Chapter 3
On the Assessment of Nature-Inspired Meta-Heuristic Optimization Techniques to Fine-Tune Deep Belief Networks

Leandro Aparecido Passos, Gustavo Henrique de Rosa, Douglas Rodrigues, Mateus Roder, and João Paulo Papa

Abstract Machine learning techniques are capable of talking, interpreting, creating, and even reasoning about virtually any subject. Also, their learning power has grown exponentially throughout the last years due to advances in hardware architecture. Nevertheless, most of these models still struggle regarding their practical usage since they require a proper selection of hyper-parameters, which are often empirically chosen. Such requirements are strengthened when concerning deep learning models, which commonly require a higher number of hyper-parameters. A collection of nature-inspired optimization techniques, known as meta-heuristics, arise as straightforward solutions to tackle such problems since they do not employ derivatives, thus alleviating their computational burden. Therefore, this work proposes a comparison among several meta-heuristic optimization techniques in the context of Deep Belief Networks hyper-parameter fine-tuning. An experimental setup was conducted over three public datasets in the task of binary image reconstruction and demonstrated consistent results, posing meta-heuristic techniques as a suitable alternative to the problem.

3.1 Introduction

In the past years, multimedia-based applications fostered the generation of a massive amount of data. These data provide a wide range of opportunities for machine learning applications in several areas of knowledge, such as medicine, financial market, intelligent manufacturing, and event classification. Among such machine

L. A. Passos (✉) · G. H. de Rosa · M. Roder · J. P. Papa
Department of Computing, São Paulo State University, Bauru, Brazil
e-mail: leandro.passos@unesp.br; gustavo.rosa@unesp.br; mateus.roder@unesp.br; joao.papa@unesp.br

D. Rodrigues
Department of Computing, São Carlos Federal University, São Carlos, Brazil

H. Iba, N. Noman (eds.), *Deep Neural Evolution*, Natural Computing Series,
https://doi.org/10.1007/978-981-15-3685-4_3

learning approaches, deep learning methods have received significant attention due to their excellent results, often surpassing even humans.

Deep learning models try to simulate the human-brain behavior on how the information is processed. The basic idea is to use multiple layers to extract higher-level features progressively, where each layer learns to transform input data into a more abstract representation. Regarding applications in the image processing area, lower layers may identify edges, while higher layers may identify human-meaningful items such as human faces and objects. Among the most employed methods, one can include Convolutional Neural Networks (CNNs) [8], Deep Belief Networks (DBNs) [5], and Deep Boltzmann Machines (DBMs) [23], among others.

Since "deep" in deep learning refers to the architecture complexity, the more complex it becomes, the higher the number of hyper-parameters to fit. Yosinski and Lipson [36], for instance, highlighted some approaches for visualizing the behavior of a single Restricted Boltzmann Machine (RBM) [24], which is an energy-based model that can be used to build DBNs and DBMs, during its learning procedure, and provided an overview toward such complexities comprehension. Such a problem was usually tackled using auto-learning tools, which combine parameter fine-tuning with feature selection techniques [26]. Despite, it can also be posed as an optimization task in which one wants to choose suitable hyper-parameters.

Therefore, meta-heuristic algorithms have become a viable alternative to solve optimization problems due to their simple implementation. Kuremoto et al. [7], for instance, employed the Particle Swarm Optimization (PSO) [6] to the context of hyper-parameter fine-tuning concerning RBMs, while Liu et. al [10] and Levy et al. [9] applied Genetic Algorithms (GA) [29] for model selection and automatic painter classification using RBMs, respectively. Later, Rosa et al. [22] addressed the Firefly Algorithm to fine-tune DBN hyper-parameters. Finally, Passos et al. [15, 16] proposed a similar approach comparing several meta-heuristic techniques to fine-tune hyper-parameters in DBMs, infinity Restricted Boltzmann Machines [13, 18], and RBM-based models in general [14].

Following this idea, this chapter presents a comparison among ten different swarm- and differential evolution-based meta-heuristic algorithms in the context of fine-tuning DBN hyper-parameters. We present a discussion about the viability of such approaches in three public datasets, as well as the statistical evaluation through the Wilcoxon signed-rank test. The remainder of this chapter is organized as follows. Section 3.2 introduces the theoretical background concerning RBMs and DBNs. Sections 3.4 and 3.5 present the methodology and the experimental results, respectively. Finally, Sect. 3.6 states conclusions and future works.

3.2 Theoretical Background

In this section, we present a theoretical background concerning Restricted Boltzmann Machines and Deep Belief Networks.

3.2.1 Restricted Boltzmann Machines

Restricted Boltzmann Machines are well-known stochastic-nature neural networks inspired by physical laws of statistical mechanics and parameterized by concepts like energy and entropy. These networks are commonly employed in the field of unsupervised learning, having at least two layers of neurons, i.e., one visible and one hidden.

The Restricted Boltzmann Machine basic architecture is composed of a visible layer $\mathbf{v} = \{v_1, v_2, \ldots, v_m\}$ with m units and a hidden layer $\mathbf{h} = \{h_1, h_2, \ldots, h_n\}$ with n units. Furthermore, a real-valued matrix $\mathbf{W}_{m \times n}$ is responsible for modeling the restricted connections, i.e., 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 3.1 describes the vanilla RBM architecture.

Regarding the learning process, a layer composed of visible units represents the input data to be processed, while the hidden layer is employed to extract deep-seated patterns and information from this data. Besides, both visible and hidden units assume only binary values, i.e., $\mathbf{v} \in \{0, 1\}^m$ and $\mathbf{h} \in \{0, 1\}^n$, once sampling process is derived from a Bernoulli distribution [4]. Finally, the training process is performed by minimizing the system's energy considering both the visible and hidden layers units, as well as the biases associated with each layer. The energy can be computed as follows:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j - \sum_{i=1}^{m} \sum_{j=1}^{n} v_i h_j w_{ij}, \tag{3.1}$$

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

Computing the system's probability is an intractable task due to the computational cost. However, one can estimate the probability of activating a single visible neuron i given the hidden units through Gibbs sampling over a Markov chain, as follows:

$$P(v_i = 1|\mathbf{h}) = \phi\left(\sum_{j=1}^{n} w_{ij} h_j + a_i\right), \tag{3.2}$$

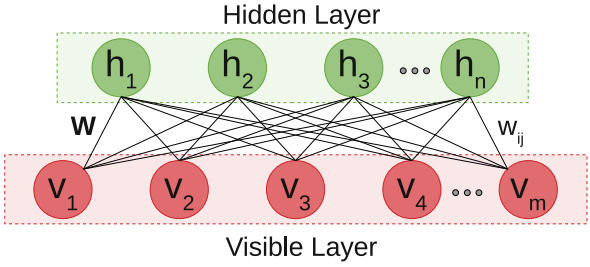

Fig. 3.1 Vanilla RBM architecture

and, in a similar fashion, the probability of activating a single hidden neuron j given the visible units is stated as follows:

$$P(h_j = 1|\mathbf{v}) = \phi\left(\sum_{i=1}^{m} w_{ij} v_i + b_j\right), \tag{3.3}$$

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

The training process consists of maximizing the product of probabilities given a set of parameters $\theta = (W, a, b)$ and the data probability distribution over the training samples. Such a process can be easily computed using either the Contrastive Divergence (CD) [3] or the Persistent Contrastive Divergence (PCD) [27] algorithms.

3.2.2 Contrastive Divergence

Hinton [3] introduced a faster methodology to compute the energy of the system based on contrastive divergence. The idea is to initialize the visible units with a training sample, to compute the states of the hidden units using Eq. (3.3), and then to compute the states of the visible unit (reconstruction step) using Eq. (3.2). In short, this is equivalent to perform Gibbs sampling using $k = 1$ and to initialize the chain with the training samples.

Therefore, the equation below leads to a simple learning rule for updating the weights matrix $\mathbf{W}$, and biases $\mathbf{a}$ and $\mathbf{b}$ at iteration t:

$$\mathbf{W}^{t+1} = \mathbf{W}^t + \underbrace{\eta(P(\mathbf{h}|\mathbf{v})\mathbf{v}^T - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})\tilde{\mathbf{v}}^T) + \Phi}_{=\Delta\mathbf{W}^t}, \tag{3.4}$$

$$\mathbf{a}^{t+1} = \mathbf{a}^t + \underbrace{\eta(\mathbf{v} - \tilde{\mathbf{v}}) + \varphi\Delta\mathbf{a}^{t-1}}_{=\Delta\mathbf{a}^t}, \tag{3.5}$$

$$\mathbf{b}^{t+1} = \mathbf{b}^t + \underbrace{\eta(P(\mathbf{h}|\mathbf{v}) - P(\tilde{\mathbf{h}}|\tilde{\mathbf{v}})) + \varphi\Delta\mathbf{b}^{t-1}}_{=\Delta\mathbf{b}^t}, \tag{3.6}$$

where η stands for the learning rate, φ denotes the momentum, $\tilde{\mathbf{v}}$ stands for the reconstruction of the visible layer given $\mathbf{h}$, and $\tilde{\mathbf{h}}$ denotes an estimation of the hidden vector $\mathbf{h}$ given $\tilde{\mathbf{v}}$. In a nutshell, Eqs. (3.4), (3.5), and (3.6) show the optimization algorithm, the well-known Gradient Descent. The additional term Φ in Eq. (3.4) is used to control the values of matrix $\mathbf{W}$ during the convergence process, and it is described as follows:

$$\Phi = -\lambda\mathbf{W}^t + \varphi\Delta\mathbf{W}^{t-1}, \tag{3.7}$$

where λ stands for the weight decay.

3.2.3 Persistent Contrastive Divergence

Most of the issues related to the Contrastive Divergence approach concern the number of iterations employed to approximate the model to the real data. Although the approach proposed by Hinton [3] takes $k = 1$ and works well for real-world problems, one can settle different values for k [1].[1]

Notwithstanding, Contrastive Divergence provides a good approximation to the likelihood gradient, i.e., it gives a reasonable estimation of the model to the data when $k \to \infty$. However, its convergence might become poor when the Markov chain has a "low mixing," as well as a good convergence only on the early iterations, getting slower as iterations go by, thus, demanding the use of parameters decay.

