Convolutional Neural Networks Ensembles Through Single-Iteration Optimization

Luiz Carlos Felix Ribeiro
Universidade Estadual Paulista Julio de Mesquita Filho
Gustavo Henrique de Rosa (gustavo.rosa@unesp.br)
Universidade Estadual Paulista Julio de Mesquita Filho https://orcid.org/0000-0002-6442-8343
Douglas Rodrigues
Universidade Estadual Paulista Julio de Mesquita Filho
João Paulo Papa
Universidade Estadual Paulista Julio de Mesquita Filho

Research Article

Keywords: Machine Learning, Convolutional Neural Networks, Ensembles, Optimization, Meta-Heuristics

Posted Date: December 20th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-854327/v1

Version of Record: A version of this preprint was published at Soft Computing on January 29th, 2022. See the published version at https://doi.org/10.1007/s00500-022-06791-9.

Noname manuscript No.
(will be inserted by the editor)

Convolutional Neural Networks Ensembles Through Single-Iteration Optimization

Luiz C. F. Ribeiro · Gustavo H. de Rosa* · Douglas Rodrigues · João P. Papa

Received: date / Accepted: date

Abstract Convolutional Neural Networks have been widely employed in a diverse range of computer vision-based applications, including image classification, object recognition, and object segmentation. Nevertheless, one weakness of such models concerns their hyperparameters' setting, being highly specific for each particular problem. One common approach is to employ meta-heuristic optimization algorithms to find suitable sets of hyperparameters at the expense of increasing the computational burden, being unfeasible under real-time scenarios. In this paper, we address this problem by creating Convolutional Neural Networks ensembles through Single-Iteration Optimization, a fast optimization composed of only one iteration that is no more effective than a random search. Essentially, the idea is to provide the same capability offered by long-term optimizations, however, without their computational loads. The results among four well-known datasets revealed that creating one-iteration optimized ensembles provide promising results while diminishing the time to achieve them.

Keywords Machine Learning · Convolutional Neural Networks · Ensembles · Optimization · Meta-Heuristics

The authors are grateful to CNPq grants #430274/2018-1, #304315/2017-6, #307066/2017-7, and #427968/2018-6, as well as São Paulo Research Foundation (FAPESP) grants #2013/07375-0, #2014/12236-1, #2017/25908-6, #2018/21934-5, #2019/07665-4, and #2019/02205-5

Luiz C. F. Ribeiro, Gustavo H. de Rosa*, Douglas Rodrigues, João P. Papa
Department of Computing, São Paulo State University, Bauru, Brazil
Tel.: +55-14-31036100
E-mail: {luiz.felix,,gustavo.rosa,joao.papa}@unesp.br, douglasrodrigues.dr@gmail.com

*Corresponding author

1 Introduction

Machine learning techniques proved to be thriving algorithms when applied to pattern recognition tasks [18], such as feature extraction, classification, and regression. Furthermore, the urge to create human-like models and solve computer-vision-related tasks enhanced traditional machine learning methods into more sophisticated techniques, known as deep learning [2].

Deep neural networks, for instance, the Convolutional Neural Networks (CNN) [23], achieved several hallmarks in a wide range of applications, e.g., image classification [20], object detection [4] and segmentation [11], among others. Nevertheless, these architectures are filled with problems, such as hyperparameters setting [29] and overfitting [31]. The former problem concerns that each architecture might have a specific set of hyperparameters for solving a particular task, requiring a substantial computational effort to find adequate hyperparameters. The latter problem regards that the network may overly learn the training data, resulting in poor performance when collated with new (unseen) data.

A notorious approach to construct more robust models, which is already employed in traditional machine learning techniques, are the well-known ensembles [8]. An ensemble consists of a combination of several weak learners (individual models) and then used together to produce more robust results [30]. Essentially, the idea is that each part of the ensemble is responsible for learning specific pieces of the problem and effectively solving the whole problem when combined.

Only in the past years, it has been possible to find works in the literature that use ensemble-based deep learning. Deng et al. [7] proposed ensemble-based deep learning systems to overcome speech recognition issues,

achieving significant increases in phone recognition accuracy. In 2016, Kumar et al. [21] proposed a combined approach of distinct CNN architectures applied to medical images, where they achieved higher classification rates than individual CNNs. Later on, Lee et al. [24] introduced a Long Short-Term Memory (LSTM) ensemble using distinct architectures to capture various temporal dependencies, obtaining state-of-the-art performance in skeleton-based action recognition tasks. Moreover, Ju et al. [15] compared distinct ensemble strategies over several deep neural networks, studying their impact and variations. Finally, Minetto et al. [25] proposed the Hydra framework, which uses an ensemble of CNNs to improve geospatial land classification quality.

A framework of techniques that have not yet been fully explored in ensemble-based deep learning is meta-heuristic optimization. A meta-heuristic is a combination of local searches and biological learning mechanisms used to solve a particular problem. When combined with optimization, one can construct algorithms that avoid being trapped into local optimums, and that produces feasible results like traditional optimization methods[1].

Nevertheless, meta-heuristic optimization carries a tremendous computational burden, as the objective function needs to be evaluated for almost every agent at each iteration. In order to overcome such a problem, we propose Single-Iteration Optimization (SIO), which stands for a rapid optimization that consists of only a single step and is comparable to a random search.

Therefore, this paper proposes to create Convolutional Neural Networks ensembles through Single-Iteration Optimization and compare them against meta-heuristic optimized models and their ensembles. Essentially, the idea is to train several CNNs with SIO-selected hyperparameters and combine them into an ensemble using a weighted-voting approach, where the importance (weight) of each model is framed as another optimization problem. Afterward, the obtained results are compared against a baseline CNN (default hyperparameters) and an optimized CNN, whose hyperparameters were fine-tuned by a meta-heuristic optimization. Additionally, we propose to keep some of the best models found during the optimization procedure and combine them. Such an ensemble is then compared against our proposed SIO-based ensemble.

In short, the main contributions of our work are twofold:

- Present SIO-based ensembles as a competitive and cheaper alternative to the traditional meta-heuristic-based hyperparameter optimization procedure;
- Propose to combine the best solutions found by meta-heuristics in ensembles incurring a minimal additional cost.

The remainder of this paper is organized as follows. Sections 2, 3 and 4 present some theoretical background concerning ensemble learning, Convolutional Neural Networks and meta-heuristic optimization, respectively, while Section 5 discusses the methodology employed in this work. Section 6 presents the experimental results and Section 7 states conclusions and future works.

2 Ensemble Learning

Ordinarily, ensembles are assortments of combined learners which focus on how to solve a unique problem. Their primary difference from single classifiers is the use of several combined classifiers, allowing them to accomplish more proper learning [12]. An ensemble classifier is usually composed of several weak learners, such as decision trees, support vector machines, optimum-path forests, and neural networks. Furthermore, when weak classifiers are combined, they create a unique and more robust model. It is known that the generalization ability of an ensemble is usually higher than weak learners due to the increase in the diversity of features extracted and decisions made [30].

A critical distinction between ensembles concerns their taxonomy, where they are divided into two categories: (i) homogeneous if the same weak learners compose the ensemble, and (ii) heterogeneous if different weak learners compose the ensemble. This work will use a homogeneous ensemble consisting of several CNNs with their hyperparameters randomly initialized from a predefined range. Additionally, we also use a weight-based strategy, as described in Section 2.1.

