ORIGINAL ARTICLE

QUALITY

Hyperparameter Optimization in Traditional Multilayer Perceptron Networks through Exploratory Search of Variable Codes

Optimización de hiperparámetros en redes perceptrón multicapa tradicionales mediante búsqueda exploratoria de códigos variables

Javier Carballo Pérez [1,*] https://orcid.org/0009-0000-0908-136X

José Arzola Ruiz [2] https://orcid.org/0000-0002-4101-878X

[1] Faculty of Industrial Engineering, Technological University of Havana «José Antonio Echeverría» (CUJAE), Havana, Cuba

[2] Faculty of Mechanical Engineering, Technological University of Havana «José Antonio Echeverría» (CUJAE), Havana, Cuba

*Autor para correspondencia: carballoperezjavier0@gmail.com

ABSTRACT

The problem of hyperparameter optimization in multilayer perceptron neural networks for non-linear continuous regression tasks was addressed. A heuristic method termed Exploratory Search at the Extreme of a Variable Code Function, derived from the Variable Integration Method, was proposed. The optimization problem was formulated using a Chebyshev distance scalarization that simultaneously balances predictive accuracy (R^2) and generalization across training, validation, and test sets. The method was

evaluated on five UCI regression datasets. Comparisons against Random Search and Bayesian Optimization, over twenty independent runs per method, showed a statistically significant improvement in the coefficient of determination (R^2) versus Bayesian Optimization ($p = 0.047$), with no significant difference versus Random Search, and low variability across runs.

Keywords: artificial neural networks; multi-objective optimization; heuristics; non-linear regression; hyperparameter optimization.

RESUMEN

Se abordó el problema de optimización de hiperparámetros en redes perceptrón multicapa para tareas de regresión continua no lineal. Se propuso el método heurístico denominado Búsqueda Exploratoria en el Extremo de una Función de Códigos Variables derivado del Método de Integración de Variables. El problema de optimización se formuló mediante una escalarización de distancia Chebyshev que equilibra simultáneamente la exactitud predictiva (R^2) y la generalización en los conjuntos de entrenamiento, validación y prueba. El método se evaluó en cinco conjuntos de datos de regresión del Repositorio UCI: Boston Housing, Friedman #1, Calidad del Aire, Resistencia a la Compresión del Hormigón y Eficiencia Energética. Las comparaciones con Búsqueda Aleatoria y Optimización Bayesiana, sobre veinte ejecuciones independientes por método, mostraron una mejora estadísticamente significativa en el coeficiente de determinación (R^2) frente a la Optimización Bayesiana ($p = 0,047$), sin diferencias significativas frente a la Búsqueda Aleatoria, con baja variabilidad entre ejecuciones.

Palabras clave: *redes neuronales artificiales; optimización multiobjetivo; heurísticas; regresión no lineal; optimización de hiperparámetros.*

Received: 23/07/2026

Approved: 23/07/2026

Introduction

Artificial neural networks are fundamental tools for modeling complex non-linear relationships in engineering, economics, and environmental sciences [1]. Among the various architectures, feedforward networks trained by backpropagation are the most widely used because of their theoretical soundness and practical effectiveness [2]. However, the performance of these networks depends on the appropriate selection of their hyperparameters: number of hidden layers, number of neurons per layer, transfer functions, and training algorithms [3, 25].

The hyperparameter search space in these networks grows combinatorially. Considering configurations with up to four hidden layers, twenty neurons per layer, nine transfer functions, and twelve training algorithms, the total number of possible architectures exceeds 8×10^{16} combinations combinations. This combinatorial explosion makes exhaustive search infeasible and demands efficient heuristic exploration strategies.

Table 1 summarizes the main differences among the approaches reported in the literature for hyperparameter optimization in neural networks, in terms of search strategy, number of evaluations required, capacity to escape local optima, and applicability to high-dimensional search spaces. While evolutionary algorithms [4, 5, 6], bio-inspired algorithms [8], and particle swarm approaches [7] explore the space through populations of solutions, their computational cost grows with population size and number of generations. Bayesian Optimization [9, 10] builds a probabilistic model of the fitness landscape, but can converge prematurely in multimodal spaces. The

proposed approach, BAEFCV, combines random sampling within progressively refined intervals with a reopening mechanism that preserves population diversity, offering an adaptive balance between global exploration and local exploitation.