Therefore, Tieleman [27] proposed the Persistent Contrastive Divergence, an interesting alternative for contrastive divergence using higher values for k while keeping the computational burden relatively low. The idea is quite simple: on CD-1, each training sample is employed to start an RBM and rebuild a model after a single Gibbs sampling iteration. Once every training sample is presented to the RBM, we have a so-called epoch. The process is repeated for each next epoch, i.e., the same training samples are used to feed the RBM, and the Markov chain is restarted at each epoch.

3.2.4 Deep Belief Networks

Deep Belief Networks [5] are graphical models composed of a visible and L hidden layers, where each layer is connected to the latter through a weight matrix $\mathbf{W}_l$, $l \in \{1, 2, \ldots, L\}$, and there is no connection between units from the same layer. In a nutshell, one can consider each set of two subsequent layers as an RBM trained in a greedy fashion such that the trained hidden layer of the bottommost RBM feeds the next RBM's visible layer, and so on. Figure 3.2 depicts the model. Notice $\mathbf{v}$ and $\mathbf{h}_l$ stand for the visible and the l-th hidden layers.

Although this work focuses on image reconstruction, one can use DBNs for supervised classification tasks. Such an approach requires, after the greedy feed-forward pass mentioned above, fine-tuning the network weights using either backpropagation or gradient descent. Afterward, a softmax layer is added at the top of the model to attribute the predicted labels.

[1] Usually, contrastive divergence with a single iteration is called CD-1.

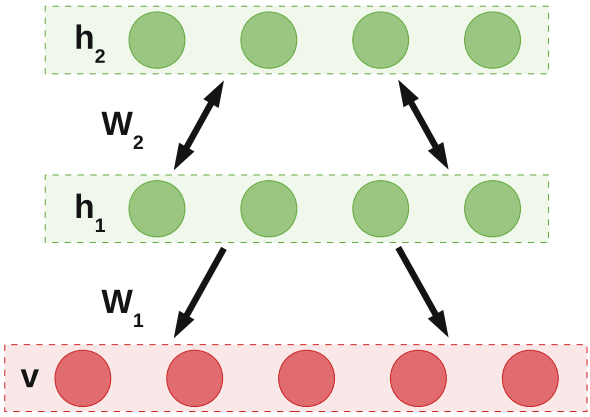

Fig. 3.2 DBN architecture with two hidden layers

3.3 Meta-heuristic Optimization Algorithms

This section presents a brief description of the meta-heuristic optimization techniques employed in this work.

- Improved Harmony Search (IHS) [11]: an improved version of the Harmony Search optimization algorithm that employs dynamic values for both the Pitch Adjusting Rate (PAR), considering values in the range [PAR_{min},PAR_{max}], and the Harmony Memory Considering Rate (HMCR), which assumes values in the range [HMCR_{min},HMCR_{max}]. Additionally, the algorithm uses the bandwidth variable ϱ in the range [ϱ_{min}, ϱ_{max}] to calculate PAR.
- Particle Swarm Optimization with Adaptive Inertia Weight (AIWPSO) [12]: an improved version of the Particle Swarm Optimization that employs self-adjusting inertia weights w over each particle along with the search space aiming to balance the global exploration and local exploitation. Notice the method uses the variables c_1 and c_2 to control the particles' acceleration.
- Flower Pollination Algorithm (FPA) [21, 35]: a meta-heuristic optimization algorithm that tries to mimic the pollination process performed by flowers. The algorithm employs four basic rules: (1) the cross-pollination, which stands for the pollination performed by birds and insects, (2) the self-pollination, representing the pollination performed by the wind diffusion or similar approaches, (3) the constancy of birds/insects, representing the probability of reproduction, and (4) the interaction of local and global pollination, controlled by the probability parameter p. Additionally, the algorithm employs an additional parameter β to control the amplitude of the distribution.
- Bat Algorithm (BA) [34]: based on the bats' echolocation system while searching for food and prey. The algorithm employs a swarm of virtual bats randomly flying in the search space at different velocities, even following a random walk approach for local search intensification. Additionally, it applies a dynamically updated wavelength frequency in the range $\{f_{\text{min}}, f_{\text{max}}\}$ according to the distance from the objective, as well as loudness A and the pulse rate r.

- Firefly Algorithm (FA) [31]: the algorithm is based on the fireflies' approach for attracting potential preys and mating partners. It employs the attractiveness β parameter, which influences the brightness of each agent, depending on its position and light absorption coefficient γ. Moreover, the model employs a random perturbation α used to perform a random walk and avoid local optima.
- Cuckoo Search (CS) [20, 32, 33]: the model combines some cuckoo species parasitic behavior with a τ-step random walk over a Markov chain. It employs three basic concepts: (1) each cuckoo lays a single egg for iteration at another bird's randomly chosen nest, (2) $p_a \in [0, 1]$ defines the probability of this bird discover and discard the cuckoo's egg or abandon it and create a new chest, i.e., a new solution, and (3) the nests with best eggs will carry over to the next generations.
- Differential Evolution (DE) [25]: evolution algorithm maintains a population of candidate solutions which are combined and improved in following generations aiming to find the characteristics that best fit the problem. The algorithm employs a mutation factor to control the mutation amplitude, as well as a parameter to control the crossover probability.
- Backtracking Search Optimization Algorithm (BSA) [2, 17]: an evolution algorithm that employs a random selection of a historical population for mutation and crossover operations to generate a new population of individuals based on past experiences. The algorithm controls the number of elements to be mutated using a mixing rate (mix_rate) parameter, as well as the amplitude of the search-direction matrix with the parameter F.
- Differential Evolution Based on Covariance Matrix Learning and Bimodal Distribution Parameter Setting Algorithm (CoBiDE) [28]: a differential evolution model that represents the search space coordinate system using a covariance matrix according to the probability parameter P_b, and the proportion of individuals employed in the process using the P_s variable. Moreover, it employs a binomial distribution to control the mutation and crossover rates, aiming a better trade-off between exploitation and exploration.
- Adaptive Differential Evolution with Optional External Archive (JADE) [19, 37]: JADE is a differential evolution-based algorithm that employs the "DE/current-to-p-best" strategy, i.e., only the $p - best$ agents are used in the mutation process. Further, the algorithm employs both a historical population and a control parameter, which is adaptively updated. Finally, it requires a proper selection of the rate of adaptation parameter c, as well as the mutation greediness parameter g.

3.4 Methodology

This section introduces the intended procedure for DBN hyper-parameter fine-tuning. Additionally, it describes the employed datasets and the experimental setup.

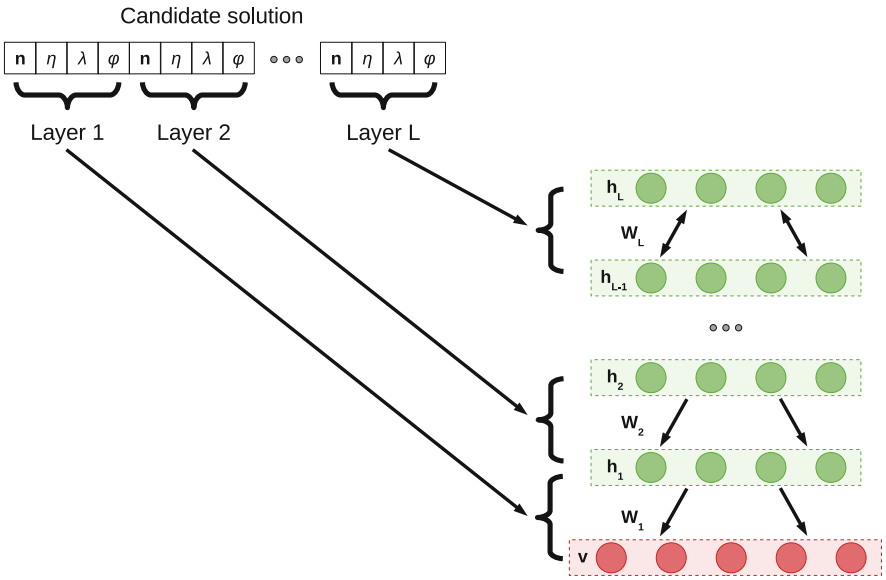

Fig. 3.3 DBN hyper-parameter optimization approach

3.4.1 Modeling DBN Hyper-parameter Fine-tuning

The learning procedure of each RBM employs four hyper-parameters, as specified in Sect. 3.2.1: the learning rate η, weight decay λ, momentum φ, and the number of hidden units n. Since DBNs are built over RBM blocks, they employ a similar process to fine-tune each of their layers individually. In short, a four-dimensional search space composed of three real- and one integer-valued variables should be selected for each layer. Notice the variable values are intrinsically real numbers, thus requiring a type casting to obtain the nearest integer. Such an approach aims at electing the assortment of DBN hyper-parameters that minimizes the training images reconstruction error, denoted by the minimum squared error (MSE). Subsequently, the selected set of parameters is applied to reconstruct the unseen images of the test set. Figure 3.3 depicts the procedure.