2.1 Weighted Voting-based Ensemble

Despite our present work focusing on CNNs, it is essential to highlight that such a procedure applies to any neural network-based classifier, such as traditional Multilayer Perceptron (MLP) or even Recurrent Neural Networks (RNN).

Given a collection of K classifiers, we are interested in finding a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ such that $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ stands for the dataset and $\mathcal{Y} = \{1, 2, \ldots, C\}$ denotes the set of outputs, i.e.,

[1] Traditional optimization methods rely on gradients and Hessians, which are computationally costly and susceptible to local optima.

classes. Further, let $P^{(i)} \in \mathbb{R}^{K \times C}$ denote the probabilities of a given sample $\boldsymbol{x}_i$ belonging to each of the C possible classes according to each model. Specifically, this matrix is the concatenation of the softmax outputs of each classifier. Therefore, the weighted voting-based ensemble combines all classifiers as follows:

$$\hat{P}^{(i)} = \boldsymbol{w}^T \cdot P^{(i)}, \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^K$ is the importance (weight) of each weak classifier in the ensemble. Additionally, $\hat{P}^{(i)} \in \mathbb{R}^C$ denotes the unnormalized scores of $\boldsymbol{x}_i$ belonging to each possible class according to the ensemble. Then, its predicted label $\hat{y}_i$ is computed as follows:

$$\hat{y}_i = \underset{j \in \mathcal{Y}}{\operatorname{argmax}} \, \boldsymbol{f}_j^{(i)}. \tag{2}$$

3 Convolutional Neural Networks

Hubel et al. [14] presented a seminal study regarding the primary cortex of cats, where they have identified two kinds of cells: (i) simple cells and (ii) complex cells. This research serves as the fundamental theory for the Convolutional Neural Networks architecture, where their filtering and sampling processes are analogous to simple and complex cellular mechanisms.

The first Convolutional Neural Network computational-based model was the famous "Neocognitron" [9], which used an unsupervised learning algorithm throughout the filtering phase, succeeded by a supervised algorithm as its final classifier. Furthermore, LeCun et al. [22] proposed to use the Backpropagation algorithm to provide supervised learning, fostering applications that would arise throughout the next decades.

One can perceive that CNN is a multi-layered data processing architecture. Given an input image, the CNN extracts its primordial pieces of information through high-level representations, called *multispectral images* or *feature maps*. Afterward, it concatenates them into a feature vector that can later be used by any pattern recognition technique. Figure 1 depicts one possible workflow for a Convolutional Neural Network.

A CNN can be composed of several layers, e.g., convolution, pooling, fully connected, or even a softmax activation. Regardless of its architecture, some layers are somewhat more critical than others, requiring a more in-depth explanation. The next sections describe three primary operations that characterize a CNN architecture, such as feature maps (convolution), sampling (pooling), and normalization.

3.1 Feature Maps

Let $\hat{I} = (D_I, \boldsymbol{I})$ be an input image, where $D_I \in \mathbb{R}^{l \times c}$ stands for the image domain and $\boldsymbol{I} = \{I_1, I_2, \ldots, I_c\}$ is an image composed of c channels. Additionally, let $I_i(p)$ be a pixel $p = (x_p, y_p) \in D_I$ over image I and channel i, $\forall i \leq c$. Furthermore, if $\hat{I}$ is a grey-scale image, $c = 1$ and $\hat{I} = (D_I, I)$.

Moreover, let $\gamma = (M, W)$ stand for a filter with weights $W(q)$ over every pixel $q \in M(p)$, where $M(p)$ is a mask of size $L_M \times L_M$ centered at pixel p, and $q \in M(p)$ if, and only if, $\max(|x_q - x_p|, |y_q - y_p|) \leq (L_M - 1)/2$. When dealing with multi-channel filters, their weights can be represented as vectors, such that $\boldsymbol{W}(q) = \{W_1(q), W_2(q), \ldots, W_c(q)\}$. Additionally, a multi-channel filter is defined as $\gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_c\}$, where $\gamma_i = (M, W_i), \forall i \leq c$.

Therefore, the convolution between the input image $\hat{I}$ and the filter γ_i creates the channel i from the filtered image $\hat{J} = (D_J, \boldsymbol{J})$, such that $D_J \in D_I$ and $\boldsymbol{J} = \{J_1(p), J_2(p), \ldots, J_c(p)\}$, $\forall p \in D_J$:

$$J_i(p) = \sum_{\forall q \in M(p)} I_i(q) \otimes W_i(q), \tag{3}$$

where $\otimes$ stands for the convolution operator. The weights of γ_i are regularly produced by uniform distributions, i.e., $\mathcal{U}(0, 1)$, and further normalized with zero mean and unitary norm.

3.2 Sampling

The sampling operation, commonly known as pooling, is essential for CNN, providing translational invariance to its extracted features. Let $B(p)$ stand for the pooling area of size $L_B \times L_B$, centered at pixel p. Moreover, let $D_S = D_J / s$ be the standard pooling operation for every s pixels. Thus, the pooling operation over image $\hat{J}$ creates the resulting image $\hat{S} = (D_S, \boldsymbol{S})$ and is defined as follows:

$$S_i(p) = \sqrt[\alpha]{\sum_{\forall q \in B(p)} J_i(q)^\alpha}, \tag{4}$$

where S_i and J_i stand for images S and J over channel i, respectively, $p \in D_S$ stands for every new pixel in the resulting image α stands for the stride parameter, which controls the downsampling factor of the pooling.

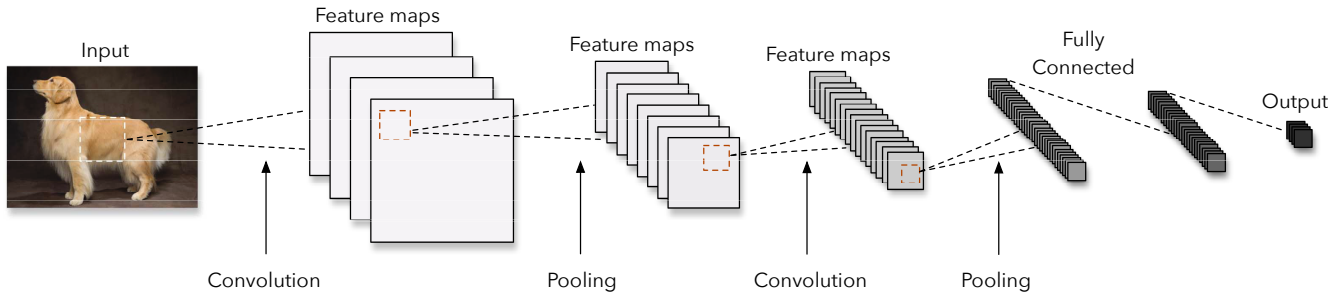


Fig. 1 Example of a Convolutional Neural Network architecture.

3.3 Normalization

Lastly, a normalization procedure can be applied in order to enhance the network's performance [6], being based on the same mechanisms found in cortical neurons [10]. Let $N(p)$ be the normalization area of size $L_N \times L_N$, centered at pixel p. The normalization operator over image $\hat{S}$ is defined as follows:

$$O_i(p) = \frac{S_i(p)}{\sum_{j=1}^{c} \sum_{\forall q \in N(p)} S_j(q) S_i(q)}. \tag{5}$$

The aforementioned operation is accomplished for every channel i and for each pixel $p \in D_O \subset D_S$ of the resulting image $\hat{O} = (D_O, \boldsymbol{O})$.