Table 1 - Methodological comparison of approaches for hyperparameter optimization in neural networks

Aspect	**Evolutionary Algorithms (GA, DE)**	**Particle Swarm Optimization (PSO)**	**Bayesian Optimization (TPE)**	**BAEFCV (Proposed)**
Search strategy	Population-based, with crossover and mutation operators	Particle movement with velocity and position	Probabilistic surrogate model (Gaussian Processes or TPE) + acquisition function	Random sampling within intervals + subinterval elimination
Exploration-exploitation balance	Controlled by mutation rate and selective pressure	Controlled by inertia coefficient and cognitive/social factors	Controlled by acquisition function (EI, UCB, PI)	Controlled by progressive interval narrowing + periodic reopening
Typical no. of evaluations (for this problem)	> 1000 (e.g., population 50 × 20 generations)	> 1000	≈ 500	≈ 300
Handling of multimodality	Yes (population diversity)	Yes (swarm diversity)	Partial (exploration term in acquisition)	Yes (interval reopening on stagnation)
Additional computational overhead	Medium (crossover/selection operations)	Low (velocity updates)	High (surrogate model fitting at every iteration)	Very low (only comparisons and bound updates)

Scalability to high-dimensional spaces	Medium	Medium	Low (surrogate model scales poorly with >10 variables)	High (coding compresses dimensionality)
Representative references	[4, 5, 6, 11]	[7]	[9, 10]	(Present work)

The objective of the present work is to develop and validate a heuristic method for hyperparameter optimization in multilayer perceptron networks applied to non-linear regression problems, balancing computational efficiency and predictive accuracy. The main contribution consists of the application of the BAEFCV algorithm—derived from the Variable Integration Method [12, 13, 14, 22]—to this domain, together with a multi-objective formulation based on Chebyshev distance that simultaneously controls performance on the training, validation, and test sets. The scope of the study is limited to single-neural-network architectures with up to four hidden layers, evaluated on five benchmark datasets from the repository.

Hyperparameter optimization in neural networks has direct applications in problems characteristic of Engineering in general, and of Industrial Engineering in particular, such as predicting the strength of composite materials from their constituent proportions [23, 24]; estimating building energy demand as a function of architectural and climatic parameters [17]; modeling urban air quality for the planning of mitigation measures [15]; and forecasting the mechanical properties of concrete as a function of its composition for quality control in production [16]; and designing gating systems in casting through the coupling of FEM and neural networks [27]. In all these cases, the appropriate selection of the neural network architecture determines the accuracy of the predictions and, therefore, the quality of the design, operation, or planning decisions derived from the models.

Materials and methods

Variable Integration Method

The Variable Integration Method (VIM) is a general metaheuristic framework for solving complex combinatorial optimization problems [17, 18, 19, 20, 21]. Its foundation lies in the evolution of coded solution representations through a set of operators that progressively improve populations of candidates across successive generations.

Unlike classical evolutionary algorithms, VIM does not operate directly on the problem variables, but on numerical codes that integrate the information of multiple variables into a compact representation. This coding reduces the effective dimensionality of the search space and facilitates efficient exploration of promising regions.

The general VIM scheme, illustrated in figure 1, comprises the following components:

- A system for coding solutions into variable codes.
- A method for generating initial populations of solutions.
- A quality or fitness function that evaluates each candidate solution.
- Operators for modifying code compositions between generations.
- Stopping criteria based on convergence or number of iterations.
- Parameters that control population size and operator behavior.

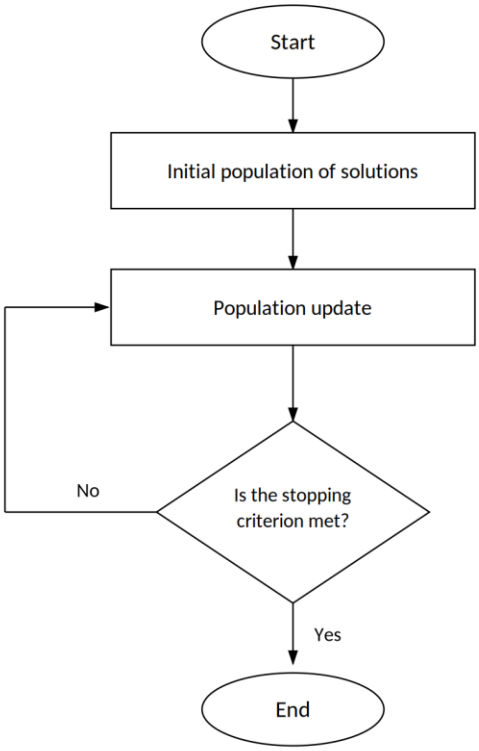

Fig. 1 - General scheme of the Variable Integration Method

Although metaheuristic methods have gained wide acceptance in the optimization literature, their application can present characteristic difficulties: loss of population diversity, premature convergence to low-quality solutions, or excessively slow search processes. Within the VIM framework, algorithms have been developed specifically to mitigate these difficulties, in particular, through the active maintenance of diversity in populations of solutions close to efficient regions of the search space. This distinguishes it from approaches such as standard genetic algorithms or Bayesian optimization, which can exhibit premature convergence in high-dimensional search spaces.

Exploratory search algorithm at the extreme of a variable code function (BAEFCV)

As a particular implementation of VIM, this work employs the Exploratory Search at the Extreme of a Variable Code Function algorithm (BAEFCV) [18-

21]. The algorithm performs random partitions of the search space associated with each variable code and, for each combination of internal partition points, generates candidate solutions that are subsequently decoded to obtain the system configuration being evaluated. The overall process is illustrated in figure 2.