3.4.2 Datasets

We employed three datasets, as described below:

- MNIST dataset[2]: a dataset composed of "0"–"9" handwritten digits images. Regarding the pre-processing, the images were converted from gray-scale to

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

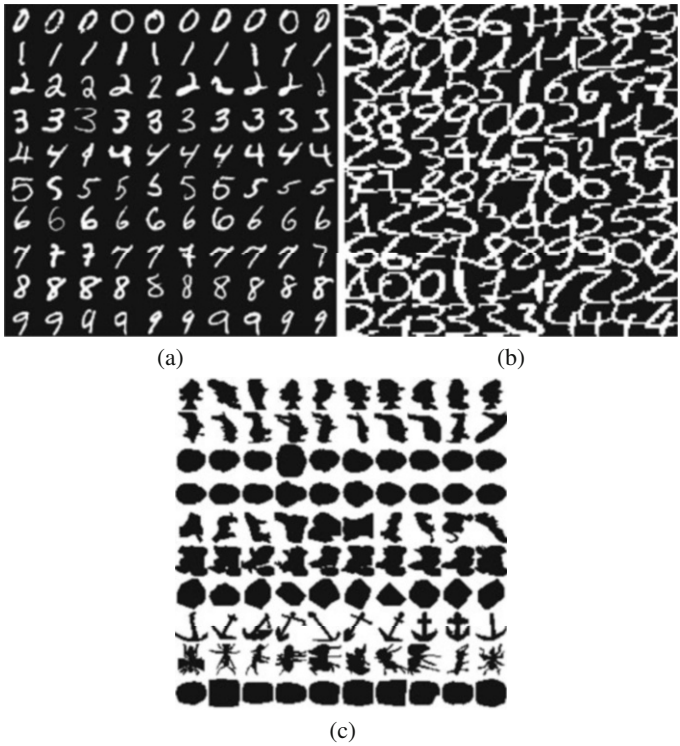

Fig. 3.4 Some training examples from (**a**) MNIST, (**b**) Semeion, and (**c**) CalTech 101 Silhouettes datasets

binary, as well as resized to 14 × 14. Additionally, the training was performed over 2% of the training set, i.e., 1200 images, due to the demanded computational burden. Moreover, the complete set of 10,000 was employed for testing.

- Semeion Handwritten Digit Dataset[3]: similar to the MNIST, Semeion is also a dataset composed of "0"–"9" handwritten digits images formed by 1593 images. In this paper, we resized the samples to 16 × 16 and binarized each pixel.
- CalTech 101 Silhouettes Dataset[4]: a dataset composed of 101 classes of silhouettes with a resolution of 28 × 28. No pre-processing step was applied to the image samples.

Figure 3.4 displays some training examples from the above datasets.

[3] https://archive.ics.uci.edu/ml/datasets/Semeion+Handwritten+Digit.

[4] https://people.cs.umass.edu/~marlin/data.shtml.

Table 3.1 Meta-heuristic algorithms' parameter configuration

Algorithm	Parameters
IHS	$HMCR = 0.7 \mid PAR_{MIN} = 0.1$
	$PAR_{MAX} = 0.7 \mid \varrho_{MIN} = 1$
	$\varrho_{MAX} = 10$
AIWPSO	$c_1 = 1.7 \mid c_2 = 1.7$
	$w = 0.7 \mid w_{MIN} = 0.5 \mid w_{MAX} = 1.5$
FPA	$\beta = 1.5 \mid p = 0.8$
BA	$f_{\min} = 0 \mid f_{\max} = 100 \mid A = 1.5 \mid r = 0.5$
FA	$\alpha = 0.2 \mid \beta = 1.0 \mid \gamma = 1.0$
CS	$\beta = 1.5 \mid p = 0.25 \mid \alpha = 0.8$
BSA	$mix\_rate = 1.0 \mid F = 3$
CoBiDE	$P_b = 0.4 \mid P_s = 0.5$
DE	$mutation\_factor = 0.8$
	$cross\_over\_probability = 0.7$
JADE	$c = 0.1 \mid g = 0.05$

3.4.3 Experimental Setup

Experiments were conducted over 20 runs and a 2-fold cross-validation for statistical analysis using the Wilcoxon signed-rank test [30] with 5% of significance. Each meta-heuristic technique employed five agents (particles) over 50 iterations for convergence purposes over the three configurations, i.e., DBNs with 1, 2, and 3 layers. Additionally, the paper compares different techniques ranging from music composition process, swarm-based, and evolutionary-inspired methods, in the context of DBN hyper-parameter fine-tuning, as presented in Sect. 3.3:

Table 3.1 exhibits the parameter configuration for every meta-heuristic technique.[5]

Finally, each DBN layer is composed of an RBM whose hyper-parameters are randomly initialized according to the following ranges: $n \in [5, 100]$, $\eta \in [0.1, 0.9]$, $\lambda \in [0.1, 0.9]$, and $\varphi \in [10^{-5}, 10^{-1}]$. Additionally, the experiments were conducted over three different depth configurations, i.e., DBNs composed of 1, 2, and 3 RBM layers, which implies on fine-tuning a $4-$, $8-$, and $12-$dimensional set of hyper-parameters. We also have employed $T = 10$ as the number of epochs for DBN learning weights procedure with mini-batches of size 20. In order to present a more in-depth experimental validation, all DBNs were trained with the Contrastive Divergence (CD) [3] and Persistent Contrastive Divergence (PCD) [27]. Figure 3.5 depicts the pipeline proposed in this paper.

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

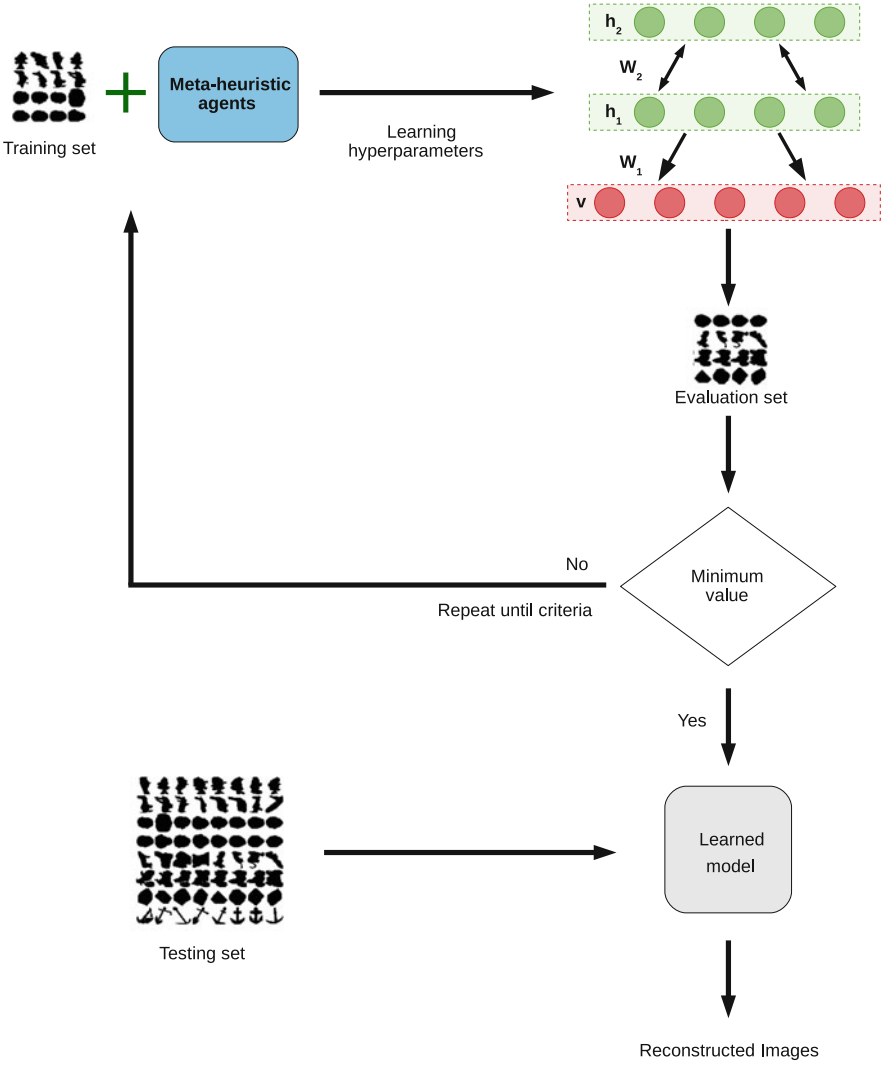

Fig. 3.5 Proposed pipeline to the task of DBN hyper-parameter fine-tuning

3.5 Experimental Results

This section introduces the results obtained during the experiments. Further, a detailed discussion about them is provided. Tables 3.2, 3.3, and 3.4 present the average MSE, and their standard deviation regarding MNIST, Semeion Handwritten Digit, and CalTech 101 Silhouettes datasets, respectively. The best results accordingly to the Wilcoxon signed-rank test with 5% of significance level are presented in bold.

Table 3.2 Average MSE values considering MNIST dataset

Layer	Alg	Statistics	IHS	AIWPSO	BA	CS	FA	FPA	BSA	CoBiDE	DE	JADE	RANDOM
1	CD	Mean	**0.08759**	0.08765	0.08762	0.08764	0.08764	**0.08759**	0.08763	0.08763	0.08762	0.08761	0.08763
		Std	0.00008	0.00006	0.00007	0.00005	0.00007	0.00009	0.00005	0.00006	0.00005	0.00007	0.00006
	PCD	Mean	0.08762	0.08762	0.08763	0.08764	0.08765	0.08761	0.08762	0.08762	0.08763	0.08763	0.08764
		Std	0.00008	0.00005	0.00006	0.00007	0.00006	0.00007	0.00007	0.00007	0.00006	0.00006	0.00005
2	CD	Mean	0.08762	0.08764	0.08763	0.08764	0.08764	0.08762	0.08763	0.08764	0.08763	0.08763	0.08763
		Std	0.00005	0.00006	0.00007	0.00007	0.00006	0.00006	0.00007	0.00006	0.00005	0.00006	0.00004
	PCD	Mean	0.08763	0.08763	**0.08763**	0.08765	0.08763	0.08766	0.08764	0.08764	0.08762	0.08765	0.08764
		Std	0.00006	0.00005	0.00007	0.00006	0.00006	0.00006	0.00006	0.00005	0.00005	0.00006	0.00005
3	CD	Mean	0.08763	0.08763	0.08765	0.08764	0.08764	0.08762	0.08763	0.08763	0.08763	0.08763	0.08764
		Std	0.00007	0.00004	0.00005	0.00007	0.00006	0.00006	0.00006	0.00006	0.00006	0.00007	0.00007
	PCD	Mean	0.08763	**0.08762**	0.08765	0.08765	0.08763	0.08763	0.08763	0.08763	0.08763	0.08764	0.08764
		Std	0.00006	0.00005	0.00007	0.00008	0.00006	0.00004	0.00007	0.00006	0.00007	0.00007	0.00006