4 Meta-Heuristic Optimization

Throughout the years, the necessity of finding suitable sets of information (parameters or hyperparameters) to solve distinct tasks, fostered the study of mathematical programming, commonly known as optimization. A classic example of an optimization task is the Traveling Salesman Problem [26], which consists of traveling to distinct localities and returning to the origin using a minimum path. Furthermore, there are diverse optimization problems faced daily, such as industrial component modeling [27], operations researches [28], market economic models [17], and molecular modeling [1], among others.

An optimization problem consists in maximizing or minimizing a function through a systematic choice of possible values to the problem. In other words, the optimization procedure finds the most suitable fitness function values, given a pre-defined domain. Equation 6 describes a maximum generic optimization model without constraints.

$$\max_{\forall \boldsymbol{z} \in \mathbb{R}^{\eta}} f(\boldsymbol{z}), \tag{6}$$

where $f(\boldsymbol{z})$ stands for the fitness function, while $\boldsymbol{z} \in \mathbb{R}^{\eta}$. The optimization of this function aims at finding the most suitable set of values for $\boldsymbol{z}$, denoted as $\boldsymbol{z}^*$, where $f(\boldsymbol{z}^*) \geq f(\boldsymbol{z})$ for Equation 6. Nevertheless, when dealing with more complex fitness functions, several maxima points (local optima) arise, making it significantly more challenging searching for the optimal point. Figure 2 illustrates an example of this situation.

Traditional optimization methods [3], such as the iterative methods, e.g., Newton method, Quasi-Newton method, Gradient Descent, Interpolation methods, use the evaluation of gradients and Hessians, being unfeasible when applied to non-differentiable problems, as well as due to their high-computational burden. However, a proposition denoted as meta-heuristic has been employed to solve several optimization problems. A meta-heuristic technique consists of high-level procedures projected to generate or select a heuristic, which

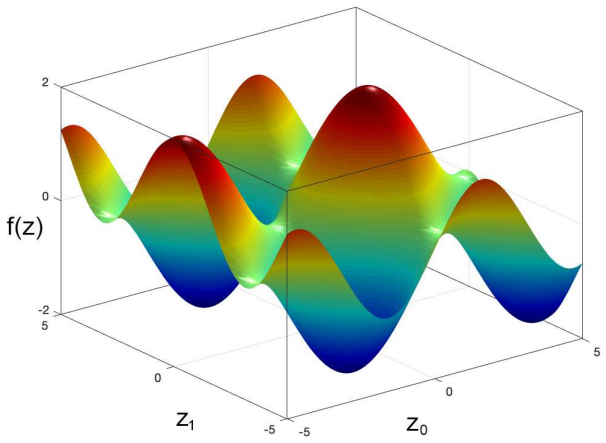


Fig. 2 Example of a mathematical function, $f(z) = sin(z_0) + cos(z_1)$, with several local optimum points.

provides a feasible solution to the optimization problem. A meta-heuristic optimization is a procedure that combines the concepts of *exploration*, used to perform searches throughout the search space, and *exploitation*, which is used to refine a promising solution based on its neighborhood.

4.1 Optimized-based Ensembles

In a nutshell, a meta-heuristic optimization technique consists in a procedure in which a given number of α agents interact for a pre-defined amount of β iterations following some algorithm (which defines the meta-heuristic behavior) to maximize (minimize) some function $f(\cdot)$, which is evaluated no more than $o(\alpha\beta)$ times.

In this work, $f(\cdot)$ represents training and computing the accuracy of a neural network, given the hyperparameters to be optimized. Additionally, the computational burden of training a neural network is considerably higher than evaluating meta-heuristic update equations. To such an extent, consider, for instance, that it is not uncommon having neural networks with thousands of parameters, whereas meta-heuristics usually possess a more restricted set of variables to be evaluated. Hence, it is reasonable to assume that the entire procedure complexity is $o(\alpha\beta T + \epsilon)$, where T denotes the complexity of training a single neural network and ϵ aggregates the meta-heuristic optimization complexity.

Instead of incurring this considerable burden to find a single best-performing model, our proposed approach suggests training α neural networks (corresponding to the number of evaluations in single iteration) with randomly selected hyperparameters and learning how to combine them via meta-heuristic optimization techniques, decreasing the procedure complexity to $o(\alpha T + \epsilon)$.

Individually, the interval for each hyperparameter to be randomly sampled from is determined[2]. Further, the $K = \alpha$ models are independently trained to share the same training and validation sets. After convergence, their outputs are combined in a weighted voting-based ensemble using another meta-heuristic optimization, as described in Section 5.2. Note that learning to combine model outputs is cheap as each model was already trained and used once to generate $P^{(i)}$ for each i sample in the dataset.

One of the main drawbacks of using ensembles is the expense of training the different models that compose it. On the other hand, performing a meta-heuristic optimization of a neural network requires training $o(\alpha\beta)$ different models in search of the best one. Consequently, such a high cost is already paid during the optimization step in a way that, instead of keeping only the top-performing model, one can keep the top K models and learn how to combine them at negligible cost.

5 Methodology

In this section, we present the proposed approach[3], as well as the employed datasets and the proposed experiments.

5.1 Datasets

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

- MNIST [23]: 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;
- K-MNIST [5]: a 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 and a test set with $10,000$ images;
- CIFAR-10 [19]: is a subset image database from the "80 million tiny images" dataset. Composed of $60,000$ 32x32 colour images in 10 classes, with $6,000$ images per class. It is divided into five training batches and one test batch, each containing $10,000$ images.

[2] Notice that this procedure is also required to perform meta-heuristic optimizations, where larger intervals may require more time for the algorithm to find suitable solutions.

[3] Our source code is available at `https://github.com/lzfelix/random_ensembles`.

– CIFAR-100 [19]: is a subset image database from the "80 million tiny images" dataset. Composed of 60,000 32x32 colour images in 100 classes, with 6,000 images per class. It is divided into five training batches and one test batch, each containing 10,000 images.

Furthermore, Figure 3 illustrates mosaics of 100 random training samples for every dataset.

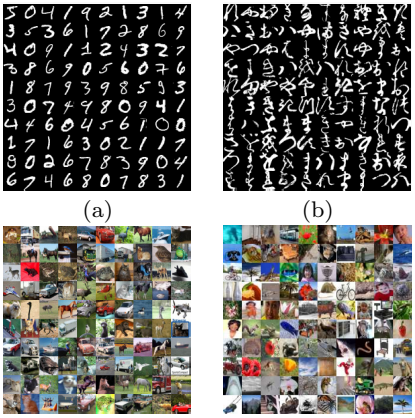


Fig. 3 Random training samples from (a) MNIST, (b) Kuzushiji-MNIST, (c) CIFAR-10 and (d) CIFAR-100 datasets.