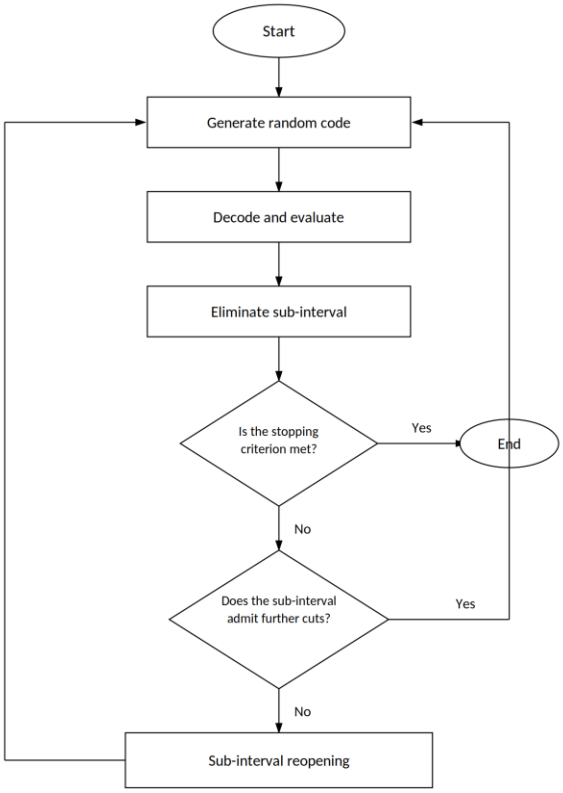

Fig. 2 - Illustration of the BAEFCV algorithm

Coding scheme

The possible solutions to the problem are represented by variable codes, where each code value corresponds to a specific combination of decision variables. The initial search interval for each variable code *j* is computed as:

$$Inter_j = \prod_i Opc_{ij} - 1 \qquad (1)$$

where Opc_{ij} *represents the number of possible solutions for variable* x_i *encoded by variable code* j. This representation transforms the combinatorial optimization problem into a process of exploring the code space, whose cardinality is significantly smaller than that of the original configuration space.

The decoding of a code Cod_j into a system configuration is performed through a bijective function that maps the integer code value to the corresponding combination of decision-variable values. Since the code space is one-dimensional for each encoded variable, the sampling and sub-interval elimination operations are computationally efficient even for high-cardinality search spaces.

Search procedure

The BAEFCV algorithm operates through iterative exploration and progressive narrowing of the search interval in the code space. At each iteration, the algorithm alternates between two complementary phases: an exploration phase through random sampling and an exploitation phase through elimination of unpromising sub-intervals. The procedure is described below:

1. Initialization: For each variable code j, the initial search bounds [0, $Inter_j$] are defined, where $Inter_j$ is computed according to equation (1).

2. Random sampling (exploration): k random codes Cod_1, Cod_2, …, Cod_k are generated within the current search bounds.

3. Decoding: Each generated code is decoded to obtain the corresponding candidate system configuration.

4. Evaluation (exploitation): The fitness function (Chebyshev distance to the desired performance targets) is computed for all combinations of codes evaluated.

5. Sub-interval elimination: The sub-interval that does not contain components of the best solution identified among the candidates evaluated in the current iteration is discarded.

6. Interval reopening: When the search interval for a variable is reduced excessively (width < 2), the initial bounds are restored to maintain population diversity and avoid premature convergence.

7. Termination: The algorithm stops when a predefined number of consecutive generations without improvement in the fitness function is reached (stagnation criterion).

The interval-reopening mechanism (step 6) is a distinctive feature of BAEFCV compared with other interval-based search methods. When the search space for a given code is detected to have contracted below a minimum threshold, the algorithm resets the bounds of the affected code while preserving the best solution found so far. This mechanism prevents premature convergence and allows the algorithm to explore regions of the space that might otherwise have been prematurely discarded, particularly in multimodal fitness landscapes.

The combination of random sampling within progressively refined intervals with the reopening mechanism gives BAEFCV an adaptive balance between global exploration and local exploitation, a feature that is essential for the efficient optimization of hyperparameters in MLP networks, where the search space can exceed 8×10^{16} possible configurations.

Algorithm pseudocode

The complete pseudocode of the BAEFCV algorithm for two codes per variable is presented in Algorithm 1. The algorithm receives as inputs the set of decision variables X with their option counts, the fitness function F(θ), the maximum number of iterations without improvement (max_stagnation), and the maximum total number of iterations (max_iter).