Bold values denote the lowest average MSE or values whose Wilcoxon's p-value is above 0.05, i.e., values that are statistically similar

Table 3.3 Average MSE values considering Semeion Handwritten Digit dataset

Layer	Alg	Statistics	IHS	AIWPSO	BA	CS	FA	FPA	BSA	CoBiDE	DE	JADE	RANDOM
1	CD	Mean	0.19359	0.20045	0.20743	0.20529	0.20638	0.19526	0.19571	**0.19328**	0.19470	0.19893	0.19711
		Std	0.00137	0.00686	0.00449	0.00495	0.00492	0.00456	0.00365	0.00133	0.00513	0.00789	0.00313
	PCD	Mean	0.20009	0.20274	0.20763	0.20648	0.20895	0.19895	0.20003	0.19896	0.19950	0.20166	0.20362
		Std	0.00197	0.00399	0.00274	0.00356	0.00209	0.00223	0.00254	0.00148	0.00377	0.00532	0.00184
2	CD	Mean	0.20961	0.20959	0.20963	0.20966	0.20966	0.20961	0.20962	0.20963	0.20962	0.20963	0.20962
		Std	0.00037	0.00035	0.00039	0.00040	0.00041	0.00035	0.00036	0.00035	0.00037	0.00036	0.00036
	PCD	Mean	0.20962	0.20961	0.20965	0.20961	0.20966	0.20964	0.20959	0.20961	0.20960	0.20960	0.20959
		Std	0.00036	0.00039	0.00038	0.00036	0.00039	0.00036	0.00037	0.00036	0.00037	0.00036	0.00035
3	CD	Mean	0.20961	0.20964	0.20964	0.20965	0.20965	0.20961	0.20962	0.20960	0.20961	0.20964	0.20961
		Std	0.00037	0.00038	0.00038	0.00037	0.00035	0.00035	0.00037	0.00036	0.00038	0.00036	0.00037
	PCD	Mean	0.20963	0.20962	0.20966	0.20963	0.20966	0.20958	0.20963	0.20962	0.20959	0.20960	0.20961
		Std	0.00038	0.00036	0.00038	0.00036	0.00041	0.00036	0.00038	0.00041	0.00036	0.00037	0.00036

Bold values denote the lowest average MSE or values whose Wilcoxon's p-value is above 0.05, i.e., values that are statistically similar

Table 3.4 Average MSE values considering CalTech 101 Silhouettes dataset

Layer	Alg	Statistics	IHS	AIWPSO	BA	CS	FA	FPA	BSA	CoBiDE	DE	JADE	RANDOM
1	CD	Mean	**0.15554**	0.15641	0.15923	0.15923	0.16003	0.15607	0.15600	0.15638	0.15726	0.15608	0.15676
		Std	0.00211	0.00241	0.00167	0.00171	0.00156	0.00230	0.00154	0.00191	0.00172	0.00184	0.00162
	PCD	Mean	0.15731	0.15825	0.16009	0.15993	0.15957	0.15810	0.15775	0.15800	0.15726	0.15790	0.15846
		Std	0.00158	0.00231	0.00076	0.00103	0.00118	0.00192	0.00151	0.00121	0.00172	0.00135	0.00122
2	CD	Mean	0.16058	0.16056	0.16059	0.16057	0.16061	0.16062	0.16056	0.16059	0.16058	0.16058	0.16060
		Std	0.00020	0.00020	0.00020	0.00019	0.00021	0.00024	0.00020	0.00023	0.00020	0.00020	0.00020
	PCD	Mean	0.16054	0.16060	0.16058	0.16062	0.16058	0.16065	0.16057	0.16058	0.16058	0.16058	0.16062
		Std	0.00029	0.00022	0.00021	0.00023	0.00021	0.00038	0.00022	0.00022	0.00020	0.00020	0.00019
3	CD	Mean	0.16059	0.16058	0.16058	0.16060	0.16061	0.16058	0.16058	0.16060	0.16057	0.16059	0.16057
		Std	0.00022	0.00022	0.00019	0.00021	0.00023	0.00022	0.00021	0.00021	0.00021	0.00019	0.00020
	PCD	Mean	0.16059	0.16058	0.16060	0.16061	0.16058	0.16060	0.16058	0.16059	0.16057	0.16058	0.16057
		Std	0.00021	0.00021	0.00020	0.00020	0.00021	0.00024	0.00020	0.00020	0.00021	0.00021	0.00022

Bold values denote the lowest average MSE or values whose Wilcoxon's p-value is above 0.05, i.e., values that are statistically similar

Table 3.2 presents the results concerning the MNIST dataset. IHS obtained the lowest errors using the Contrastive Divergence algorithm over one single layer. BA and AIWPSO obtained statistically similar results using the PCD algorithm over two and three layers, respectively. One can notice that FPA using CD over a single layer also obtained the same average errors as the IHS, although the Wilcoxon signed-rank test does not consider both statistically similar. Moreover, the evolutionary algorithms also obtained good results, though not statistically similar as well.

Regarding Semeion Handwritten Digit dataset, Table 3.3 demonstrates the best results were obtained using CoBiDe technique over the CD algorithm with one layer. Worth pointing that none of the other methods achieved similar statistical results, which confirms the robustness of evolutionary-based meta-heuristic optimization algorithms.

Similar to MNIST dataset, the best results over CalTech 101 Silhouettes dataset was obtained using the IHS method with the CD algorithm over a single-layered DBN, as presented in Table 3.4. IHS was also the sole technique to achieve the lowest errors since none of the other methods obtained statistically similar results.

3.5.1 Training Evaluation

Figure 3.6 depicts the learning steps considering MNIST dataset. Except for the BA algorithm (and the random search), all techniques converged equally to the same point since the initial iterations. Notice FA outperformed such results, achieving the lowest error at iteration number 20. However, the training error regresses to the initial values, which suggests the problem presents a local optimum hard to be overpassed, given the set of optimized parameters.

An interesting behavior is depicted in Fig. 3.7. One can observe AIWPSO converges faster than the other techniques obtaining an average MSE of 0.2 after ten iterations. However, AIWPSO gets stuck at this time step and is outperformed by both JADE and DE after approximately 15 iterations. Moreover, DE still improves its performance until reaching its optimum at nearly 40 iterations. The behavior is not observed over the testing set, where although DE obtained good results, CoBiDE was the most accurate technique.

Regarding the Caltech 101 Silhouettes, the learning curve depicted in Fig. 3.8 showed that AIWPSO presented a similar behavior as presented over Semeion dataset, and a faster convergence in the 15 initial iterations, being outperformed by JADE afterward. Notice that IHS and FPA also demonstrated a good convergence, which is expected since IHS obtained the best results over the testing set and FPA achieved very close results. Additionally, CoBiDE and BSA are also among the best techniques together with JADE and DE, confirming the robustness of evolution techniques to the task of DBN meta-parameter fine-tuning.

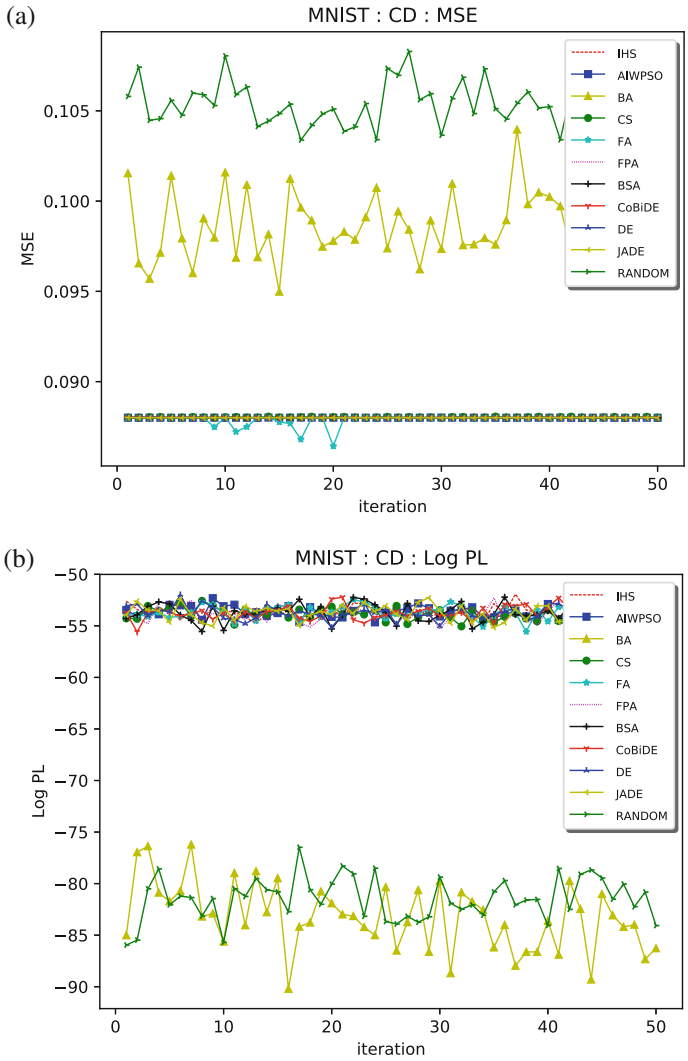

Fig. 3.6 Training convergence (**a**) MSE and (**b**) log pseudo-likelihood using the CD algorithm and a single layer of hidden units over the MNIST dataset

3.5.2 Time Analysis

Tables 3.5, 3.6, and 3.7 present the computational burden, in hours, regarding MNIST, Semeion Handwritten Digit, and Caltech 101 Silhouettes datasets, respectively. One can observe that CS is the fastest technique, followed by IHS. Such a result is expected since IHS evaluates a single solution per iteration, and CS employs a probability of evaluating or not each solution. On the other hand, the

