrastive Divergence [14] algorithm, with $k = 1$ (CD-1).

For every dataset, each meta-heuristic[7] was evaluated under a training and testing sets, as the original datasets suggest[8]. Additionally, for every meta-heuristic, 15 agents (particles) were used over 20 convergence iterations. To provide a thorough and fair comparison among meta-heuristics in the context of T-RBM temperature parameter fine-tuning, we have chosen different techniques, ranging from swarm-based to evolutionary-inspired ones:

- Artificial Bee Colony (ABC);
- Bat Algorithm (BA); and
- Particle Swarm Optimization (PSO).

Table II exhibits the parameter configuration for every meta-heuristic technique[9]. Moreover, we conducted an additional baseline experiment through a random search, i.e., a temperature value sampled randomly from a uniform distribution.

TABLE II
Meta-heuristic algorithms parameters configuration.

Algorithm	Parameters
ABC	$\texttt{n\_trials} = 10$
BA	$f = [0, 2]$ | $A = 0.5$ | $r = 0.5$
PSO	$\mu = 0.7$ | $c_1 = 1.7$ | $c_2 = 1.7$

To perform a reasonable comparison among distinct meta-heuristic techniques, we must rely on mathematical methods that will sustain these observations. The first step is to decide whether to use a parametric or a non-parametric statistical test [17]. Unfortunately, we can not consider a normality state from our experiments due to insufficient data and sensitive outliers, restraining our analysis to non-parametric approaches.

Secondly, acknowledging that our experiments' results are independent (e.g., reconstruction error) and continuous over a particular dependent variable (e.g., number of observations), we can identify that the Wilcoxon signed-rank test [18] will satisfy our obligations. It is a non-parametric hypothesis test used to compare two or more related observations (in our case, repeated measurements over a certain meta-heuristic) to assess whether there are statistically significant differences between them.

[7]All meta-heuristics are available in Opytimizer: https://github.com/gugarosa/opytimizer

[8]Each dataset was evaluated 10 times in the attempt to mitigate the RBMs stochastic nature.

[9]Note that these values were empirically chosen according to their author's definition.

V. Experimental Results

Table III presents the reconstruction error and the associated standard deviation values over the test set, considering all techniques, architectures, and datasets. Remember that, in order to perform a fair comparison among all T-RBMs, we evaluated a new T-RBM using the mean best temperatures found by each meta-heuristic. Furthermore, we calculated their mean and standard deviation values to compose the final results (e.g., PSO found 10 best temperatures, thus producing a mean best temperature, which is further used to evaluate a new T-RBM. The output of each T-RBM was used to calculate the mean, standard deviation, and Wilcoxon's test). Finally, every statistically similar result, according to Wilcoxon's signed-rank test (5% significance), is presented in bold.

First of all, from Table III, it is notable that increasing the models' hidden units, it tends to learn better and generate images with lower reconstruction errors. We can still observe that the meta-heuristic optimizations achieved the lowest mean reconstruction errors and small standard deviations in the test set regarding almost all datasets, architectures, and baseline approaches. Also, notice that these algorithms obtained satisfactory results even in a low-dimensional search space. Observing the baselines R_{low}, R_{no}, and R_{high}, it is possible to note, for all datasets, that an increasing in temperature value also increases the MSE, being $T = 1.5$ the worst one. This fact is explained by the scaling caused by the temperature factor, which tends to increase or decrease the sampling range, i.e., the probabilities of activating or not the neurons.

However, one can note that for Kuzushiji-MNIST, the fewer neurons RBMs have, the more similar are the results. Such a fact points out that, probably, the search space for less hidden neurons has a deeper local minimum, which made the results so similar.

Additionally, in Table IV, one can observe and analyze the best temperature values achieved by the optimization process. For the sake of space, we opted not to include the randomly initialized temperatures in Table IV, allowing us to describe them here. Note that they follow the same [x, y, z] pattern, and are described over MNIST, Fashion-MNIST and Kuzushiji-MNIST, respectively: [1.104, 0.416, 1.423], [0.564, 0.511, 1.468] and [0.683, 0.488, 0.418].