Algorithm 1: BAEFCV (Exploratory Search of a Variable Code Function, 2 codes per variable)

Input:

Decision variables $X = \{x_1, \ldots, x_n\}$, option counts Opc_{ij},

fitness function F(θ), max_stagnation, max_iter

Output: Optimal configuration θ*

Initialization:

For each variable code j:

$Inter_j \leftarrow \prod_i (Opc_{ij} - 1)$

$lower_j \leftarrow 0$, $upper_j \leftarrow Inter_j$

best_fitness ← ∞, stagnation_counter ← 0

For iteration = 1 to max_iter:

For each variable code j:

$Cod_1 \leftarrow random(lower_j, upper_j)$

$Cod_2 \leftarrow random(lower_j, upper_j)$

$\theta_1 \leftarrow Decode(Cod_1)$, $\theta_2 \leftarrow Decode(Cod_2)$

$f_1 \leftarrow F(\theta_1)$, $f_2 \leftarrow F(\theta_2)$

Update best if improved:

If f_1 < best_fitness: best_fitness ← f_1, $\theta^* \leftarrow \theta_1$

If f_2 < best_fitness: best_fitness ← f_2, $\theta^* \leftarrow \theta_2$

Sub-interval elimination:

If $f_1 > f_2$: $lower_j \leftarrow Cod_1$ else: $upper_j \leftarrow Cod_2$

If $(upper_j - lower_j) < 2$: restore initial bounds

If there was improvement: stagnation_counter ← 0

else: stagnation_counter ← stagnation_counter + 1

If stagnation_counter > max_stagnation: stop

Return θ*

Complexity analysis and algorithm properties

The per-iteration computational complexity of BAEFCV is $O(J \cdot k \cdot C)$, where J is the number of variable codes, k is the number of random samples generated per code (k = 2 in the base implementation), and C is the cost of evaluating the fitness function F(θ). For the hyperparameter optimization problem in MLP networks, C corresponds to the training and evaluation of a neural network, which is the dominant step in computational cost.

The total number of objective-function evaluations until convergence is bounded by max_iter × J × k, which makes BAEFCV substantially more efficient than exhaustive search. In the reported experiments, max_iter = 150 and max_stagnation = 50 were used, resulting in a maximum of 300 network evaluations per run, compared with 2,400 for random search and 500 for Bayesian optimization.

The sub-interval elimination mechanism guarantees that, under ideal conditions (unimodal function), the algorithm converges to the optimum with increasing probability as the interval narrows. The periodic reopening of intervals introduces a global exploration component that allows the algorithm to escape local optima, which makes BAEFCV particularly well suited to non-convex and multimodal fitness landscapes such as those that characterize the optimization of neural network architectures.

Formulation of the optimization model

The decision variables are: number of hidden layers, number of nodes in each layer, transfer function of each layer, and training function. The performance indicators are the standard errors (S_1, S_2, S_3) and the coefficients of determination (R_1, R_2, R_3) for training, validation, and test, respectively.

The objective function is based on the Chebyshev distance:

$$Z = \min\left[\max \frac{|\ R_j - R_j^d\ |}{R_j^d}\right] \qquad (2)$$

This formulation minimizes the distances between the computed and the desired values of the indicators, avoiding both overfitting and underfitting. The desirable R^2 values were determined from ANOVA analysis as lower bounds, given the linear nature of the variance test [26]. The transfer and training functions employed are presented in tables 2 and 3.

Table 2 - Transfer functions employed in the model

Function	Expression
Purelin	f(x) = x
Tansig	f(x) = tanh(x)
Logsig	f(x) = 1/(1+e^(-x))
poslin	f(x) = max(0, x)

Table 3 - Training functions employed in the model

Function	Algorithm
trainlm	Levenberg-Marquardt
traingd	Gradient descent
trainrp	Resilient backpropagation
trainscg	Scaled conjugate gradient

traingdx	Gradient descent with variable learning rate

Experimental setup

Five datasets from the UCI Machine Learning Repository were used for continuous regression tasks: Boston Housing (13 socioeconomic predictors, output: median home value), Friedman #1 (synthetic problem: $Y = 10 \cdot \sin(\pi \cdot X_1 \cdot X_2) + 20 \cdot (X_3 - 0.5)^2 + 10 \cdot X_4 + 5 \cdot X_5 + \varepsilon$), Air Quality (hourly measurements of atmospheric pollutants; outputs: CO and NMHC), Concrete Compressive Strength (eight composition variables; output: strength in MPa), and Energy Efficiency (eight architectural parameters; outputs: heating load Y1 and cooling load Y2) [15, 16, 17, 18, 19]. All input data were normalized to zero mean and unit variance. For the multi-output datasets, one network was trained per output variable. The split was 70% training, 15% validation, and 15% test.

The algorithm parameters were: maximum hidden layers = 4, maximum nodes per layer = 35, maximum iterations without improvement = 50, maximum total iterations = 150. Random Search and Bayesian Optimization were used as comparison methods. The non-parametric Wilcoxon signed-rank test, suitable for small samples without a normality assumption, was used for statistical analysis. To strengthen the statistical evidence, the number of independent runs was increased to twenty for each method and dataset, using the same twenty consecutive random seeds for the three methods in each run. The results reported in Table 4 correspond to the mean and standard deviation over these twenty runs.

All experiments were run on a machine with an Intel Core i7-12700H processor (2.3 GHz, 14 cores), 32 GB of RAM, and Windows 11 Pro. The BAEFCV algorithm and the network training were implemented in MATLAB.