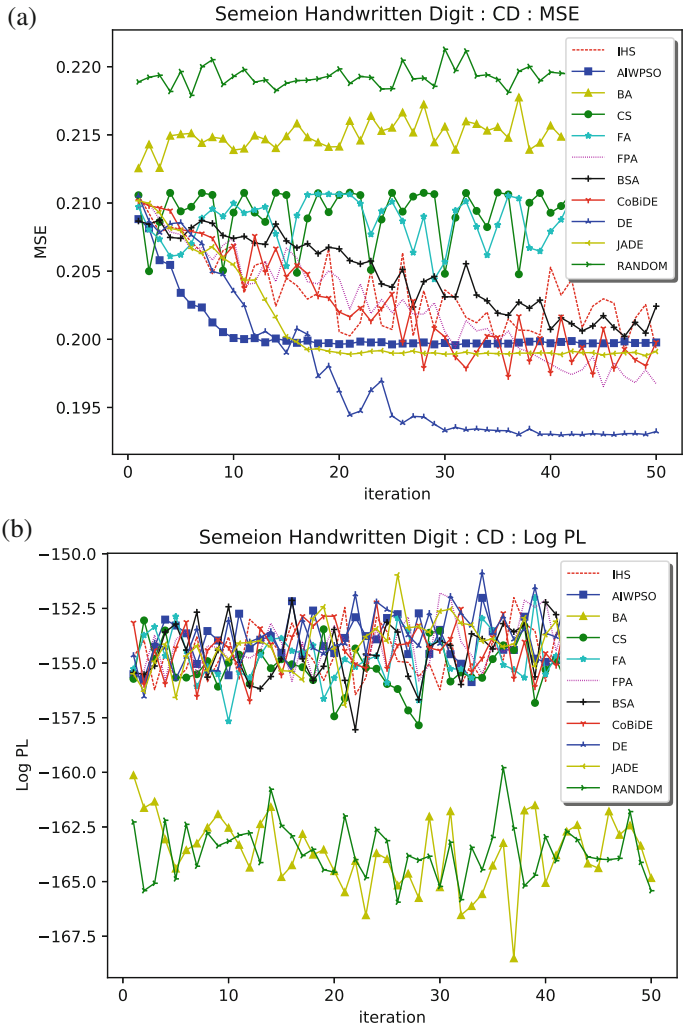

Fig. 3.7 Training convergence (**a**) MSE and (**b**) log pseudo-likelihood using the CD algorithm and a single layer of hidden units over the Semeion Hand Written Digit dataset

remaining techniques evaluate every solution for each iteration, contributing to a higher computational burden.

Additionally, evolutionary algorithms, in general, present a higher computation burden than swarm-based approaches. AIWPSO stands for an exception, offering itself as the most costly technique among all the others, due to its updating mechanism.

In most cases, the best results were obtained using a single layer as well as the CD algorithm. Such behavior is probably related to the limited number of epochs

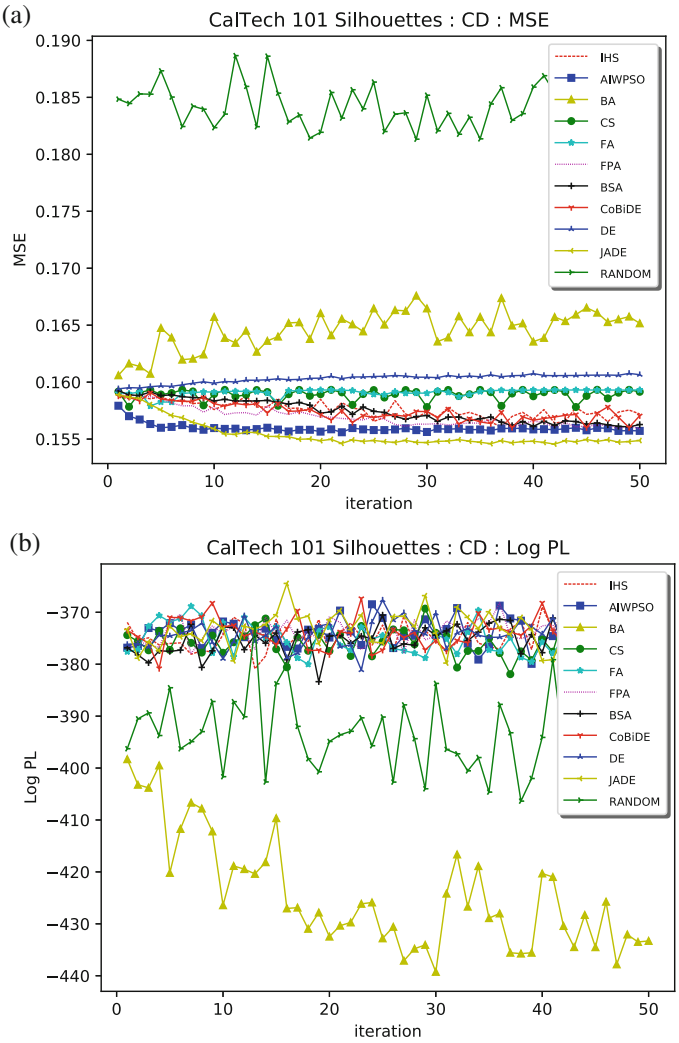

Fig. 3.8 Training convergence (**a**) MSE and (**b**) log pseudo-likelihood using the CD algorithm and a single layer of hidden units over the CalTech 101 Silhouettes dataset

employed for training, i.e., more complex models composed of a more significant amount of layers would require a higher number of epochs for convergence than the 10 epochs employed in this work. However, running the experiments over such conditions is not plausible in this context due to the massive amount of executions performed for the comparisons presented in the chapter. The same is valid for the PCD algorithm.

Table 3.5 Average computational burden (in hours) considering MNIST dataset

Layer	Alg	IHS	AIWPSO	BA	CS	FA	FPA	BSA	CoBiDE	DE	JADE
1	CD	0.18229	1.15104	0.53797	**0.15625**	0.39062	0.58773	0.58982	0.65554	0.92451	0.52083
	PCD	0.13020	1.18750	0.54166	0.23437	0.77604	0.66907	0.59709	0.67061	0.95791	0.84895
2	CD	0.31250	1.76562	0.57388	0.25520	0.71354	0.78310	0.58333	0.80579	1.09327	1.00520
	PCD	0.27083	1.39583	0.78111	**0.23437**	0.67708	0.80880	0.36979	0.83888	1.28069	0.91666
3	CD	0.28125	2.24479	1.30138	0.24479	1.16145	1.03157	0.77083	1.14248	1.40536	0.69791
	PCD	0.29166	2.46354	1.53472	**0.15104**	1.15625	0.85460	0.74479	1.09691	1.80823	0.88020

Bold values denote the lowest average MSE or values whose Wilcoxon's p-value is above 0.05, i.e., values that are statistically similar

Table 3.6 Average computational burden (in hours) considering Semeion Handwritten Digit dataset

Layer	Alg	IHS	AIWPSO	BA	CS	FA	FPA	BSA	CoBiDE	DE	JADE
1	CD	**0.08421**	0.59375	0.44277	0.13756	0.25925	0.40037	0.31366	0.41404	0.70628	0.28571
	PCD	0.10052	0.52910	0.26055	0.09523	0.39153	0.39609	0.32504	0.38522	0.52661	0.11640
2	CD	0.12169	0.85185	0.58083	0.13756	0.32804	0.30717	0.28571	0.41388	0.53883	0.13227
	PCD	**0.10416**	0.74603	0.65111	0.12698	0.51851	0.41691	0.23280	0.40367	0.45277	0.42328
3	CD	0.14814	1.13756	0.73333	0.14834	0.94382	0.44263	0.43915	0.46728	0.57517	0.19576
	PCD	0.14583	0.95238	0.80250	**0.12169**	0.60317	0.37777	0.48677	0.47605	0.69325	0.68253

Bold values denote the lowest average MSE or values whose Wilcoxon's p-value is above 0.05, i.e., values that are statistically similar

Table 3.7 Average computational burden (in hours) considering CalTech 101 Silhouettes dataset

Layer	Alg	IHS	AIWPSO	BA	CS	FA	FPA	BSA	CoBiDE	DE	JADE
1	CD	0.93181	5.03977	2.16003	0.88068	1.92045	3.47451	3.75112	3.34347	6.12850	4.68181
	PCD	0.83522	5.36363	2.37611	**0.57386**	2.99431	3.63909	3.13693	3.20528	6.12850	2.44886
2	CD	0.72727	5.34659	3.41027	**0.52840**	3.55113	2.68115	3.43181	3.12311	5.45633	4.38068
	PCD	0.77840	5.59832	2.12305	1.42613	1.85795	2.95302	3.18181	3.14890	5.45633	2.32954
3	CD	0.64204	6.34659	4.78611	0.82954	2.03409	2.73834	2.40340	3.19263	5.90374	2.59659
	PCD	**0.60227**	4.51704	3.98444	0.76704	4.59090	3.03666	2.57386	3.34851	5.90374	5.35795

Bold values denote the lowest average MSE or values whose Wilcoxon's p-value is above 0.05, i.e., values that are statistically similar

3.5.3 Hyper-Parameters Analysis

This section provides a complete list of the average values of hyper-parameters obtained during the execution of every possible experimental configuration. Notice that values in bold stand for the configuration that obtained the best results accordingly to the Wilcoxon signed-rank test.

Table 3.8 presents the average hyper-parameter values considering the MNIST dataset. The similarity between both IHS and FPA considering a single layer over the Contrastive Divergence algorithm is evident, which is expected since both obtained similar results. However, comparing these results with the ones obtained with a higher number of layers, i.e., BA with 2 layers and AIWPSO with 3 layers, over the PCD algorithm denotes a harder task, since the number of hyper-parameters is also higher, and each one exerts a degree of influence over the others.