5.2 Modeling SIO- and Optimized-based Ensembles

Recall that, according to Equation 1, weighted voting ensembles are formed by two parts: (i) the models, which were already trained, as described in Section 5.3; and (ii) their corresponding weights, which are going to be determined using meta-heuristics as well. In such a scenario, we want to find a set of weights $\boldsymbol{w}^\star$ that maximize the ensemble accuracy, which can be formulated as the following maximization problem:

$$\begin{aligned} \boldsymbol{w}^\star &= \underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left(y_i = \hat{y}_i\right), \\ \text{s.t.} &\sum_{k=1}^{k} \boldsymbol{w}_k = 1, \end{aligned} \tag{7}$$

where $\mathbb{I}(\cdot)$ is the indicator function (i.e., it returns 1 if the prediction is equal to the ground-truth label y_i and 0 otherwise), whereas N corresponds to the number of samples used to learn the models' importance. The constraint, in turn, ensures that no single model becomes much more important than the others. Moreover, as the training set has already been used to learn the models' parameters, it cannot be used to learn their importance. Hence, a validation set is employed for training the ensembles.

Regarding the meta-heuristic optimization techniques used to learn $\boldsymbol{w}^\star$, we opted to employ the Particle Swarm Optimization (PSO) [16] as it is a state-of-the-art nature-inspired algorithm and provides a proper balance between exploration and exploitation. Thus, it is suitable to fulfill the proposed approach's needs as we are dealing with a complex fitness landscape (validation accuracy of a CNN) and need an algorithm that can fully explore an n-dimensional search space-enhancing promising solutions.

5.3 Proposed Experiments

To provide a more transparent organization and a more robust experimental evaluation, we divided the experiments into three parts: weak learners, weight-based ensembles composed of the weak learners, and weight-based ensembles composed of the top-K weak learners. We have used two distinct meta-heuristic techniques to create optimized weak learners, such as Particle Swarm Optimization and Black Hole (BH) [13]. Furthermore, note that all the experiments were evaluated 15 times to provide enough data for further statistical evaluation.

5.3.1 Weak Learners

Three distinct architectures have been proposed as weak learners, as follows:

– Default (D): it stands for networks trained with their default layer configurations[4] and hyperparameters, depicted by Table 1. Such values were selected empirically and are subsequently fine-tuned by the in the Optimization-based models described next.
– SIO-based ($\mathcal{U}$): it stands for networks trained with SIO-selected hyperparameters and number of layers drawn from uniform distributions with different lower and upper bounds[5];
– SIO-based ($\mathcal{N}$): it stands for networks trained with SIO-selected hyperparameters and number of layers drawn from Gaussian distributions with $\mu = \frac{lb+ub}{2}$ and $\sigma = \mid \frac{\mu-1}{3} \mid$, where lb and ub stand for the upper and lower bounds, respectively. Except for learning rate and momentum hyperparameters, one must

[4] One can find the proposed architectures at `https://github.com/lzfelix/random_ensembles/tree/master/experiments/models`.

[5] One can find the distribution's ranges at `https://github.com/lzfelix/random_ensembles/blob/master/experiments/models/model_specs.py`.

adopt $\sigma = |\ \frac{\mu}{3}\ |$ since for these specific cases one may want values in the range $[0, 1]$;
- Optimized-based (P and B): it stands for networks trained with the most suitable hyperparameters and layers configurations found by the PSO (P) and BH (B), using the same ranges proposed by the SIO-based ($\mathcal{U}$) networks. In this architecture, we provide three distinct meta-heuristic configurations, such as 10 agents with 10 iterations (P_{10} and B_{10}), 15 agents with 10 iterations (P_{15} and B_{15}), and 20 agents with 10 iterations (P_{20} and B_{20}).

5.3.2 Weight-based Ensembles

According to Section 5.2, we proposed to create SIO- and optimized-based ensembles using pre-trained weak learners, as follows:

- SIO-based Ensemble ($E_{\mathcal{U}}$): it stands for the ensemble composed of the top-10 SIO-based networks ($\mathcal{U}$);
- SIO-based Ensemble ($E_{\mathcal{N}}$): it stands for the ensemble composed of the top-10 SIO-based networks ($\mathcal{N}$);
- Optimized-based Ensemble (E_P and E_B): it stands for the ensemble composed of the top-10 optimized-based networks (P_{10}, P_{15}, P_{20}, B_{10}, B_{15}, and B_{20}). Even though we have used PSO and BH to create the optimized weak learners, we have only employed PSO for finding the best weights when composing the ensemble.

Additionally, in an attempt to verify whether distinct ensemble creation methodologies affect our experimental results, we opted to use three distinct approaches, such as majority voting, $1/K$-weights[6], and optimized-weights. Majority voting stands for assigning a label to a sample according to the highest counting of predictions from the ensemble's networks, e.g., let the output classes of three networks be $[0, 1, 1]^T$, hence, as we have two occurrences of the class 1, the final label assignment will be 1.

On the other hand, $1/K$- and optimized-weights consider a set of weights for each network considered in the ensemble and calculates a linear combination over their predictions before the label assignment. For example, let the output predictions of three networks be $\boldsymbol{a} = [0.9, 0.1]^T, \boldsymbol{b} = [0.45, 0.55]^T$, and $\boldsymbol{c} = [0.2, 0.8]^T$, where $a, b, c \in \mathbb{R}^{\mathcal{C}}$. The $1/K$-weights approach calculates the final prediction as a weighted average between the predictions and assigns a label to the class that has the maximum probability, as follows:

$$\begin{aligned} &\underset{\mathcal{C}}{\text{argmax}}\, \boldsymbol{q} = \frac{1}{3}\boldsymbol{a} + \frac{1}{3}\boldsymbol{b} + \frac{1}{3}\boldsymbol{c} \\ &\underset{\mathcal{C}}{\text{argmax}}\, \left([0.5167, 0.4833]^T\right) = 0. \end{aligned} \tag{8}$$

Finally, the optimized-weights approach uses a meta-heuristic optimization to calculate the weights instead of using pre-defined ones.

5.3.3 Top-K Weight-based Ensembles

As an additional experiment, we also provide a thorough assessment of whether ensembles composed of K networks influence the final results. To accomplish such an observation, we have used the same ensembles proposed in Section 5.3.2 with distinct K-values, such as 5, 10, and 15, which stand for the number of top networks used to compose the referred ensembles.

6 Experiments

This section aims at presenting the experimental results concerning the SIO-based ensembles over the three datasets previously mentioned. The proposed models and ensembles have their evaluation measures computed and compared under the test set with the Wilcoxon [32] signed-rank test using $p = 0.05$. Additionally, the best results are depicted by the bolded cells.

6.1 Evaluating the Weak Learners

Table 2 presents the obtained results in terms of observed mean and standard deviation for the weak learners over the considered datasets. Initially, it is vital to highlight that the accuracy measure lies in the $[0, 1]$ interval, while the time measure stands for the number of seconds that a single network took to be trained for $\mathbf{D}$, $\mathcal{U}$, and $\mathcal{N}$ architectures. For those Optimized-based ones, the time measure stands for the number of seconds for the execution of the entire optimization process.

One can perceive that even though SIO-based ($\mathcal{U}$) weak learners could be trained with the least amount of time, they suffered due to the random initialization of parameters and severely underperformed compared to the other architectures. Furthermore, when comparing default- (D) and optimized-based (P and B) weak learners, it is possible to observe that they achieved comparable results in the MNIST dataset. In contrast, optimized ones outperformed the default in the K-MNIST dataset, and vice-versa regarding the CIFAR-10 dataset.