To ensure a fair comparison among methods, three experimental scenarios were carried out: Scenario A (equal budget), with all three methods limited to 300 objective-function evaluations; Scenario B (original budget); and Scenario C (maximum time), with all three methods limited to 30 minutes of execution. The results presented in Table 4 correspond to Scenario C, while Scenario A is reported in the supplementary material. Scenario C equalizes the time budget across methods, so per-dataset computation times are not reported separately in Table 4; temporal comparability is guaranteed by the experimental design in this scenario.

Results

Comparative performance

Table 4 presents the comparative results on the seven regression problems (twenty independent runs per method).

Table 4 - Comparative results of BAEFCV, Random Search, and Bayesian Optimization (TPE) on seven regression tasks. Values expressed as mean ± standard deviation over twenty independent seeds. MSE: mean squared error; RMSE: root mean squared error

Dataset	**Metric**	**BAEFCV (proposed)**	**Random Search**	**Bayesian Optimization (TPE)**
Boston Housing	R^2 test	0,8899 ± 0,0289	0,8896 ± 0,0292	0,8888 ± 0,0269
	RMSE test	2,9927 ± 0,4867	2,9869 ± 0,4630	3,0041 ± 0,4327
	MSE test	9,1814 ± 3,2447	9,1250 ± 2,9333	9,2025 ± 2,6851
Friedman #1	R^2 test	0,9566 ± 0,0067	0,9568 ± 0,0066	0,9562 ± 0,0062
	RMSE test	1,0497 ± 0,0644	1,0462 ± 0,0576	1,0547 ± 0,0543
	MSE test	1,1057 ± 0,1378	1,0978 ± 0,1213	1,1153 ± 0,1163

Concrete Compressive Strength	R^2 test	0,9112 ± 0,0094	0,9114 ± 0,0101	0,9114 ± 0,0131
	RMSE test	4,9834 ± 0,2703	4,9752 ± 0,2914	4,9683 ± 0,3495
	MSE test	24,9035 ± 2,6879	24,8336 ± 2,8995	24,7997 ± 3,5143
Air Quality (CO)	R^2 test	0,9254 ± 0,0073	0,9236 ± 0,0069	0,9242 ± 0,0055
	RMSE test	0,3911 ± 0,0166	0,3958 ± 0,0174	0,3945 ± 0,0118
	MSE test	0,1532 ± 0,0130	0,1570 ± 0,0136	0,1558 ± 0,0094
Air Quality (NMHC)	R^2 test	0,9366 ± 0,0130	0,9374 ± 0,0128	0,9348 ± 0,0128
	RMSE test	50,744 ± 4,617	50,442 ± 4,959	51,529 ± 4,954
	MSE test	2595,3 ± 470,5	2567,8 ± 511,6	2678,6 ± 513,0
Energy Efficiency (Y1)	R^2 test	0,9983 ± 0,0003	0,9983 ± 0,0003	0,9982 ± 0,0003
	RMSE test	0,4051 ± 0,0337	0,4118 ± 0,0382	0,4266 ± 0,0365
	MSE test	0,1652 ± 0,0277	0,1710 ± 0,0324	0,1833 ± 0,0321
Energy Efficiency (Y2)	R^2 test	0,9956 ± 0,0022	0,9940 ± 0,0015	0,9947 ± 0,0017
	RMSE test	0,6174 ± 0,1168	0,7266 ± 0,0893	0,6808 ± 0,1018
	MSE test	0,3941 ± 0,1813	0,5356 ± 0,1338	0,4733 ± 0,1430

For Air Quality CO and Energy Efficiency (Y1, Y2), BAEFCV achieved the highest R^2 among the three methods. Random Search was superior on Friedman #1, Concrete Strength, and Air Quality NMHC on at least one error metric. In terms of RMSE and MSE, BAEFCV and Random Search were comparable to each other, each best on three of the seven datasets, with Bayesian Optimization being the best method on Concrete Strength. The lowest variability was observed for Energy Efficiency Y1 across all three methods (R^2 $\sigma \leq 0.0003$).

Overfitting analysis

Table 5 presents the differences between training and test performance. The ΔR^2 metric (training R^2 − test R^2) quantifies the performance drop; the RMSE ratio (test RMSE / training RMSE) indicates error degradation. Lower ΔR^2 values and ratios closer to 1.0 indicate better generalization.

Table 5 - Overfitting analysis: comparison between training and test performance (mean values over twenty seeds). AQ: Air Quality; EE: Energy Efficiency

Dataset	**Metric**	**BAEFCV**	**Random search**	**Bayesian optim.**
Boston Housing	ΔR^2	0.0590	0.0623	0.0646
	RMSE ratio	1,523	1,611	1,608
Friedman #1	ΔR^2	0,0063	0,0056	0,0062
	RMSE ratio	1,073	1,062	1,071
Concrete strength	ΔR^2	0,0445	0,0479	0,0458
	RMSE ratio	1,459	1,512	1,524
AQ (CO)	ΔR^2	0,0115	0,0103	0.0120
	RMSE ratio	1.182	1.072	1.090
AQ (NMHC)	ΔR^2	0.0196	0.0181	0.0158
	RMSE ratio	1.084	1.165	1.126
EE (Y1)	ΔR^2	0.0007	0.0007	0.0007