Regarding the Semeion Handwritten Digit dataset, presented in Table 3.9, one can once again identify some relation between the set of hyper-parameters and the final results. Although IHS did not obtain the best results over the CD algorithm with a single layer, its results are pretty close to the best obtained using the CoBiDE algorithm. The resemblance is reflected in their hyper-parameter sets. Another example of this resemblance is observed in the 1-layered FPA and BSA over the CD algorithm: a close set of hyper-parameters leads to close results in the experiments.

An analogous behavior is observed in Table 3.10 regarding Caltech 101 Silhouettes dataset. Although FPA, BSA, and CoBiDE did not obtain statistically similar results to IHS according to the Wilcoxon signed-rank test, their results are very much alike, which is perceptible in their selected sets of hyper-parameter. Regarding more complex models, i.e., with two and three layers, one can still observe some likeness. Notice, for instance, the similarity between AIWPSO trained with both CD and PCD, and BSA trained with CD over three layers. However, since they require a larger number of hyper-parameters to be fine-tuned, the combination is exponentially larger, thus providing more diverse combination sets.

3.6 Conclusions and Future Works

This chapter dealt with the problem of Deep Belief Network's hyper-parameter parameter fine-tuning through meta-heuristic approaches. Experiments were conducted using three architectures, i.e., one (naïve RBM), two, and three layers, which were trained using both the Contrastive Divergence and the Persistent Contrastive Divergence algorithms. Further, the performance of ten techniques, as well as a random search, were compared over three public binary image datasets. Results demonstrated that Improved Harmony Search obtained the best results in two out of three datasets, while CoBiDE obtained the best values regarding Semeion Handwritten Digit dataset, denoting the efficiency of differential evolution-based techniques. Concerning the training steps, in general, AIWPSO converges faster

Table 3.8 Average hyper-parameter values considering MNIST dataset

Layer	Alg	L.	Statistics	IHS	AIWPSO	BA	CS	FA	FPA	BSA	CoBiDE	DE	JADE	RANDOM
1	CD	1	Hidden units	**24**	47	39	53	32	**22**	29	25	27	30	40
			Learning rate	**0.40108**	0.58222	0.39295	0.49517	0.45688	**0.45357**	0.53070	0.55669	0.49457	0.52729	0.60503
			Weight decay	**0.50896**	0.63658	0.50958	0.56223	0.55476	**0.46455**	0.62393	0.66229	0.62942	0.59766	0.64491
			Momentum	**0.00538**	0.00468	0.00438	0.00575	0.00412	**0.00496**	0.00422	0.00458	0.00517	0.00514	0.00471
	PCD	1	Hidden units	31	49	58	54	45	26	30	33	28	35	36
			Learning rate	0.58896	0.49638	0.46768	0.56154	0.60276	0.54862	0.59345	0.47164	0.59364	0.63215	0.44548
			Weight decay	0.61429	0.70234	0.68814	0.42895	0.61434	0.57970	0.68172	0.54690	0.56728	0.64276	0.57213
			Momentum	0.00532	0.00497	0.00563	0.00355	0.00570	0.00558	0.00480	0.00469	0.00526	0.00586	0.00575
2	CD	1	Hidden units	33	38	35	57	39	39	29	23	23	34	47
			Learning rate	0.63873	0.48991	0.52931	0.53659	0.51596	0.46246	0.48493	0.40542	0.53679	0.52998	0.56503
			Weight decay	0.65397	0.67948	0.60555	0.61635	0.60650	0.69145	0.72092	0.64380	0.73774	0.73380	0.67747
			Momentum	0.00518	0.00552	0.00511	0.00568	0.00414	0.00475	0.00317	0.00529	0.00542	0.00452	0.00401
		2	Hidden units	61	55	53	60	62	52	56	54	65	49	55
			Learning rate	0.55104	0.55017	0.50505	0.45789	0.59360	0.50327	0.43345	0.46479	0.44132	0.49408	0.42773
			Weight decay	0.52753	0.50562	0.45094	0.51444	0.51906	0.41652	0.51250	0.41738	0.54725	0.42865	0.48087
			Momentum	0.00503	0.00490	0.00464	0.00492	0.00414	0.00412	0.00457	0.00481	0.00597	0.00387	0.00526
	PCD	1	Hidden units	36	40	**40**	43	47	35	35	25	24	47	43
			Learning rate	0.56116	0.59384	**0.51544**	0.54896	0.60736	0.53856	0.49305	0.55222	0.69303	0.47322	0.48475
			Weight decay	0.74886	0.65189	**0.64484**	0.49600	0.61311	0.64874	0.63995	0.58538	0.72703	0.66851	0.71593
			Momentum	0.00406	0.00559	**0.00516**	0.00417	0.00500	0.00477	0.00459	0.00512	0.00406	0.00528	0.00430
		2	Hidden units	51	58	**44**	47	59	45	56	53	56	52	44
			Learning rate	0.52316	0.55521	**0.43453**	0.55172	0.55607	0.44427	0.46964	0.49459	0.49107	0.46367	0.60980
			Weight decay	0.48901	0.54542	**0.44236**	0.44567	0.53144	0.51670	0.51330	0.51885	0.52582	0.48421	0.54849
			Momentum	0.00455	0.00410	**0.00375**	0.00575	0.00500	0.00490	0.00486	0.00424	0.00611	0.00428	0.00471

(continued)

Table 3.8 (continued)

Layer	Alg	L.	Statistics	IHS	AIWPSO	BA	CS	FA	FPA	BSA	CoBiDE	DE	JADE	RANDOM
3	CD	1	Hidden units	29	40	44	51	43	30	30	30	29	38	43
			Learning rate	0.33715	0.48692	0.40539	0.39280	0.56626	0.63316	0.43006	0.57987	0.51192	0.56590	0.47168
			Weight decay	0.76890	0.64490	0.59879	0.54979	0.61017	0.59380	0.69188	0.67870	0.70871	0.70026	0.68841
			Momentum	0.00451	0.00573	0.00385	0.00512	0.00592	0.00430	0.00491	0.00464	0.00474	0.00597	0.00494
		2	Hidden units	54	64	59	65	42	51	52	55	40	57	48
			Learning rate	0.62897	0.54763	0.53962	0.50163	0.60940	0.58996	0.52708	0.52667	0.48958	0.51287	0.43878
			Weight decay	0.52169	0.47783	0.50376	0.50002	0.58921	0.59686	0.46750	0.45819	0.58481	0.56367	0.50658
			Momentum	0.00454	0.00517	0.00357	0.00493	0.00592	0.00472	0.00516	0.00479	0.00454	0.00488	0.00480
		3	Hidden units	47	44	63	54	53	65	59	51	51	56	56
			Learning rate	0.56236	0.60001	0.40769	0.56111	0.60791	0.52334	0.50628	0.46757	0.50156	0.44181	0.44104
			Weight decay	0.47333	0.43488	0.47475	0.54808	0.52987	0.57111	0.47501	0.50394	0.47855	0.44166	0.45783
			Momentum	0.00529	0.00450	0.00619	0.00404	0.00592	0.00542	0.00475	0.00483	0.00520	0.00493	0.00503
	PCD	1	Hidden units	24	**38**	46	55	47	31	25	25	28	38	25
			Learning rate	0.51945	**0.59828**	0.55540	0.48854	0.61870	0.59200	0.52847	0.52223	0.51159	0.59228	0.56341
			Weight decay	0.73241	**0.70840**	0.61023	0.63885	0.63205	0.72091	0.64644	0.69680	0.74135	0.63626	0.66017
			Momentum	0.00594	**0.00535**	0.00537	0.00604	0.00636	0.00531	0.00540	0.00621	0.00517	0.00462	0.00654
		2	Hidden units	53	**60**	43	42	47	41	54	50	60	45	52
			Learning rate	0.45762	**0.45897**	0.47631	0.46645	0.65033	0.56477	0.48949	0.42149	0.39934	0.51284	0.49131
			Weight decay	0.49819	**0.50317**	0.52903	0.50410	0.55894	0.49775	0.49806	0.50538	0.52402	0.49927	0.43492
			Momentum	0.00486	**0.00520**	0.00497	0.00436	0.00636	0.00464	0.00507	0.00548	0.00440	0.00552	0.00603
		3	Hidden units	45	**46**	52	56	47	43	65	45	67	60	50
			Learning rate	0.42481	**0.48181**	0.46640	0.50393	0.58703	0.44242	0.47473	0.48108	0.44318	0.53141	0.51465
			Weight decay	0.43727	**0.57752**	0.51211	0.52331	0.63729	0.46758	0.44450	0.51499	0.48335	0.52187	0.46255
			Momentum	0.00395	**0.00417**	0.00566	0.00533	0.00636	0.00505	0.00418	0.00417	0.00595	0.00519	0.00500

Bold values denote the lowest average MSE or values whose Wilcoxon's p-value is above 0.05, i.e., values that are statistically similar

Table 3.9 Average hyper-parameter values considering Semeion Handwritten Digit dataset

