Metaheuristic Algorithms with Applications

www.maa.reapress.com

Metaheur. Algor. Appl. Vol. 1, No. 4 (2024) 385–391.

Paper Type: Original Article

Hybrid Metaheuristic Algorithms for Deep Neural Network Hyperparameter Optimization: A Comprehensive Review and Framework

Zahra Montazeri*

Department of Computer Engineering, Ayandegan University, Tonkabon, Iran; zahra.montazeri@aihe.ac.ir.

Citation:

Received: 28 March 2024 *Revised: 06 June 2024* *Accepted: 15 August 2024*	Montazeri, Z. (2026). Hybrid metaheuristic algorithms for deep neural network hyperparameter optimization: A comprehensive review and framework. *Metaheuristic algorithms with applications, 1*(4), 385-391.

Abstract

Deep Neural Networks (DNNs) have achieved state-of-the-art performance across numerous domains, yet their efficacy is intrinsically tied to the selection of optimal hyperparameters, a process known as Hyperparameter Optimization (HPO). Traditional exhaustive search methods, such as Grid Search (GS) and Random Search (RS), suffer from prohibitive computational costs and poor scalability, particularly as model complexity and dataset size increase. This review addresses the limitations of conventional HPO techniques by comprehensively analyzing the application of Metaheuristic Algorithms (MAs) in this domain. We formulate the HPO problem as a black-box global optimization task, clearly defining the objective function and the constrained search space. Furthermore, this paper introduces a structured taxonomy for Hybrid Metaheuristic Algorithms (HMAs), classifying them into sequential, parallel, and integrated architectures. We provide detailed reviews of foundational algorithms: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Differential Evolution (DE), and subsequently analyze their synergistic combinations, focusing on frameworks like GA-PSO and Memetic Algorithms. To demonstrate practical efficacy, a case study involving the hyperparameter tuning of a Convolutional Neural Network (CNN) on the Canadian Institute for Advanced Research-10 (CIFAR-10) benchmark is presented, illustrating how HMAs balance global exploration with fine-grained exploitation. This comprehensive review establishes a foundational framework for designing robust and computationally efficient HPO strategies tailored for modern Deep Learning (DL) systems, serving as a critical resource for future research.

Keywords: Hyperparameter optimization, Deep neural networks, Hybrid metaheuristics, Genetic algorithm, Particle swarm optimization.

1 | Introduction

The proliferation of Deep Learning (DL) models, including Convolutional Neural Networks (CNNs) for vision tasks and Recurrent Neural Networks (RNNs) for sequential data, has necessitated significant

Corresponding Author: zahra.montazeri@aihe.ac.ir

https://doi.org/10.48313/maa.v1i4.106

Licensee System Analytics. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

advancements in model configuration [1]. Central to achieving peak performance from these complex architectures is Hyperparameter Optimization (HPO), the process of selecting configuration settings that govern the training process itself, independent of the learned weights [2]. These hyperparameters, such as learning rate, batch size, number of layers, regularization coefficients, and activation functions, critically influence convergence speed, generalization capability, and final predictive accuracy [3].

The challenge in HPO stems from the high dimensionality and the non-convex, noisy nature of the objective function, which is defined by the model's validation performance (e.g., accuracy or error) [4]. Conventional HPO approaches, namely Grid Search (GS) and Random Search (RS), while conceptually simple, are fundamentally limited. GS exhaustively evaluates every combination in a discretized space, exhibiting exponential computational growth, $O(m^n)$ for m values across n hyperparameters [5]. RS, while statistically more efficient due to its ability to sample the full range of each hyperparameter independently, still provides no guarantee of convergence to optimal regions and wastes evaluations on unpromising configurations [6].

More advanced approaches like Bayesian Optimization (BO), which constructs a probabilistic surrogate model (e.g., Gaussian process) of the objective function, offer improved sample efficiency [7]. However, BO struggles with high-dimensional search spaces (typically >20 dimensions), non-stationary objectives, and conditional hyperparameters. These limitations have driven increasing interest in population-based Metaheuristic Algorithms (MAs), which offer inherent parallelism, derivative-free operation, and robust exploration-exploitation balance [8].