Roughly speaking, low temperatures (the denominators in Equations 12 and 13) increase the input value to the activation function, which causes its saturation quite fastly. The learning process tries to overcome such an issue by decreasing the weights, thus inducing sparsity in the model. Such a process works similarly to other regularization approaches, such as the so-called Dropout [19]. As expected, for each dataset, the temperature values followed a pattern between certain limits, showing that the optimization algorithms were able to converge even with a lower number of iterations. Also, we

TABLE III
MEAN RECONSTRUCTION ERRORS AND THEIR RESPECTIVE STANDARD DEVIATION. EACH CELL IS COMPOSED BY $[x, y, z]$, WHERE x, y AND z BELONGS TO THE 128, 256 AND 512 UNITS ARCHITECTURES, RESPECTIVELY.

Technique	**MNIST**	**F-MNIST**	**K-MNIST**
ABC	[**43.29 ± 0.15**, **34.76 ± 0.09**, **29.39 ± 0.05**]	[**100.95 ± 0.34**, **92.13 ± 0.15**, **88.77 ± 0.23**]	[93.94 ± 0.23, 74.76 ± 0.15, **61.77 ± 0.10**]
BA	[**43.20 ± 0.10**, **34.71 ± 0.05**, **29.45 ± 0.07**]	[**101.14 ± 0.45**, **92.25 ± 0.24**, **88.71 ± 0.35**]	[93.79 ± 0.20, 74.81 ± 0.16, **61.63 ± 0.13**]
PSO	[**43.36 ± 0.08**, **34.76 ± 0.08**, **29.33 ± 0.10**]	[**100.64 ± 0.39**, **92.14 ± 0.15**, **88.72 ± 0.20**]	[94.00 ± 0.25, 74.84 ± 0.15, **61.67 ± 0.10**]
Random	[47.43 ± 0.13, 39.85 ± 0.16, 46.63 ± 0.17]	[101.94 ± 0.58, 93.90 ± 0.25, 118.92 ± 0.29]	[95.87 ± 0.24, **74.35 ± 0.17**, 75.52 ± 0.24]
R_{low}	[46.81 ± 0.14, 52.05 ± 0.67, 66.18 ± 1.57]	[103.23 ± 0.23, 104.29 ± 0.64, 107.83 ± 1.33]	[**93.54 ± 0.10**, 86.84 ± 0.20, 108.19 ± 1.01]
R_{no}	[45.49 ± 0.11, 37.73 ± 0.09, 32.43 ± 0.10]	[111.19 ± 0.28, 98.39 ± 0.13, 94.28 ± 0.16]	[99.46 ± 0.17, 81.56 ± 0.20, 68.37 ± 0.06]
R_{high}	[58.58 ± 0.33, 55.38 ± 0.18, 48.06 ± 0.24]	[125.04 ± 0.28, 123.13 ± 0.23, 119.35 ± 0.30]	[127.51 ± 0.69, 123.59 ± 0.46, 116.52 ± 0.48]

TABLE IV
BEST TEMPERATURES FOUND BY META-HEURISTICS. EACH CELL IS COMPOSED BY $[x, y, z]$, WHERE x, y AND z BELONGS TO THE 128, 256 AND 512 UNITS ARCHITECTURES, RESPECTIVELY.