Layer	Alg	L.	Statistics	IHS	AIWPSO	BA	CS	FA	FPA	BSA	CoBiDE	DE	JADE	RANDOM
1	CD	1	Hidden units	70	36	51	50	45	60	55	**69**	73	55	55
			Learning rate	0.48848	0.42688	0.48042	0.47405	0.39179	0.41753	0.45060	**0.39458**	0.45125	0.45550	0.46821
			Weight decay	0.10000	0.30207	0.39280	0.25304	0.45168	0.13612	0.13876	**0.10128**	0.16786	0.32846	0.10972
			Momentum	0.00455	0.00357	0.00401	0.00499	0.00274	0.00455	0.00424	**0.00409**	0.00444	0.00518	0.00606
	PCD	1	Hidden units	38	42	45	49	34	53	48	49	35	43	49
			Learning rate	0.44052	0.60338	0.48166	0.48278	0.44034	0.61623	0.47053	0.42232	0.41345	0.51677	0.44187
			Weight decay	0.10010	0.18887	0.33532	0.20518	0.47422	0.10000	0.14014	0.10291	0.14620	0.29313	0.11509
			Momentum	0.00439	0.00430	0.00446	0.00449	0.00491	0.00435	0.00505	0.00517	0.00538	0.00471	0.00577
2	CD	1	Hidden units	16	24	43	33	40	23	18	27	16	27	29
			Learning rate	0.64269	0.50740	0.63046	0.48661	0.60856	0.51439	0.63148	0.60708	0.67221	0.51436	0.54418
			Weight decay	0.77736	0.73480	0.70937	0.62454	0.66910	0.71700	0.64489	0.77167	0.78755	0.76895	0.71001
			Momentum	0.00558	0.00509	0.00446	0.00395	0.00647	0.00346	0.00714	0.00491	0.00401	0.00445	0.00555
		2	Hidden units	49	62	52	41	51	51	48	50	55	45	57
			Learning rate	0.47364	0.48705	0.51975	0.56356	0.59650	0.52145	0.51645	0.46144	0.46614	0.59406	0.50009
			Weight decay	0.51644	0.45940	0.51385	0.51220	0.56529	0.47774	0.56552	0.38143	0.35235	0.54786	0.49230
			Momentum	0.00435	0.00559	0.00488	0.00487	0.00666	0.00460	0.00519	0.00420	0.00528	0.00508	0.00409
	PCD	1	Hidden units	30	38	36	46	47	46	26	24	16	28	21
			Learning rate	0.51145	0.52399	0.48580	0.61589	0.53791	0.57940	0.54080	0.58485	0.55226	0.55490	0.50456
			Weight decay	0.76035	0.73038	0.66895	0.69923	0.66610	0.76146	0.70819	0.74702	0.70493	0.77949	0.64618
			Momentum	0.00506	0.00546	0.00570	0.00565	0.00417	0.00460	0.00535	0.00354	0.00458	0.00463	0.00516
		2	Hidden units	49	59	39	50	49	44	46	59	37	55	53
			Learning rate	0.40693	0.54979	0.45286	0.51862	0.61284	0.54677	0.53121	0.51762	0.62388	0.67465	0.47355
			Weight decay	0.52690	0.49358	0.52551	0.48621	0.54843	0.46499	0.53908	0.44290	0.44684	0.52841	0.56107
			Momentum	0.00617	0.00656	0.00429	0.00598	0.00417	0.00411	0.00551	0.00576	0.00577	0.00533	0.00530

(continued)

Table 3.9 (continued)

Layer	Alg	L.	Statistics	IHS	AIWPSO	BA	CS	FA	FPA	BSA	CoBiDE	DE	JADE	RANDOM
3	CD	1	Hidden units	25	31	35	40	36	21	15	21	13	23	19
			Learning rate	0.45623	0.47945	0.44844	0.49554	0.65543	0.58816	0.46255	0.49091	0.67924	0.43849	0.47585
			Weight decay	0.83627	0.71301	0.70618	0.67923	0.72174	0.68005	0.65421	0.70858	0.77792	0.70635	0.73262
			Momentum	0.00534	0.00587	0.00421	0.00549	0.00650	0.00438	0.00502	0.00411	0.00471	0.00575	0.00480
		2	Hidden units	48	66	57	57	57	55	56	53	46	65	43
			Learning rate	0.49218	0.50469	0.51419	0.47704	0.61155	0.51277	0.44383	0.47918	0.40487	0.58144	0.49837
			Weight decay	0.56203	0.64634	0.56889	0.50804	0.61125	0.60213	0.53538	0.42734	0.46880	0.48684	0.51269
			Momentum	0.00515	0.00493	0.00434	0.00419	0.00650	0.00340	0.00399	0.00607	0.00549	0.00560	0.00537
		3	Hidden units	49	47	47	50	63	54	53	47	60	48	52
			Learning rate	0.44241	0.47279	0.46076	0.48646	0.63866	0.45881	0.44885	0.42203	0.44067	0.52481	0.51374
			Weight decay	0.58616	0.41996	0.51011	0.48673	0.65420	0.42822	0.53627	0.48018	0.52056	0.55745	0.57896
			Momentum	0.00420	0.00641	0.00507	0.00636	0.00650	0.00510	0.00398	0.00436	0.00383	0.00396	0.00475
	PCD	1	Hidden units	25	35	34	32	35	13	29	29	13	24	21
			Learning rate	0.54736	0.50874	0.44835	0.46830	0.58631	0.61603	0.63401	0.55630	0.55689	0.53534	0.52184
			Weight decay	0.78853	0.75428	0.68738	0.65386	0.70350	0.72065	0.68351	0.76314	0.76541	0.71391	0.66489
			Momentum	0.00391	0.00437	0.00521	0.00560	0.00406	0.00551	0.00545	0.00458	0.00342	0.00440	0.00514
		2	Hidden units	52	39	53	45	53	51	60	57	55	45	58
			Learning rate	0.54361	0.44707	0.51948	0.52628	0.63192	0.49611	0.45516	0.44575	0.51724	0.44793	0.52341
			Weight decay	0.41558	0.50275	0.46625	0.47226	0.59228	0.52341	0.46947	0.53055	0.38021	0.46047	0.57222
			Momentum	0.00463	0.00524	0.00428	0.00471	0.00406	0.00451	0.00453	0.00533	0.00473	0.00380	0.00560
		3	Hidden units	52	53	55	60	50	51	53	50	43	47	61
			Learning rate	0.39714	0.56683	0.58973	0.55100	0.56347	0.50745	0.46446	0.53937	0.44090	0.64079	0.55210
			Weight decay	0.49576	0.54001	0.42108	0.44537	0.63785	0.51183	0.51329	0.58296	0.49074	0.57689	0.47675
			Momentum	0.00502	0.00462	0.00547	0.00427	0.00406	0.00498	0.00474	0.00450	0.00489	0.00481	0.00305

Bold values denote the lowest average MSE or values whose Wilcoxon's p-value is above 0.05, i.e., values that are statistically similar

Table 3.10 Average hyper-parameter values considering CalTech 101 Silhouettes dataset

Layer	Alg	L.	Statistics	IHS	AIWPSO	BA	CS	FA	FPA	BSA	CoBiDE	DE	JADE	RANDOM
1	CD	1	Hidden units	**79**	63	63	62	43	80	79	77	74	85	65
			Learning rate	**0.42075**	0.50650	0.53685	0.52474	0.33131	0.50059	0.32520	0.36895	0.44642	0.44917	0.38533
			Weight decay	**0.10000**	0.14979	0.28279	0.31002	0.36294	0.16180	0.11270	0.11124	0.14447	0.10524	0.11622
			Momentum	**0.00395**	0.00575	0.00579	0.00480	0.00301	0.00448	0.00581	0.00595	0.00445	0.00615	0.00439
	PCD	1	Hidden units	70	54	52	35	49	72	63	71	74	65	62
			Learning rate	0.48994	0.51606	0.53898	0.54823	0.30069	0.43238	0.39707	0.49553	0.44642	0.46906	0.43330
			Weight decay	0.10063	0.18316	0.36682	0.27416	0.18867	0.10048	0.11245	0.10605	0.14447	0.10451	0.11558
			Momentum	0.00556	0.00484	0.00517	0.00496	0.00449	0.00445	0.00425	0.00566	0.00445	0.00411	0.00547
2	CD	1	Hidden units	52	52	38	46	53	50	39	37	32	57	59
			Learning rate	0.43484	0.42893	0.48042	0.47495	0.55443	0.47328	0.49388	0.52195	0.51051	0.53284	0.57386
			Weight decay	0.68476	0.73549	0.62374	0.64525	0.63411	0.67390	0.72068	0.68482	0.65160	0.64469	0.71296
			Momentum	0.00515	0.00535	0.00552	0.00368	0.00600	0.00390	0.00425	0.00344	0.00533	0.00444	0.00454
		2	Hidden units	47	53	50	41	59	48	43	59	67	50	50
			Learning rate	0.51389	0.47241	0.60258	0.48250	0.56618	0.39817	0.57398	0.55238	0.53212	0.50475	0.55511
			Weight decay	0.40143	0.46011	0.56419	0.51764	0.58570	0.50445	0.55294	0.42686	0.47895	0.48927	0.54928
			Momentum	0.00365	0.00453	0.00564	0.00387	0.00600	0.00450	0.00590	0.00431	0.00544	0.00540	0.00597
	PCD	1	Hidden units	45	64	37	48	51	66	38	50	32	64	38
			Learning rate	0.55227	0.51450	0.48681	0.46714	0.47946	0.37420	0.49990	0.50915	0.51051	0.52568	0.47179
			Weight decay	0.58967	0.64592	0.60823	0.54185	0.59370	0.64974	0.66568	0.67036	0.65160	0.68887	0.55182
			Momentum	0.00425	0.00459	0.00552	0.00565	0.00554	0.00601	0.00457	0.00371	0.00533	0.00363	0.00481
		2	Hidden units	64	62	54	51	39	52	45	51	67	47	54
			Learning rate	0.59710	0.63044	0.50150	0.46488	0.52302	0.53368	0.52452	0.42603	0.53212	0.53134	0.47991
			Weight decay	0.44184	0.44811	0.49725	0.53918	0.45025	0.48501	0.47066	0.51208	0.47895	0.50288	0.42359
			Momentum	0.00578	0.00584	0.00575	0.00512	0.00554	0.00428	0.00399	0.00565	0.00544	0.00572	0.00495