This paper provides a comprehensive review of MAs for Deep Neural Network (DNN) HPO, introduces a structured taxonomy for Hybrid Metaheuristic Algorithms (HMAs), and presents a case study demonstrating the practical advantages of hybrid approaches.

The practical advantages of hybrid approaches.

2 | Foundational Concepts

2.1 | Hyperparameter Optimization Problem Formulation

The HPO problem is formulated as a black-box global optimization task. The objective is to find the hyperparameter vector that minimizes the validation loss:

$$\Theta^* = \arg\min_{\theta \in \Theta} \mathcal{L}(\mathcal{D}_{val}, \quad (1)$$

$$\text{subject to: } w^*(\theta) = \arg\min_w \mathcal{L}(\mathcal{D}_{train}; w, \theta), \quad (2)$$

where θ represents the hyperparameter vector, Θ is the constrained search space, $\mathcal{L}$ is the loss function, $\mathcal{D}_{train}$ and $\mathcal{D}_{val}$ are training and validation datasets respectively, and $w^*(\theta)$ denotes the optimal model weights obtained by training with hyperparameters θ. The objective function is expensive to evaluate (each evaluation requires full model training), noisy (due to stochastic training), and generally non-convex [4].

2.2 | Limitations of Conventional Methods

I. GS: computational complexity O(mn) is prohibitive. For 5 hyperparameters with 10 values each, GS requires 100,000 evaluations. It also misses inter-grid optima.

II. RS: Bergstra and Bengio [6] showed RS is more efficient when some hyperparameters are more important than others, but it still lacks directed search.

III. BO: effective for low-dimensional spaces (<20 dimensions) but scales poorly. Surrogate model updates become expensive. Struggles with categorical and conditional hyperparameters [7].

2.3 | Genetic Algorithm

GA is inspired by natural selection and genetics [9]. The key components of GA are as follows:

I. Encoding: hyperparameters encoded as chromosomes (binary or real-valued).

II. Selection: tournament or roulette-wheel selection based on fitness (validation performance).

III. Crossover: single-point, multi-point, or uniform crossover to combine parent solutions.

IV. Mutation: random perturbation of individual genes to maintain diversity.

V. Population: typically 20–100 individuals, evolved over 50–200 generations.

GA excels at global exploration through its population-based search and genetic operators but can be slow to converge to precise optima due to its discrete crossover operations [10].

2.4 | Particle Swarm Optimization

Particle Swarm Optimization (PSO) simulates the social behavior of bird flocking or fish schooling [11]. Each particle i updates its velocity and position according to:

$$V\,i(t+1) = w \cdot v\,i(t) + c\,1 \cdot r\,1 \cdot (pbest\,i - x\,i(t)) + c\,2 \cdot r\,2 \cdot (gbest - xi(t)), \quad (3)$$

$$x\,i(t+1) = x\,i(t) + v\,i(t+1), \quad (4)$$

where w is the inertia weight (typically 0.4–0.9), c1 and c2 are cognitive and social coefficients (typically 1.5–2.0), r1 and r2 are random numbers in [0, 1], pbesti is the personal best position, and gbest is the global best.

PSO converges faster than GA due to continuous position updates but is prone to premature convergence, especially in high-dimensional spaces with many local optima [12].

2.5 | Ant Colony Optimization

Ant Colony Optimization (ACO) is inspired by the foraging behavior of ants, which deposit pheromones on paths to food sources [13]. The probability of ant k selecting hyperparameter value j for parameter i is:

$$P\,ij\,k = (\tau\,ij\,\alpha \cdot \eta\,ij\,\beta) \,/\, \sum(\tau\,il\,\alpha \cdot \eta\,il\,\beta). \quad (5)$$

Pheromone update follows the rule:

$$\tau\,ij(t+1) = (1-\rho) \cdot \tau\,ij(t) + \Delta\tau ij. \quad (6)$$

where ϱ is the evaporation rate (0.1–0.5), and $\Delta\tau ij$ is the pheromone deposit proportional to solution quality.

ACO is particularly suited for discrete/combinatorial hyperparameters (e.g., number of layers, activation function type) but requires discretization for continuous parameters [14].