	RMSE ratio	1.364	1.334	1.264
EE (Y2)	ΔR^2	0.0028	0.0036	0.0034
	RMSE ratio	1.664	1.673	1.754

With twenty runs, the three methods show comparable and moderate overfitting values: the maximum ΔR^2 was 0.0646 (Bayesian Optimization on Boston Housing) and the maximum RMSE ratio was 1.754 (Bayesian Optimization on Energy Efficiency Y2), with no method exceeding a ratio of 2.0. BAEFCV obtained the lowest ΔR^2 on Boston Housing (0.0590), Concrete Strength (0.0445), and Energy Efficiency Y2 (0.0028); Random Search on Friedman #1 (0.0056) and Air Quality CO (0.0103); Bayesian Optimization on Air Quality NMHC (0.0158). No systematic overfitting trend is observed for any of the three methods.

Statistical significance analysis

For each dataset i, the paired difference was defined as di = Metric(reference, i) − Metric(proposed, i), so that positive values indicate better performance of the proposed algorithm. Table 6 summarizes the results of the Wilcoxon test.

Table 6 - Wilcoxon signed-rank test results between BAEFCV and the reference methods on seven regression tasks (twenty runs)

Comparison	**Metric**	**Positive diff.**	**Negative diff.**	**W statistic**	**p-value**	**Significant α=0.05**
Random search	MSE	3	4	9.000	0.469	No
Random search	R^2	4	3	12,000	0,813	No

Bayesian optim.	MSE	6	1	6,000	0,219	No
Bayesian optim.	R^2	6	1	2,000	0,047	Yes

BAEFCV achieved a statistically significant improvement in R^2 versus Bayesian Optimization ($p = 0,047$), outperforming it on 6 of 7 datasets. The difference in MSE versus Bayesian Optimization, although favorable to BAEFCV on 6 of 7 datasets, did not reach statistical significance with twenty runs ($p = 0,219$). No significant differences were observed versus Random Search on any metric (MSE: $p = 0,469$; R^2: $p = 0,813$), with results practically equivalent between the two methods.

Parameter sensitivity analysis

A sensitivity analysis was performed on the Air Quality (CO) dataset by varying two key parameters: the maximum number of iterations without improvement (MaxIterNoImprovement $\in \{15, 30, 45\}$) and the maximum number of hidden layers (max. layers $\in \{2, 3, 4\}$). Table 7 summarizes the results.

Tabla 7 - Sensitivity analysis on Air Quality (CO): effect of the maximum number of hidden layers and the patience parameter on test performance and computational cost

Max. layers	**MaxIterNoImprovement**	**R^2 test (mean ± sd)**	**RMSE test (mean ± sd)**	**Time (min)**
2	15	0,9755 ± 0,0079	0,2174 ± 0,0396	1,34
2	30	0,9718 ± 0,0058	0,2334 ± 0,0248	2,09
3	30	0,9726 ± 0,0044	0,2132 ± 0,0084	6,49
4	45	0,9747 ± 0,0047	0,2115 ± 0,0292	35,64

The simplest configuration (max. layers = 2, MaxIterNoImprovement = 15) achieved competitive results ($R^2 = 0,9755$, RMSE = 0,2174) in only 1,34

minutes. Increasing to four layers with patience 45 slightly improved the RMSE (0,2115) but increased the time to 35,64 minutes. The small standard deviations (R^2 $\sigma \leq 0,0079$) confirm the algorithm's stability under moderate parameter variations.

Discussion

The results obtained validate the applicability of the proposed algorithm for optimizing MLP architectures in non-linear regression problems. The method consistently identified configurations with high coefficients of determination and low mean squared errors across the five datasets evaluated. The multi-objective formulation based on Chebyshev distance proved effective for simultaneously controlling performance on the three sets (training, validation, and test), reducing the risk of overfitting without explicit regularization terms.

When compared with GeNNsem [11], which uses an ensemble of 50 networks and reaches $R^2 = 0,9997$ on Friedman #1 and $R^2 = 0,9418$ on Boston Housing, the performance gap is expected given that ensemble averaging reduces variance and improves generalization. Nevertheless, the proposed algorithm approaches that level ($R^2 = 0,9566$ on Friedman #1) with a single optimized network, which implies substantial advantages in structural simplicity, computational efficiency, interpretability, and ease of deployment in resource-constrained environments.

The main limitations identified are: higher computational cost relative to random search (5–10 times slower); lack of statistical significance versus random search with the current number of test datasets, and versus Bayesian Optimization on the error metric (MSE), despite showing significance in R^2; dependence on R^2 target values derived from linear ANOVA, which may be a

weak reference in highly non-linear problems; and scalability not explored in high-dimensional problems or with millions of samples.

Conclusions

The proposed algorithm for hyperparameter optimization in MLP networks applied to continuous non-linear regression yields the following main results:

1. The method achieved R^2 values between 0,8899 (Boston Housing) and 0,9983 (Energy Efficiency Y1), outperforming Bayesian Optimization in R^2 on 6 of 7 datasets with a statistically significant difference ($p = 0,047$).