3	CD	1	Hidden units	43	51	51	34	45	48	31	39	59	37	50
			Learning rate	0.52239	0.51207	0.52097	0.48690	0.63751	0.52362	0.51773	0.42346	0.38359	0.47503	0.56581
			Weight decay	0.71342	0.72279	0.61374	0.62583	0.68437	0.67406	0.69570	0.68193	0.67821	0.71961	0.73120
			Momentum	0.00431	0.00514	0.00497	0.00553	0.00428	0.00600	0.00446	0.00391	0.00512	0.00546	0.00577
		2	Hidden units	49	54	52	41	43	47	49	49	61	49	44
			Learning rate	0.52924	0.51316	0.53610	0.42102	0.49939	0.52684	0.50960	0.57533	0.57925	0.52413	0.54938
			Weight decay	0.61271	0.48068	0.44030	0.44930	0.57062	0.53085	0.40449	0.46727	0.56396	0.40061	0.44172
			Momentum	0.00442	0.00495	0.00597	0.00473	0.00428	0.00505	0.00604	0.00419	0.00422	0.00506	0.00605
		3	Hidden units	55	52	55	63	46	67	50	59	57	51	55
			Learning rate	0.51658	0.48736	0.45940	0.53802	0.55106	0.53752	0.55456	0.51279	0.55418	0.55914	0.50365
			Weight decay	0.39922	0.52719	0.58714	0.45855	0.58377	0.58716	0.54719	0.51086	0.42597	0.54949	0.49739
			Momentum	0.00594	0.00399	0.00633	0.00518	0.00428	0.00457	0.00588	0.00396	0.00575	0.00387	0.00523
	PCD	1	Hidden units	56	53	49	51	49	58	51	37	59	48	49
			Learning rate	0.46399	0.40623	0.37432	0.50307	0.48364	0.59600	0.53473	0.44439	0.38359	0.43948	0.49811
			Weight decay	0.69432	0.63476	0.59465	0.51970	0.59724	0.63553	0.61455	0.60310	0.67821	0.63087	0.65155
			Momentum	0.00341	0.00550	0.00477	0.00531	0.00451	0.00472	0.00516	0.00504	0.00512	0.00408	0.00444
		2	Hidden units	56	51	51	52	59	40	52	49	61	40	51
			Learning rate	0.51054	0.42227	0.52618	0.55580	0.50243	0.42248	0.57549	0.55893	0.57925	0.56304	0.52403
			Weight decay	0.55409	0.42987	0.47076	0.46793	0.42596	0.43440	0.47748	0.49206	0.56396	0.54592	0.51342
			Momentum	0.00586	0.00452	0.00493	0.00378	0.00451	0.00623	0.00508	0.00499	0.00422	0.00492	0.00446
		3	Hidden units	59	49	52	42	42	55	50	49	57	44	50
			Learning rate	0.49215	0.53762	0.52214	0.59451	0.45669	0.49241	0.58855	0.51762	0.55418	0.48230	0.58397
			Weight decay	0.53427	0.55750	0.46411	0.45421	0.51050	0.51246	0.47173	0.51046	0.42597	0.52163	0.44847
			Momentum	0.00522	0.00467	0.00540	0.00457	0.00451	0.00454	0.00527	0.00536	0.00575	0.00434	0.00619

Bold values denote the lowest average MSE or values whose Wilcoxon's p-value is above 0.05, i.e., values that are statistically similar

than the other methods on the initial iterations. However, it is outperformed by evolution techniques after approximately 15 iterations. Finally, one can also verify that CS is the fastest technique, followed by IHS. On the other hand, AIWPSO is the slowest one.

Regarding future works, we intend to compare meta-heuristic approaches to fine-tuning DBNs to the task of classification.

Acknowledgments This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES)—Finance Code 001. The authors also appreciate FAPESP grants #2013/07375-0, #2014/12236-1, #2016/19403-6, #2017/02286-0, #2017/25908-6, #2018/21934-5 and #2019/02205-5, and CNPq grants 307066/2017-7 and 427968/2018-6.

References

1. Carreira-Perpiñán, M.A., Hinton, G.E.: On contrastive divergence learning. In: Cowell, R.G., Ghahramani, Z. (eds.) Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics. Society for Artificial Intelligence and Statistics, pp. 33–40 (2005)
2. Civicioglu, P.: Backtracking search optimization algorithm for numerical optimization problems. Appl. Math. Comput. **219**(15), 8121–8144 (2013)
3. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural Comput. **14**(8), 1771–1800 (2002)
4. Hinton, G.E.: A practical guide to training restricted Boltzmann machines. In: Montavon, G., Orr, G., Müller, K.R. (eds.) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol. 7700, pp. 599–619. Springer, Berlin (2012)
5. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006)
6. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers Inc., San Francisco (2001)
7. Kuremoto, T., Kimura, S., Kobayashi, K., Obayashi, M.: Time series forecasting using restricted Boltzmann machine. In: International Conference on Intelligent Computing, pp. 17–22. Springer, Berlin (2012)
8. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
9. Levy, E., David, O.E., Netanyahu, N.S.: Genetic algorithms and deep learning for automatic painter classification. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, pp. 1143–1150. ACM, New York (2014)
10. Liu, K., Zhang, L.M., Sun, Y.W.: Deep Boltzmann machines aided design based on genetic algorithms. In: Applied Mechanics and Materials, vol. 568, pp. 848–851. Trans Tech, Clausthal (2014)
11. Mahdavi, M., Fesanghary, M., Damangir, E.: An improved harmony search algorithm for solving optimization problems. Appl. Math. Comput. **188**(2), 1567–1579 (2007)
12. Nickabadi, A., Ebadzadeh, M.M., Safabakhsh, R.: A novel particle swarm optimization algorithm with adaptive inertia weight. Appl. Soft Comput. **11**, 3658–3670 (2011)
13. Passos, L.A., Papa, J.P.: Fine-tuning infinity restricted Boltzmann machines. In: 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), pp. 63–70. IEEE, New York (2017)
14. Passos, L.A., Papa, J.P.: On the training algorithms for restricted Boltzmann machine-based models. Ph.D. thesis, Universidade Federal de São Carlos (2018)

15. Passos, L.A., Papa, J.P.: A metaheuristic-driven approach to fine-tune deep Boltzmann machines. Appl. Soft Comput., 105717 (2019, in press). https://www.sciencedirect.com/science/article/abs/pii/S1568494619304983
16. Passos, L.A., Rodrigues, D.R., Papa, J.P.: Fine tuning deep Boltzmann machines through meta-heuristic approaches. In: 2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI), pp. 000,419–000,424. IEEE, New York (2018)
17. Passos, L.A., Rodrigues, D., Papa, J.P.: Quaternion-based backtracking search optimization algorithm. In: 2019 IEEE Congress on Evolutionary Computation. IEEE, New York (2019)
18. Passos, L.A., de Souza Jr, L.A., Mendel, R., Ebigbo, A., Probst, A., Messmann, H., Palm, C., Papa, J.P.: Barrett's esophagus analysis using infinity restricted Boltzmann machines. J. Vis. Commun. Image Represent. **59**, 475–485 (2019)
19. Pereira, C.R., Passos, L.A., Rodrigues, D., Nunes, S.A., Papa, J.P.: JADE-based feature selection for non-technical losses detection. In: VII ECCOMAS Thematic Conference on Computational Vision and Medical Image Processing: VipIMAGE 2019 (2019)
20. Rodrigues, D., Pereira, L.A.M., Almeida, T.N.S., Papa, J.P., Souza, A.N., Ramos, C.O., Yang, X.S.: BCS: a binary cuckoo search algorithm for feature selection. In: IEEE International Symposium on Circuits and Systems, pp. 465–468 (2013)
21. Rodrigues, D., de Rosa, G.H., Passos, L.A., Papa, J.P.: Adaptive improved flower pollination algorithm for global optimization. In: Nature-Inspired Computation in Data Mining and Machine Learning, pp. 1–21. Springer, Berlin (2020)
22. Rosa, G., Papa, J.P., Costa, K., Passos, L.A., Pereira, C., Yang, X.S.: Learning parameters in deep belief networks through firefly algorithm. In: IAPR Workshop on Artificial Neural Networks in Pattern Recognition, pp. 138–149. Springer, Berlin (2016)
23. Salakhutdinov, R., Hinton, G.: Deep Boltzmann machines. In: Artificial Intelligence and Statistics, pp. 448–455 (2009)
24. Smolensky, P.: Parallel distributed processing: explorations in the microstructure of cognition. In: Chap. Information Processing in Dynamical Systems: Foundations of Harmony Theory, pp. 194–281. MIT Press, Cambridge (1986)
25. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
26. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 847–855. ACM, New York (2013)
27. Tieleman, T.: Training restricted Boltzmann machines using approximations to the likelihood gradient. In: Proceedings of the 25th International Conference on Machine Learning (ICML '08), pp. 1064–1071. ACM, New York (2008)
28. Wang, Y., Li, H.X., Huang, T., Li, L.: Differential evolution based on covariance matrix learning and bimodal distribution parameter setting. Appl. Soft Comput. **18**, 232–247 (2014)
29. Whitley, D.: A genetic algorithm tutorial. Stat. Comput. **4**(2), 65–85 (1994)
30. Wilcoxon, F.: Individual comparisons by ranking methods. Biom. Bull. **1**(6), 80–83 (1945)
31. Yang, X.S.: Firefly algorithm, stochastic test functions and design optimisation. Int. J. Bio-Inspired Comput. **2**(2), 78–84 (2010)
32. Yang, X.S., Deb, S.: Cuckoo search via lévy flights. In: Nature and Biologically Inspired Computing (NaBIC 2009). World Congress on, pp. 210–214. IEEE, New York (2009)
33. Yang, X.S., Deb, S.: Engineering optimisation by cuckoo search. Int. J. Math. Model. Numer. Optim. **1**, 330–343 (2010)
34. Yang, X.S., Gandomi, A.H.: Bat algorithm: a novel approach for global engineering optimization. Eng. Comput. **29**(5), 464–483 (2012)

35. Yang, S.S., Karamanoglu, M., He, X.: Flower pollination algorithm: a novel approach for multiobjective optimization. Eng. Optim. **46**(9), 1222–1237 (2014)
36. Yosinski, J., Lipson, H.: Visually debugging restricted Boltzmann machine training with a 3D example. In: Representation Learning Workshop, 29th International Conference on Machine Learning (2012)
37. Zhang, J., Sanderson, A.C.: Jade: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**(5), 945–958 (2009)