2.6 | Differential Evolution

Differential Evolution (DE) is a population-based optimization algorithm that uses difference vectors for perturbation [15]. For each target vector x_i, a donor vector is created:

$$V\,i = x\,r1 + F \cdot (x\,r2 - x\,r3), \quad (7)$$

where r1, r2, r3 are distinct random indices, and F is the scaling factor (typically 0.5–1.0). Crossover with rate CR (0.1–0.9) produces trial vectors. DE is effective for continuous optimization and exhibits good convergence properties but limited exploration in early stages [16].

3 | Hybrid Metaheuristic Algorithm Taxonomy

3.1 | Motivation for Hybridization

No single metaheuristic universally outperforms others across all problem landscapes, a principle formalized by the No Free Lunch theorem [17]. Hybridization combines complementary strengths of individual algorithms:

I. GA: strong global exploration, weak local exploitation.

II. PSO: fast convergence, risk of premature convergence.

III. ACO: excellent for discrete spaces, limited for continuous.

IV. DE: good for continuous optimization, limited exploration.

3.2 | Sequential Hybrid Architecture

In sequential hybrids, one algorithm runs first to provide initial solutions that seed the second algorithm [18]:

I. GA → PSO: GA explores the global landscape (20–50 generations), then PSO refines the best solutions with focused exploitation (30–50 iterations).

II. PSO → GA: PSO provides fast initial convergence; GA applies crossover/mutation to escape local optima.

Advantages: clear separation of exploration and exploitation phases.

Disadvantages: transition timing is critical; early switching loses global coverage, late switching wastes computation.

3.3 | Parallel Hybrid Architecture

Multiple algorithms run simultaneously on the same population or subpopulations, periodically sharing best solutions:

I. Island Model: population divided into islands, each optimized by a different MA. Migration of top solutions occurs every k generations.

II. Cooperative Co-evolution: different MAs optimize different subsets of hyperparameters simultaneously.

Advantages: leverages multiple search strategies; natural parallelism for distributed computing.

Disadvantages: communication overhead; requires careful migration/synchronization policies.

3.4 | Integrated Hybrid Architecture

Operators from different MAs are combined within a single algorithmic framework [19]:

I. Memetic Algorithms: GA + local search (gradient descent or hill climbing) applied to each individual after genetic operations.

II. Hybrid Operators: PSO velocity update combined with GA crossover; ACO pheromone guides DE mutation.

Advantages: tight integration allows dynamic adaptation; can switch strategies per individual.

Disadvantages: complex implementation; more hyperparameters for the optimizer itself.

Table 1. Comparison of hybrid metaheuristic architectures.

Architecture	Exploration	Exploitation	Computational Cost	Implementation Complexity
Sequential	High (first phase)	High (second phase)	Moderate	Low
Parallel	High (multiple strategies)	Moderate	High	Moderate
Integrated	Moderate–high	High	Moderate–high	High

Table 2. Individual metaheuristic algorithm characteristics for HPO.

Algorithm	Search Type	Convergence Speed	Global Exploration	Local Exploitation	Best Suited For
GA	Population-based	Slow–moderate	Excellent	Moderate	Mixed discrete/Continuous
PSO	Swarm-based	Fast	Good	Good	Continuous spaces
ACO	Colony-based	Moderate	Good	Moderate	Discrete/Combinatorial
DE	Population-based	Moderate–fast	Moderate	Excellent	Continuous spaces

4 | Case Study: Convolutional Neural Network Hyperparameter Optimization on Canadian Institute for Advanced Research-10

4.1 | Experimental Setup

A CNN architecture was tuned for the Canadian Institute for Advanced Research-10 (CIFAR-10) image classification benchmark (60,000 32×32 color images, 10 classes) [13]. The hyperparameters tuned were:

I. Learning rate: [0.0001, 0.1] (continuous, log-scale).

II. Batch size: {16, 32, 64, 128, 256} (discrete).

III. Number of convolutional filters: {16, 32, 64, 128} per layer (discrete).

IV. Kernel size: {3, 5, 7} (discrete).

V. Dropout rate: [0.0, 0.5] (continuous).

VI. Fitness function: validation accuracy after 50 training epochs. Each evaluation takes approximately 10–15 minutes on a single GPU.