[6] Note that K stands for the number of networks considered in the ensemble.

Table 1 Default hyperparameters' values considering all employed datasets.

Hyperparameter	MNIST	K-MNIST	CIFAR-10	CIFAR-100
Epochs	15	15	15	15
Batch size	128	128	128	128
Training split	0.8	0.8	0.8	0.8
Optimizer	SGD	Adadelta	SGD	SGD
Learning rate	0.8	0.8	0.8	0.8
Momentum	0	0	0	0

Table 2 Experimental results concerning the weak learners.

Dataset	Measure	D	$\mathcal{U}$	$\mathcal{N}$	P_{10}	B_{10}	P_{15}	B_{15}	P_{20}	B_{20}
MNIST	Accuracy	**0.9857 ± 0.0012**	0.7616 ± 0.3317	0.8871 ± 0.2241	**0.9876 ± 0.0010**	**0.9860 ± 0.0009**	**0.9878 ± 0.0050**	**0.9858 ± 0.0010**	**0.9865 ± 0.0009**	**0.9874 ± 0.0013**
	Time (s)	580.07 ± 9.78	**314.88 ± 3.83**	480.07 ± 25.82	27,200.65	59,832.30	40,976.02	90,645.66	68,287.70	147,894.31
K-MNIST	Accuracy	0.9439 ± 0.0019	0.8207 ± 0.2779	0.7151 ± 0.3777	**0.9477 ± 0.0027**	**0.9462 ± 0.0027**	**0.9476 ± 0.0023**	**0.9449 ± 0.0019**	**0.9471 ± 0.0019**	**0.9463 ± 0.0020**
	Time (s)	2,208.52 ± 93.29	**549.68 ± 17.05**	688.45 ± 242.21	30,389.55	61,005.29	41,676.90	88,865.30	67,084.03	151,377.22
CIFAR-10	Accuracy	**0.6981 ± 0.0154**	0.4445 ± 0.1204	0.5095 ± 0.0540	0.6693 ± 0.0079	0.6811 ± 0.0048	0.6845 ± 0.0048	0.6806 ± 0.0056	0.6833 ± 0.0068	0.6669 ± 0.0080
	Time (s)	3,038.03 ± 344.89	**628.29 ± 31.24**	1,217.52 ± 336.81	18,933.42	54,646.16	28,175.38	81,311.15	148,977.81	136,357.16
CIFAR-100	Accuracy	0.3041 ± 0.0080	0.1398 ± 0.0733	0.1539 ± 0.0423	0.3027 ± 0.0074	0.2528 ± 0.0066	0.2799 ± 0.0050	0.2903 ± 0.0050	**0.3399 ± 0.0051**	0.2865 ± 0.0040
	Time (s)	**270.20 ± 18.34**	2552.97 ± 326.14	538.41 ± 19.61	442,189.74	791,886.23	865,456.10	976,462.01	1,072,287.63	1,291,929.37

One crucial point regarding optimized-based learners is the amount of time they take to be trained, where approaches with a more significant number of agents, such as P_{20} and B_{20}, have a significantly higher training time. Additionally, PSO- and BH-based optimization achieved equivalent results amongst all configurations according to Wilcoxon's signed-rank test, enabling the use of shallower search spaces and reducing their training time.

6.2 Analyzing Weight-based Ensembles

Table 3 describes the experimental results using top-10 weak learners-based ensembles over the datasets. Note that we did not provide the time of ensembles' creation as their time is not significant compared to the weak learners' training time. Additionally, as stated in Section 5.3.2, we provide three distinct types of ensembles, such as optimized, majority, and $1/K$.

One can perceive that SIO-based ensembles drastically improved SIO-based weak learners, prompting to be an alternative approach when dealing with this type of network. Additionally, as their training time is relatively short compared to other architectures, they can provide a feasible solution within a small computational burden. On the other hand, when evaluating optimized-based ensembles, it is possible to observe that their accuracy is slightly better than optimized-based weak learners and achieved the best results according to Wilcoxon's signed-rank test. However, their discrepancy, i.e., accuracy difference between ensemble and weak learners, is not as significant as the SIO-based ones, thus, not providing a performance boost compared to its amount of burden.

It is possible to observe that a suitable initialization of hyperparameters highly affects the ensembles. When comparing Tables 2 and 3; one can behold that whenever R's accuracy is closer to D, the SIO-based ensemble (E_R) achieves a comparable result to D and even outperforms it (K-MNIST). Nevertheless, as shown in the CIFAR-10 dataset, poor hyperparameters' initialization led to an inferior ensemble's performance, not achieving comparable results to any other architecture.

Finally, even though optimized-based weak learners take more time to be pre-trained, their ensemble could achieve the best results regarding all datasets, especially in CIFAR-10. One can also notice that distinct ensemble strategies or the usage of distinct meta-heuristic optimization techniques and the number of agents provide any significant performance increase, leading us to conclude that optimization is suitable when time is not an important variable.

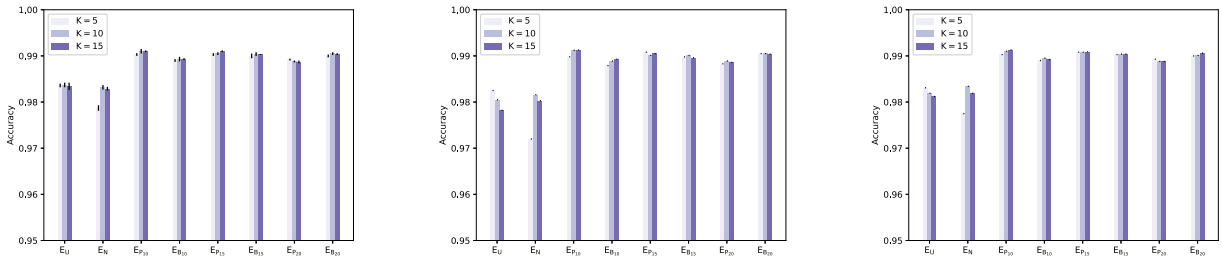
Table 3 Weight-based ensembles experimental results concerning the proposed datasets and their accuracy.