	MNIST			**F-MNIST**			**K-MNIST**		
Running	**ABC**	**BA**	**PSO**	**ABC**	**BA**	**PSO**	**ABC**	**BA**	**PSO**
1	[0.780, 0.840, 0.929]	[0.712, 0.840, 0.944]	[0.770, 0.844, 0.918]	[0.353, 0.775, 0.905]	[0.385, 0.806, 0.896]	[0.448, 0.778, 0.893]	[0.471, 0.713, 0.894]	[0.200, 0.693, 0.896]	[0.295, 0.715, 0.888]
2	[0.755, 0.849, 0.931]	[0.674, 0.854, 0.944]	[0.816, 0.864, 0.932]	[0.349, 0.759, 0.905]	[0.362, 0.801, 0.905]	[0.366, 0.768, 0.902]	[0.363, 0.685, 0.892]	[0.138, 0.733, 0.878]	[0.367, 0.679, 0.893]
3	[0.787, 0.836, 0.932]	[0.772, 0.870, 0.934]	[0.807, 0.876, 0.931]	[0.359, 0.811, 0.906]	[0.363, 0.771, 0.905]	[0.425, 0.791, 0.907]	[0.369, 0.735, 0.901]	[0.369, 0.709, 0.886]	[0.324, 0.683, 0.878]
4	[0.741, 0.844, 0.935]	[0.717, 0.871, 0.927]	[0.813, 0.848, 0.934]	[0.390, 0.774, 0.902]	[0.389, 0.765, 0.926]	[0.373, 0.762, 0.898]	[0.341, 0.711, 0.887]	[0.220, 0.717, 0.871]	[0.410, 0.719, 0.871]
5	[0.737, 0.854, 0.928]	[0.763, 0.868, 0.934]	[0.815, 0.863, 0.928]	[0.383, 0.776, 0.905]	[0.403, 0.768, 0.889]	[0.386, 0.785, 0.900]	[0.378, 0.703, 0.890]	[0.330, 0.709, 0.866]	[0.272, 0.709, 0.886]
6	[0.764, 0.881, 0.929]	[0.717, 0.826, 0.946]	[0.809, 0.852, 0.930]	[0.370, 0.794, 0.892]	[0.411, 0.784, 0.911]	[0.382, 0.784, 0.903]	[0.381, 0.721, 0.884]	[0.390, 0.718, 0.880]	[0.444, 0.709, 0.876]
7	[0.773, 0.857, 0.927]	[0.713, 0.842, 0.933]	[0.816, 0.856, 0.937]	[0.431, 0.798, 0.903]	[0.400, 0.827, 0.911]	[0.360, 0.767, 0.897]	[0.363, 0.737, 0.878]	[0.410, 0.687, 0.892]	[0.412, 0.708, 0.876]
8	[0.744, 0.827, 0.932]	[0.789, 0.854, 0.943]	[0.802, 0.857, 0.935]	[0.411, 0.783, 0.898]	[0.475, 0.801, 0.894]	[0.426, 0.754, 0.891]	[0.252, 0.708, 0.885]	[0.193, 0.714, 0.889]	[0.413, 0.742, 0.879]
9	[0.727, 0.875, 0.932]	[0.771, 0.843, 0.923]	[0.798, 0.843, 0.928]	[0.408, 0.781, 0.899]	[0.426, 0.770, 0.877]	[0.324, 0.781, 0.898]	[0.360, 0.744, 0.877]	[0.426, 0.703, 0.877]	[0.376, 0.730, 0.890]
10	[0.726, 0.873, 0.925]	[0.679, 0.803, 0.943]	[0.808, 0.842, 0.936]	[0.406, 0.783, 0.900]	[0.447, 0.794, 0.876]	[0.369, 0.806, 0.914]	[0.300, 0.684, 0.889]	[0.459, 0.686, 0.875]	[0.326, 0.730, 0.900]
Mean	[0.753, 0.854, 0.930]	[0.731, 0.847, 0.937]	[0.805, 0.854, 0.931]	[0.386, 0.783, 0.901]	[0.406, 0.789, 0.899]	[0.386, 0.778, 0.900]	[0.358, 0.714, 0.888]	[0.313, 0.707, 0.881]	[0.364, 0.712, 0.884]

can perceive that each dataset has a particular temperature rate, extremely different from $T = 1$, which is used by standard RBM. As already mentioned, the temperature plays a significant role in the RBM learning process, being a suitable hyperparameter to be optimized.

Moreover, to provide better visualization of the optimization process, Figures 3, 4 and 5 illustrate the average MSE convergence across the learning epochs (i.e., training set) considering MNIST, Fashion-MNIST, and Kuzushiji-MNIST, respectively. For that purpose, we used the mean temperature values from Table IV to training the models again and then build the convergence charts.

From all figures, it is possible to observe that all baselines provide the worst reconstruction errors, highlighting the extreme temperature values, i.e., $T = 0.1$ and $T = 1.5$. Additionally, the meta-heuristics were able to converge better than baseline temperatures for all datasets. Nevertheless, it is essential to highlight that these plots evaluate the learning procedure (training set) and use the mean temperature found by the meta-heuristics. It can serve as auxiliary support to understand why fine-tuning the temperature hyperparameter might improve RBMs' learning.

Figures 6, 7 and 8 depict some reconstructed samples using the T-RBM with the simplest architecture, i.e., 128 hidden units, over MNIST, Fashion-MNIST and Kuzushiji-MNIST datasets, respectively, such that:

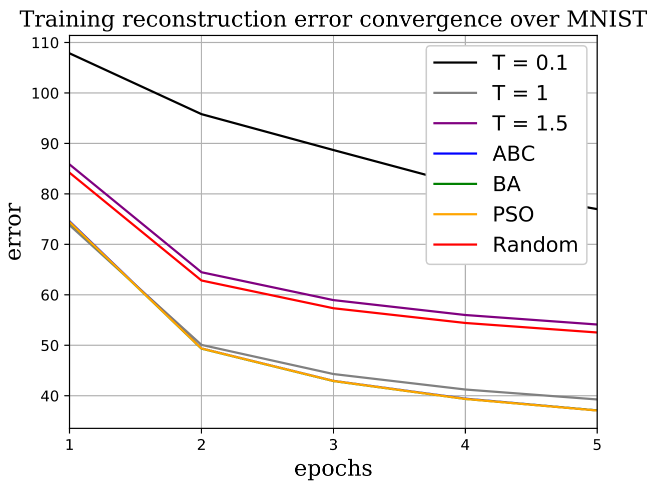

Fig. 3. T-RBM, with 512 hidden units, training reconstruction error convergence over MNIST dataset.

- First quadrant stands for the reconstruction with PSO's mean temperature;
- Second quadrant denotes the reconstruction with $T =$

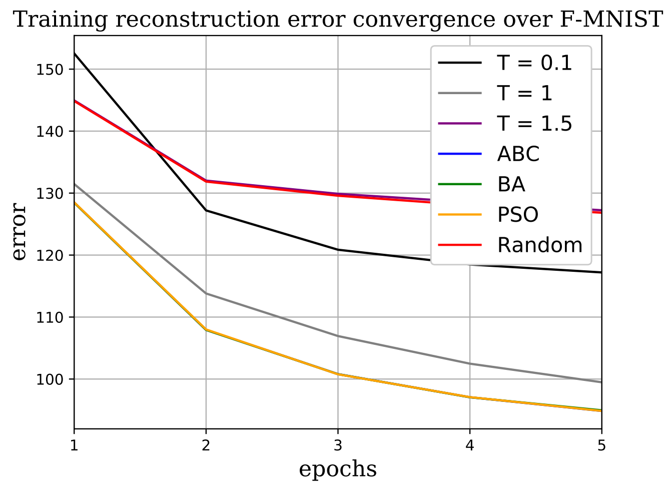

Fig. 4. T-RBM, with 512 hidden units, training reconstruction error convergence over Fashion-MNIST dataset.

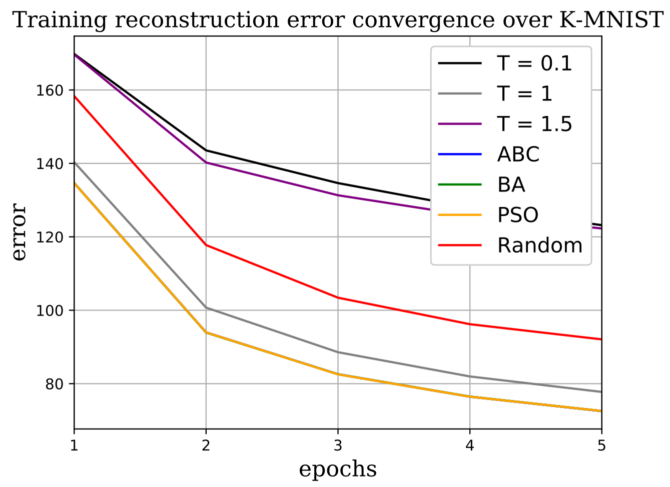

Fig. 5. T-RBM, with 512 hidden units, training reconstruction error convergence over Kuzushiji-MNIST dataset.

0.1;
- Third quadrant is the reconstruction with $T = 1$; and
- Fourth quadrant corresponds to the reconstruction with $T = 1.5$.

As previously discussed, it is clear that temperature hyperparameter plays a significant role in the image reconstruction task. Furthermore, meta-heuristic optimization seems to be a better approach regarding all baselines, as they provided smaller reconstruction errors and generated images with sharper details.

VI. Conclusion

This paper discussed the problem of fine-tuning the temperature in Restricted Boltzmann Machines through meta-heuristic optimization algorithms, such as Artificial Bee Colony, Bat Algorithm, and Particle Swarm Optimization.

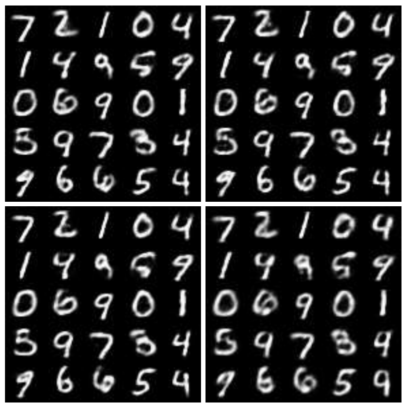

Fig. 6. T-RBM, with 128 hidden units, reconstructed samples over MNIST dataset.