2. The overfitting analysis showed that all three methods maintain moderate ΔR^2 values (0.0007-0.0590) across all datasets, with no evidence of systematic overfitting in any of them.

3. The low variability across independent runs (R^2 $\sigma = 0.0003$ for Energy Efficiency Y1; $\sigma = 0.0067$ for Friedman 1) confirms the reproducibility of the method.

4. A single network optimized with this method can approach the performance level of ensembles of 50 networks (GeNNsem), while maintaining structural simplicity.

5. The sensitivity analysis confirmed the method's robustness under moderate parameter variations, with simplified configurations offering a favorable accuracy-to-computational-cost ratio.

Future work is proposed along the following lines: implementation of parallelization strategies to reduce computation time; validation on additional benchmark datasets to strengthen the statistical evidence versus random search; integration of automatic variable selection within the variable-code scheme; and extension to deeper architectures (convolutional networks).

References

[1] BENARDOS, P. G.; VOSNIAKOS, G. C., «Optimizing feedforward artificial neural network architecture», Engineering Applications of Artificial Intelligence [online], 2007, vol. 20, no. 3, p. 365-382. Available at: <doi:10.1016/j.engappai.2006.06.005>.

[2] RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J., «Learning internal representations by error propagation», in Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambridge, MIT Press, 1986, vol. 1, p. 318-362.

[3] RAIAAN, M. A. K.; et al., «A systematic review of hyperparameter optimization techniques in convolutional neural networks», Decision Analytics Journal [online], 2024, vol. 11, art. 100470. Available at: <doi:10.1016/j.dajour.2024.100470>.

[4] EL-HASSANI, F. Z.; et al., «A new optimization model for MLP hyperparameter tuning: modeling and resolution by real-coded genetic algorithm», Neural Processing Letters [online], 2024, vol. 56. Available at: <doi:10.1007/s11063-024-11578-0>.

[5] KAUR, B. P.; et al., «A genetic algorithm aided hyper parameter optimization based ensemble model for respiratory disease prediction with explainable AI», PLoS One [online], 2024, vol. 19. Available at: <doi:10.1371/journal.pone.0308015>.

[6] NGUYEN-TRONG, K.; NGUYEN, T. T. T., «Optimization of multi-layer perceptron deep neural networks using genetic algorithms for hand gesture recognition», Journal of Computer Science [online], 2022, vol. 18, p. 57-66. Available at: <doi:10.3844/jcssp.2022.57.66>.

[7] LI, D.; et al., «Landslide susceptibility prediction using particle-swarm-optimized multilayer perceptron», Applied Sciences [online], 2019, vol. 9, no. 18. Available at: <doi:10.3390/app9183664>.

[8] ALI, A.; et al., «An optimized multilayer perceptron-based network intrusion detection using gray wolf optimization», Computers & Electrical Engineering [online], 2024, vol. 120, art. 109838. Available at: <doi:10.1016/j.compeleceng.2024.109838>.

[9] TIEP, N. H.; et al., «A new hyperparameter tuning framework for regression tasks in deep neural network: combined-sampling algorithm to search the optimized hyperparameters», Mathematics [online], 2024, vol. 12, no. 24. Available at: <doi:10.3390/math12243892>.

[10] MARTÍNEZ-COMESAÑA, M.; et al., «Use of optimised MLP neural networks for spatiotemporal estimation of indoor environmental conditions of existing buildings», Building and Environment [online], 2021, vol. 205, art. 108243. Available at: <doi:10.1016/j.buildenv.2021.108243>.

[11] KRSTIC, L.; et al., «Evolutionary approach for composing a thoroughly optimized ensemble of regression neural networks», Egyptian Informatics Journal [online], 2024, vol. 28, art. 100581. Available at: <doi:10.1016/j.eij.2024.100581>.

[12] ASGHER, U.; et al., «Novel metaheuristic approach: integration of variables method (IVM) and human-machine interaction for subjective evaluation», in International Conference on Applied Human Factors and Ergonomics (AHFE 2019), Advances in Intelligent Systems and Computing, vol. 965, Cham, Springer, 2019, p. 467-478. Available at: <doi:10.1007/978-3-030-51041-1_50>.

[13] ARZOLA-RUIZ, J.; et al., «Using adaptive integration of variables algorithm for analysis and optimization of 2D irregular nesting problem», in

AHFE 2019, Advances in Intelligent Systems and Computing, vol. 965, Cham, Springer, 2019, p. 479-489. Available at: <doi:10.1007/978-3-030-20473-0_47>.

[14] VALDES, O. M.; ARZOLA-RUIZ, J., «Exploration of variable codes algorithm for linings materials and its thickness selection of steel casting ladles», IEEE Latin America Transactions [online], 2017, vol. 15, p. 1528-1535. Available at: <doi:10.1109/TLA.2017.7994802>.

[15] VITO, S., Air Quality [online], UCI Machine Learning Repository, 2008 [accessed: 2026-03-18]. Available at: <https://archive.ics.uci.edu/dataset/360/air+quality>.