Dataset	Type	k	$E_{\mathcal{U}}$	$E_{\mathcal{N}}$	$E_{P_{10}}$	$E_{B_{10}}$	$E_{P_{15}}$	$E_{B_{15}}$	$E_{P_{20}}$	$E_{B_{20}}$
MNIST	Optimized	5	0.9836 ± 0.0004	0.9787 ± 0.0006	$\mathbf{0.9903 \pm 0.0003}$	0.9890 ± 0.0003	$\mathbf{0.9903 \pm 0.0003}$	$\mathbf{0.9900 \pm 0.0005}$	0.9892 ± 0.0002	$\mathbf{0.9900 \pm 0.0003}$
		10	0.9837 ± 0.0005	0.9832 ± 0.0005	$\mathbf{0.9910 \pm 0.0005}$	0.9893 ± 0.0005	$\mathbf{0.9905 \pm 0.0003}$	0.9904 ± 0.0004	0.9888 ± 0.0002	$\mathbf{0.9905 \pm 0.0003}$
		15	0.9834 ± 0.0008	0.9829 ± 0.0004	$\mathbf{0.9910 \pm 0.0002}$	0.9893 ± 0.0002	$\mathbf{0.9910 \pm 0.0002}$	0.9903 ± 0.0001	0.9887 ± 0.0003	0.9904 ± 0.0002
	Majority	5	0.9825 ± 0.0000	0.9720 ± 0.0000	0.9898 ± 0.0000	0.9879 ± 0.0000	$\mathbf{0.9908 \pm 0.0000}$	0.9898 ± 0.0000	0.9883 ± 0.0000	0.9905 ± 0.0000
		10	0.9805 ± 0.0000	0.9816 ± 0.0000	$\mathbf{0.9912 \pm 0.0000}$	0.9889 ± 0.0000	0.9901 ± 0.0000	0.9901 ± 0.0000	0.9889 ± 0.0000	0.9905 ± 0.0000
		15	0.9782 ± 0.0000	0.9803 ± 0.0000	$\mathbf{0.9913 \pm 0.0000}$	0.9893 ± 0.0000	0.9905 ± 0.0000	0.9896 ± 0.0000	0.9886 ± 0.0000	0.9904 ± 0.0000
	$1/K$	5	0.9831 ± 0.0000	0.9775 ± 0.0000	0.9903 ± 0.0000	0.9890 ± 0.0000	$\mathbf{0.9908 \pm 0.0000}$	0.9903 ± 0.0000	0.9893 ± 0.0000	0.9900 ± 0.0000
		10	0.9819 ± 0.0000	0.9834 ± 0.0000	$\mathbf{0.9910 \pm 0.0000}$	0.9895 ± 0.0000	0.9908 ± 0.0000	0.9904 ± 0.0000	0.9888 ± 0.0000	0.9901 ± 0.0000
		15	0.9812 ± 0.0000	0.9819 ± 0.0000	$\mathbf{0.9913 \pm 0.0000}$	0.9892 ± 0.0000	0.9909 ± 0.0000	0.9904 ± 0.0000	0.9888 ± 0.0000	0.9906 ± 0.0000
K-MNIST	Optimized	5	0.9540 ± 0.0008	0.9537 ± 0.0007	0.9572 ± 0.0003	0.9592 ± 0.0010	$\mathbf{0.9621 \pm 0.0005}$	0.9549 ± 0.0010	0.9599 ± 0.0009	0.9578 ± 0.0007
		10	0.9573 ± 0.0009	0.9570 ± 0.0007	0.9585 ± 0.0005	0.9606 ± 0.0005	$\mathbf{0.9617 \pm 0.0005}$	0.9583 ± 0.0007	$\mathbf{0.9622 \pm 0.0006}$	0.9607 ± 0.0006
		15	0.9578 ± 0.0008	0.9584 ± 0.0007	0.9590 ± 0.0006	0.9607 ± 0.0004	0.9618 ± 0.0005	0.9589 ± 0.0007	$\mathbf{0.9631 \pm 0.0006}$	0.9605 ± 0.0006
	Majority	5	0.9511 ± 0.0000	0.9505 ± 0.0000	0.9580 ± 0.0000	0.9588 ± 0.0000	$\mathbf{0.9613 \pm 0.0000}$	0.9537 ± 0.0000	0.9585 ± 0.0000	0.9571 ± 0.0000
		10	0.9556 ± 0.0000	0.9532 ± 0.0000	0.9575 ± 0.0000	0.9609 ± 0.0000	0.9606 ± 0.0000	0.9570 ± 0.0000	$\mathbf{0.9617 \pm 0.0000}$	0.9600 ± 0.0000
		15	0.9548 ± 0.0000	0.9564 ± 0.0000	0.9587 ± 0.0000	0.9591 ± 0.0000	0.9614 ± 0.0000	0.9575 ± 0.0000	$\mathbf{0.9625 \pm 0.0000}$	0.9612 ± 0.0000
	$1/K$	5	0.9539 ± 0.0000	0.9546 ± 0.0000	0.9572 ± 0.0000	0.9605 ± 0.0000	$\mathbf{0.9621 \pm 0.0000}$	0.9558 ± 0.0000	0.9610 ± 0.0000	0.9582 ± 0.0000
		10	0.9576 ± 0.0000	0.9568 ± 0.0000	0.9592 ± 0.0000	0.9611 ± 0.0000	0.9617 ± 0.0000	0.9578 ± 0.0000	$\mathbf{0.9629 \pm 0.0000}$	0.9608 ± 0.0000
		15	0.9569 ± 0.0000	0.9580 ± 0.0000	0.9594 ± 0.0000	0.9607 ± 0.0000	0.9621 ± 0.0000	0.9591 ± 0.0000	$\mathbf{0.9633 \pm 0.0000}$	0.9611 ± 0.0000
CIFAR-10	Optimized	5	0.6462 ± 0.0014	0.6343 ± 0.0041	0.7294 ± 0.0018	0.7297 ± 0.0012	$\mathbf{0.7375 \pm 0.0014}$	$\mathbf{0.7383 \pm 0.0010}$	$\mathbf{0.7370 \pm 0.0015}$	0.7295 ± 0.0023
		10	0.6427 ± 0.0034	0.6541 ± 0.0036	0.7364 ± 0.0010	0.7310 ± 0.0013	$\mathbf{0.7427 \pm 0.0017}$	0.7391 ± 0.0012	$\mathbf{0.7427 \pm 0.0012}$	$\mathbf{0.7405 \pm 0.0010}$
		15	0.6374 ± 0.0046	0.6472 ± 0.0040	0.7375 ± 0.0012	0.7324 ± 0.0012	$\mathbf{0.7467 \pm 0.0017}$	0.7405 ± 0.0010	$\mathbf{0.7477 \pm 0.0014}$	0.7401 ± 0.0013
	Majority	5	0.6245 ± 0.0000	0.5991 ± 0.0000	0.7251 ± 0.0000	0.7235 ± 0.0000	0.7272 ± 0.0000	0.7323 ± 0.0000	$\mathbf{0.7331 \pm 0.0000}$	0.7191 ± 0.0000
		10	0.6048 ± 0.0000	0.6204 ± 0.0000	0.7345 ± 0.0000	0.7294 ± 0.0000	$\mathbf{0.7393 \pm 0.0000}$	0.7368 ± 0.0000	0.7386 ± 0.0000	0.7343 ± 0.0000
		15	0.5920 ± 0.0000	0.6245 ± 0.0000	0.7379 ± 0.0000	0.7348 ± 0.0000	0.7438 ± 0.0000	0.7410 ± 0.0000	$\mathbf{0.7426 \pm 0.0000}$	0.7367 ± 0.0000
	$1/K$	5	0.6452 ± 0.0000	0.6218 ± 0.0000	0.7323 ± 0.0000	0.7315 ± 0.0000	0.7380 ± 0.0000	$\mathbf{0.7397 \pm 0.0000}$	0.7390 ± 0.0000	0.7320 ± 0.0000
		10	0.6262 ± 0.0000	0.6417 ± 0.0000	0.7374 ± 0.0000	0.7318 ± 0.0000	$\mathbf{0.7440 \pm 0.0000}$	0.7412 ± 0.0000	0.7435 ± 0.0000	0.7406 ± 0.0000
		15	0.6174 ± 0.0000	0.6343 ± 0.0000	0.7384 ± 0.0000	0.7335 ± 0.0000	0.7474 ± 0.0000	0.7411 ± 0.0000	$\mathbf{0.7481 \pm 0.0000}$	0.7415 ± 0.0000
CIFAR-100	Optimized	5	0.1829 ± 0.0028	0.2801 ± 0.0031	0.3673 ± 0.0009	0.3255 ± 0.0020	0.3357 ± 0.0020	0.3508 ± 0.0014	$\mathbf{0.4017 \pm 0.0013}$	0.3412 ± 0.0022
		10	0.2915 ± 0.0048	0.2782 ± 0.0034	0.3729 ± 0.0012	0.3320 ± 0.0020	0.3432 ± 0.0010	0.3602 ± 0.0012	$\mathbf{0.4133 \pm 0.0014}$	0.3501 ± 0.0016
		15	0.3063 ± 0.0050	0.2772 ± 0.0038	0.3784 ± 0.0011	0.3380 ± 0.0022	0.3464 ± 0.0012	0.3622 ± 0.0008	$\mathbf{0.4158 \pm 0.0013}$	0.3529 ± 0.0018
	Majority	5	0.1297 ± 0.0000	0.2355 ± 0.0000	0.3539 ± 0.0000	0.3062 ± 0.0000	0.3232 ± 0.0000	0.3319 ± 0.0000	$\mathbf{0.3872 \pm 0.0000}$	0.3283 ± 0.0000
		10	0.2219 ± 0.0000	0.2334 ± 0.0000	0.3622 ± 0.0000	0.3189 ± 0.0000	0.3335 ± 0.0000	0.3513 ± 0.0000	$\mathbf{0.4059 \pm 0.0000}$	0.3419 ± 0.0000
		15	0.2521 ± 0.0000	0.2314 ± 0.0000	0.3682 ± 0.0000	0.3228 ± 0.0000	0.3366 ± 0.0000	0.3545 ± 0.0000	$\mathbf{0.4095 \pm 0.0000}$	0.3456 ± 0.0000
	$1/K$	5	0.1826 ± 0.0000	0.2759 ± 0.0000	0.3677 ± 0.0000	0.3266 ± 0.0000	0.3381 ± 0.0000	0.3518 ± 0.0000	$\mathbf{0.4044 \pm 0.0000}$	0.3416 ± 0.0000
		10	0.2688 ± 0.0000	0.2680 ± 0.0000	0.3702 ± 0.0000	0.3322 ± 0.0000	0.3423 ± 0.0000	0.3600 ± 0.0000	$\mathbf{0.4148 \pm 0.0000}$	0.3529 ± 0.0000
		15	0.2877 ± 0.0000	0.2647 ± 0.0000	0.3767 ± 0.0000	0.3372 ± 0.0000	0.3474 ± 0.0000	0.3621 ± 0.0000	$\mathbf{0.4164 \pm 0.0000}$	0.3531 ± 0.0000