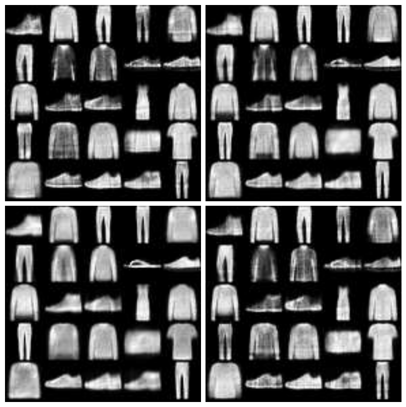

Fig. 7. T-RBM, with 128 hidden units, reconstructed samples over Fashion-MNIST dataset.

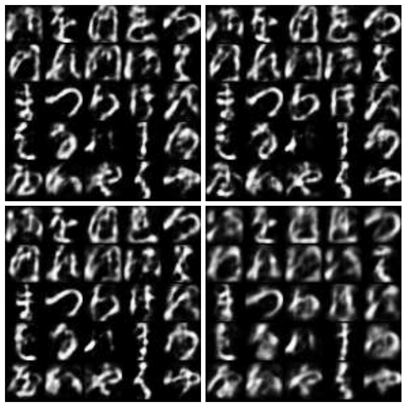

Fig. 8. T-RBM, with 128 hidden units, reconstructed samples over Kuzushiji-MNIST dataset.

All three algorithms were able to fine-tune the temperature parameter in all datasets, supported by the reconstructed error decay. Moreover, it is possible to correlate the reconstruction error directly to the temperature's choice, as RBMs are systems that interact with the data, i.e., environments with intrinsic characteristics. Therefore, meta-heuristics are suitable algorithms to find proper temperatures for distinct problems, achieving better results than naïve approaches, i.e., empirically choosing parameters.

For future works, we aim to explore the correlations between the temperature hyperparameter and the RBM learning step deeply, as well as to consider such an approach in the context of Deep Belief Networks (DBNs) and Deep Boltzmann Machines (DBMs).

References

[1] L. Fedorovici, R. Precup, F. Dragan, R. David, and C. Purcaru, "Embedding gravitational search algorithms in convolutional neural networks for OCR applications," in *7th IEEE International Symposium on Applied Computational Intelligence and Informatics*, ser. SACI '12, 2012, pp. 125–130.

[2] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[4] J. P. Papa, G. H. Rosa, K. A. P. Costa, A. N. Marana, W. Scheirer, and D. D. Cox, "On the model selection of bernoulli restricted boltzmann machines through harmony search," in *Proceedings of the Genetic and Evolutionary Computation Conference*. New York, NY, USA: ACM, 2015, pp. 1449–1450.

[5] J. P. Papa, G. H. Rosa, A. N. Marana, W. Scheirer, and D. D. Cox, "Model selection for discriminative restricted boltzmann machines through meta-heuristic techniques," *Journal of Computational Science*, vol. 9, pp. 14–18, 2015.

[6] J. P. Papa, W. Scheirer, and D. D. Cox, "Fine-tuning deep belief networks using harmony search," *Applied Soft Computing*, 2015.

[7] L. A. Passos and J. P. Papa, "Temperature-based deep boltzmann machines," *Neural Processing Letters*, vol. 48, no. 1, pp. 95–107, 2018.

[8] G. E. Hinton, "A practical guide to training restricted boltzmann machines," in *Neural Networks: Tricks of the Trade*, ser. Lecture Notes in Computer Science, G. Montavon, G. Orr, and K.-R. Müller, Eds. Springer Berlin Heidelberg, 2012, vol. 7700, pp. 599–619.

[9] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[10] G. Li, L. Deng, Y. Xu, C. Wen, W. Wang, J. Pei, and L. Shi, "Temperature based restricted boltzmann machines," *Scientific reports*, vol. 6, p. 19133, 2016.

[11] J. Kennedy and R. Eberhart, *Swarm Intelligence*. M. Kaufman, 2001.

[12] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[13] X.-S. Yang and A. H. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, pp. 464–483, 2012.

[14] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[15] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.

[16] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep learning for classical japanese literature," 2018.

[17] M. Hollander and D. A. Wolfe, "Nonparametric statistical methods," 1999.

[18] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[19] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.