[16] YEH, I.-C., Concrete Compressive Strength [online], UCI Machine Learning Repository, 1998 [accessed: 2026-03-18]. Available at: <https://archive.ics.uci.edu/dataset/165/concrete+compressive+strength>.

[17] TSANAS, A.; XIFARA, A., Energy Efficiency [online], UCI Machine Learning Repository, 2012 [accessed: 2026-03-18]. Available at: <https://archive.ics.uci.edu/dataset/242/energy+efficiency>.

[18] HARRISON, D.; RUBINFELD, D. L., «Hedonic housing prices and the demand for clean air», Journal of Environmental Economics and Management [online], 1978, vol. 5, p. 81-102. Available at: <doi:10.1016/0095-0696(78)90006-2>.

[19] FRIEDMAN, J. H., «Multivariate adaptive regression splines», The Annals of Statistics [online], 1991, vol. 19, no. 1, p. 1-67. Available at: <doi:10.1214/aos/1176347963>.

[20] GLOROT, X.; BENGIO, Y., «Understanding the difficulty of training deep feedforward neural networks», in Proceedings of the 13th International

Conference on Artificial Intelligence and Statistics (AISTATS 2010), JMLR Workshop and Conference Proceedings, vol. 9, 2010, p. 249-256.

[21] ASGHER, U.; et al., «Multi-level optimization of reactive power compensation in industrial nets with heuristic modelling techniques», in AHFE 2020, Advances in Intelligent Systems and Computing, vol. 1213, Cham, Springer, 2020, p. 429-438. Available at: <doi:10.1007/978-3-030-51041-1_57>.

[22] HECHAVARRÍA HERNÁNDEZ, J. R.; et al., «Novel multi-objective optimization algorithm incorporating decisions factors in design modeling of hydraulic nets», in AHFE 2018, Advances in Intelligent Systems and Computing, vol. 781, Cham, Springer, 2018, p. 690-696. Available at: <doi:10.1007/978-3-319-94709-9_67>.

[23] ALHARBI, A. H.; et al., «Forecasting of energy efficiency in buildings using multilayer perceptron regressor with waterwheel plant algorithm hyperparameter», Frontiers in Energy Research [online], 2024, vol. 12, art. 1393794. Available at: <doi:10.3389/fenrg.2024.1393794>.

[24] ZHANG, Y.; et al., «Predicting the compressive strength of high-performance concrete using an interpretable machine learning model», Scientific Reports [online], 2024, vol. 14. Available at: <doi:10.1038/s41598-024-79502-z>.

[25] YURSHIN, V. G.; STANOVOV, V. V., «Artificial neural network architecture tuning algorithm», in EpCT HMMOCS 2022 International Workshop on Hybrid Methods of Modeling and Optimization in Complex Systems [online], 2023. Available at: <doi:10.15405/epct.23021.29>.

[26] WALPOLE, R. E.; MYERS, R. H.; MYERS, S. L., Probability and Statistics for Engineers, Mexico, Pearson Educación, 1999, ISBN 978-9701702642.

[27] KTARI, A.; ELMANSORI, M., «Bridging FEM and artificial neural network in gating system design for smart 3D sand casting», Procedia Manufacturing [online], 2020, vol. 52, p. 795-800. Available at: <doi:10.1016/j.promfg.2020.10.111>.

Conflict of interest

The authors declare that no conflicts of interest exist.

Author contributions:

Javier Carballo Pérez: Conceptualization of the algorithm's computational implementation; software development; design and execution of the experiments; validation of results; data analysis and interpretation; preparation of figures and tables; writing of the original draft and manuscript revision.

José Arzola Ruiz: Conception and development of the heuristic method used in the research; scientific supervision; methodological formulation; interpretation of results; critical review of the manuscript's intellectual content and approval of the final version.

Both authors participated in the discussion of the results, the review and editing of the manuscript, and approved the final version for publication.

About the authors:

Javier Carballo Pérez: Industrial Engineer from the Technological University of Havana «José Antonio Echeverría» (CUJAE), graduated with Gold Degree distinction, Scientific Merit Award. Professor at the Faculty of Industrial Engineering of CUJAE and researcher in optimization, artificial intelligence, and data analysis, with an emphasis on the optimization of artificial neural network architectures through metaheuristic methods. He has participated in

industrial process optimization projects and is the author of national and international publications and presentations in the areas of industrial engineering and artificial intelligence.

Email: carballoperezjavier0@gmail.com. ORCID: https://orcid.org/0009-0000-0908-136X.

José Arzola Ruiz: Metallurgical Engineer, specialization in Automation since 1971. PhD in Technical Cybernetics, since 1982, from the Saint Petersburg Polytechnic University, Russia. His research focuses on the analysis and synthesis of engineering systems, associated methods, and their applications. He has authored 2 books and around 100 published articles in his research field. Professor at the Faculty of Mechanical Engineering of CUJAE.

Email: josearzolaruiz1945@gmail.com. ORCID: https://orcid.org/0000-0002-4101-878X