6.3 How Do Top-K Networks Affect Ensembles?

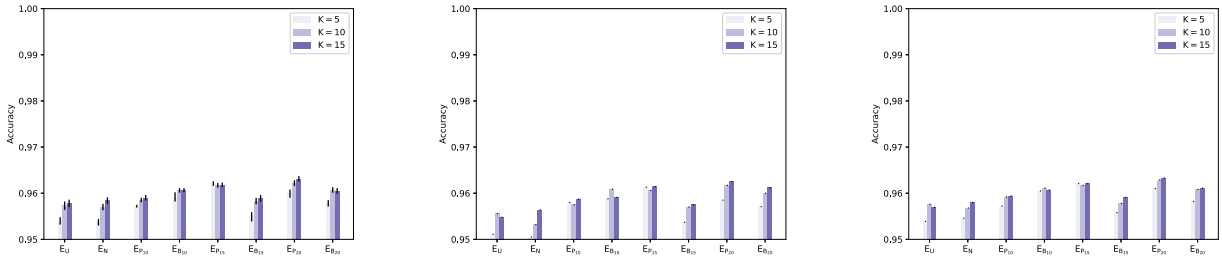
Figures 4, 5 and 6 illustrate a comparison between the usage of different K to build the ensembles regarding MNIST, K-MNIST and CIFAR-10 datasets, respectively.

Regarding the MNIST dataset, it is possible to observe a small accuracy difference between the usage of distinct K, where $K = 15$ almost obtained the best results considering all architectures. Nevertheless, one thing to perceive is that the SIO-based ensembles (E_R) obtained slightly worse results than optimized-based ones, and in two out of three occasions, the top-5 SIO-based ensemble outperformed the top-10 and top-15 SIO-based ones.



(a) (b) (c)

Fig. 4 Top-K ensembles comparison regarding MNIST dataset: (a) optimized ensembles, (b) majority-voted ensembles, and (c) $1/K$-weighted ensembles.



(a) (b) (c)

Fig. 5 Top-K ensembles comparison regarding K-MNIST dataset: (a) optimized ensembles, (b) majority-voted ensembles, and (c) $1/K$-weighted ensembles.

Concerning the K-MNIST dataset, it is vital to highlight a small difference between the usage of distinct ensemble creation methodologies. Nonetheless, top-5 SIO- and optimized-based ensembles could not achieve the best results, leaving it to top-10 and mostly top-15 ensembles. Additionally, one can perceive that majority voting and $1/K$-weights do not produce any standard deviation, as their runnings produce the same ensembles.

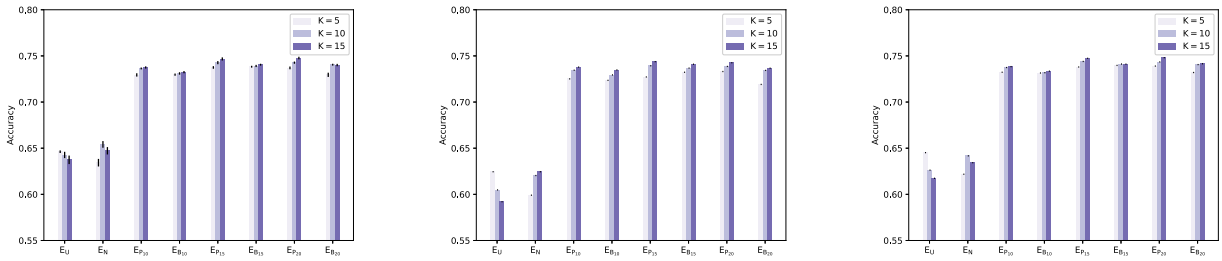
Finally, when analyzing the CIFAR-10 dataset, one can perceive in Figure 6 that due to the poor performance of SIO-based weak learners, their SIO-based ensemble achieved a highly inferior accuracy than optimized-based ensembles. Moreover, it seems that in this particular dataset, which uses a more profound architecture and a higher number of hyperparameters, the difference between top-5, top-10, and top-15 ensembles was more substantial than the other datasets.

7 Conclusion

This paper addressed creating ensembles through meta-heuristic optimization algorithms, such as Particle Swarm Optimization and Black Hole. Essentially, the overall idea is to pre-train a set of weak learners by using random uniform hyperparameters (random-based weak learners) and by finding the most suitable hyperparameters through meta-heuristics (optimized-based weak learners). Furthermore, with the pre-trained networks, we opted to construct ensembles composed of the top-K networks, i.e., the networks that achieved the best accuracy over the validation sets and evaluated their performance over the testing sets.

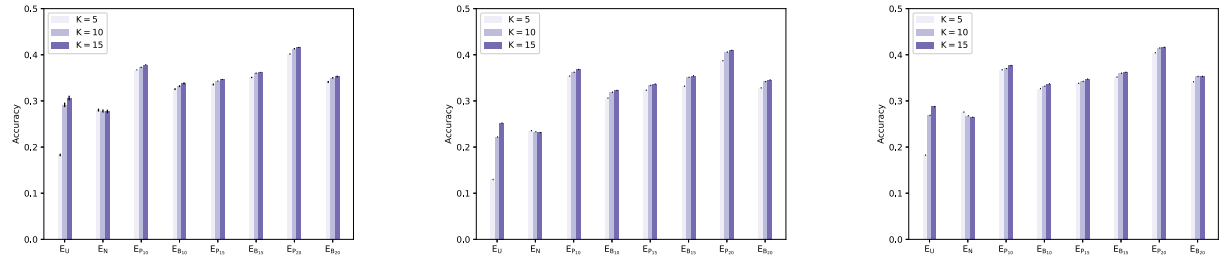
The experimental setup was conducted over three image classification literature datasets, such as MNIST, K-MNIST, and CIFAR-10. Additionally, we provided a robust comparison between distinct ensemble creation methodologies, e.g., majority voting, $1/K$-weights, and optimized-weights, as well as, we assessed the influence of using the top-K networks to compose the ensembles, with $K = 5$, $K = 10$, and $K = 15$.

Experimental results reported that it is possible to create comparable random-based ensembles and even outperform default architectures (K-MNIST) when their weak learners' hyperparameter initialization is sufficiently proper. Additionally, they provide a feasible alternative when time is a constraint to be taken into account. Nonetheless, when comparing their results with optimized-based ensembles, it is clear that the meta-heuristic optimization plays an essential role in finding the most suitable hyperparameters and creat-



(a) (b) (c)

Fig. 6 Top-K ensembles comparison regarding CIFAR-10 dataset: (a) optimized ensembles, (b) majority-voted ensembles, and (c) $1/K$-weighted ensembles.



(a) (b) (c)

Fig. 7 Top-K ensembles comparison regarding CIFAR-100 dataset: (a) optimized ensembles, (b) majority-voted ensembles, and (c) $1/K$-weighted ensembles.

ing adequate weak learners, thus composing ensembles that achieved the best results amongst all comparisons.

Regarding future works, we aim to extend the proposed approach to distinct neural networks, such as Recurrent Neural Networks and Restricted Boltzmann Machines and apply it to different tasks, such as text classification, image reconstruction, and image denoising. Furthermore, we aim at exploring whether one-shot optimizations, i.e., extremely fast optimizations, might bring an improvement over the random initialization of hyperparameters or not. Such an approach may take advantage of both random- and optimized-based learning, where we expect that it will deliver feasible results within a small amount of time.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Barone, V., Cossi, M., Tomasi, J.: Geometry optimization of molecular structures in solution by the polarizable continuum model. Journal of Computational Chemistry **19**(4), 404–417 (1998)
2. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(8), 1798–1828 (2013)
3. Bertsekas, D.P.: Nonlinear programming. Athena Scientific (1999)
4. Cai, Z., Fan, Q., Feris, R.S., Vasconcelos, N.: A unified multi-scale deep convolutional neural network for fast object detection. In: european conference on computer vision, pp. 354–370. Springer (2016)
5. Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., Ha, D.: Deep learning for classical japanese literature (2018)
6. Cox, D., Pinto, N.: Beyond simple features: A large-scale feature search approach to unconstrained face recognition. In: Proceedings of the IEEE International Conference on Automatic Face Gesture Recognition and Workshops, pp. 8–15 (2011)
7. Deng, L., Platt, J.C.: Ensemble deep learning for speech recognition. In: Fifteenth Annual Conference of the International Speech Communication Association (2014)
8. Dietterich, T.G.: Ensemble methods in machine learning. In: International workshop on multiple classifier systems, pp. 1–15. Springer (2000)
9. Fukushima, K., Miyake, S.: Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. Pattern Recognition **15**(6), 455–469 (1982)
10. Geisler, W.S., Albrecht, D.G.: Cortical neurons: isolation of contrast gain control. Vision Research **32**(8), 1409–1410 (1992)
11. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and seman-

tic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580–587 (2014)
12. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence **12**(10), 993–1001 (1990). DOI 10.1109/34.58871
13. Hatamlou, A.: Black hole: A new heuristic optimization approach for data clustering. Information sciences **222**, 175–184 (2013)
14. Hubel, D.H., Wiesel, T.N.: Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. The Journal of physiology **160**(1), 106–154 (1962)
15. Ju, C., Bibaut, A., Laan, M.: The relative performance of ensemble methods with deep convolutional neural networks for image classification. Journal of Applied Statistics **45**(15), 2800–2818 (2018). DOI 10.1080/02664763.2018.1441383
16. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers Inc., San Francisco, USA (2001)
17. Konno, H., Yamazaki, H.: Mean-absolute deviation portfolio optimization model and its applications to tokyo stock market. Management Science **37**(5), 519–531 (1991)
18. Kotsiantis, S.B.: Supervised machine learning: A review of classification techniques. In: Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies, pp. 3–24. IOS Press, Amsterdam, The Netherlands, The Netherlands (2007)
19. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
20. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)
21. Kumar, A., Kim, J., Lyndon, D., Fulham, M., Feng, D.: An ensemble of fine-tuned convolutional neural networks for medical image classification. IEEE Journal of Biomedical and Health Informatics **21**(1), 31–40 (2017). DOI 10.1109/JBHI.2016.2635663
22. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Computation **1**(4), 541–551 (1989)
23. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
24. Lee, I., Kim, D., Kang, S., Lee, S.: Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks. In: The IEEE International Conference on Computer Vision (ICCV) (2017)
25. Minetto, R., Segundo, M.P., Sarkar, S.: Hydra: an ensemble of convolutional neural networks for geospatial land classification. IEEE Transactions on Geoscience and Remote Sensing (2019)
26. Papadimitriou, C.H.: The euclidean travelling salesman problem is np-complete. Theoretical Computer Science **4**(3), 237–244 (1977)
27. Rao, S.S., Rao, S.S.: Engineering optimization: theory and practice. John Wiley & Sons (2009)
28. Rardin, L.R.: Optimization in operations research, vol. 166. Prentice Hall (1998)
29. de Rosa, G.H., Papa, J.P., Marana, N.A., Scheirer, W., Cox, D.D.: Fine-tuning convolutional neural networks using harmony search. In: Iberoamerican Congress on Pattern Recognition, pp. 683–690. Springer (2015)
30. Schapire, R.E.: The strength of weak learnability. Mach. Learn. **5**(2), 197–227 (1990). DOI 10.1023/A:1022648800760
31. Smirnov, E.A., Timoshenko, D.M., Andrianov, S.N.: Comparison of regularization methods for imagenet classification with deep convolutional neural networks. AASRI Procedia **6**, 89 – 94 (2014). DOI https://doi.org/10.1016/j.aasri.2014.05.013. 2nd AASRI Conference on Computational Intelligence and Bioinformatics
32. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin **1**(6), 80–83 (1945)