4.2 | Methods Compared

I. GS: exhaustive evaluation over a discretized grid.

II. RS: 200 random configurations.

III. GA-only: population 30, 50 generations.

IV. PSO-only: swarm 30, 50 iterations.

V. GA-PSO Hybrid (Sequential): GA exploration (20 generations) → PSO refinement (30 iterations), population 30.

4.3 | Result

Table 3. CIFAR-10 HPO results.

Method	Best Test Accuracy (%)	Total Evaluations	Convergence Iteration	Wall-Clock Time (Hours)
GS	88.2	1,200	N/A	240
RS	87.5	200	N/A	40
GA-only	89.7	1,500	42	300
PSO-only	90.1	1,500	35	300
GA-PSO hybrid	92.3	1,500	~50 total	300

The GA-PSO hybrid achieved the highest accuracy (92.3%) by leveraging GA's broad exploration in early generations to identify promising regions, followed by PSO's rapid convergence for fine-tuning within those regions. Notably, convergence occurred around iteration 50 (total), compared to iteration 42 for GA-only and 35 for PSO-only, but at a significantly higher accuracy level.

4.4 | Analysis

The results confirm the complementary nature of GA and PSO: GA's crossover and mutation operators effectively explore diverse hyperparameter configurations in the first 20 generations, while PSO's velocity-based updates efficiently exploit the identified promising regions. The hybrid achieves 2.6% higher accuracy than GA-only and 2.2% higher than PSO-only, demonstrating that the sequential combination produces synergistic benefits exceeding either component alone.

5 | Conclusion and Future Directions

This paper presented a comprehensive review of MAs for DNN HPO. We formulated HPO as a black-box global optimization problem, reviewed four foundational MAs (GA, PSO, ACO, DE), and introduced a

structured taxonomy for Hybrid MAs classifying them into sequential, parallel, and integrated architectures. The case study on CNN hyperparameter tuning for CIFAR-10 demonstrated that the GA-PSO sequential hybrid achieves superior performance compared to individual methods and conventional search approaches.

Future research directions include:

I. Adaptive hybrid switching dynamically selecting which MA to apply based on search progress metrics.

II. Multi-objective HPO simultaneously optimizing accuracy, inference latency, and model size using multi-objective MAs like Non-dominated Sorting Genetic Algorithm II (NSGA-II).

III. Scalability extending HMAs to larger architectures (e.g., transformers with 100+ hyperparameters) through hierarchical decomposition.

IV. Transfer learning for HPO using meta-learning to warm-start MA populations based on previously optimized similar architectures.

V. Integration with Neural Architecture Search (NAS), jointly optimizing architecture and hyperparameters [15].

Authors' Contributions

All aspects of the research and manuscript preparation were carried out by the author. The author has read and approved the final version of the manuscript.

Funding

Not applicable.

Data Availability

All data supporting the reported findings in this research paper are provided within the manuscript.

Conflict of Interest

The author declares that they do not have any conflict of interest.

Consent for Publication

The author confirms consent for the publication of this work.

Ethics Approval and Consent to Participate

This article does not contain any studies with human participants performed by the author.

References

[1] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

[2] Goodfellow, I. (2016). *Deep learning*. MIT Press. http://www.deeplearningbook.org

[3] Smith, L. N. (2017). Cyclical learning rates for training neural networks. *2017 IEEE winter conference on applications of computer vision (WACV)* (pp. 464-472). IEEE. https://doi.org/10.1109/WACV.2017.58

[4] Feurer, M., & Hutter, F. (2019). *Automated machine learning*. Cham: Springer. https://doi.org/10.1007/978-3-030-05318-5

[5] Liashchynskyi, P., & Liashchynskyi, P. (2019). *Grid search, random search, genetic algorithm: A big comparison for NAS*. https://doi.org/10.48550/arXiv.1912.06059

[6] Bergstra, J., & Bengio, Y. (2012). Random search for hyper parameter optimization. *Journal of machine learning research, 13*(2012), 281-305. http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

[7] Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, *25*. https://papers.nips.cc/paper_files/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html

[8] Yang, X. S. (2010). *Nature inspired metaheuristic algorithms*. Luniver Press. https://dl.acm.org/doi/10.5555/1893084

[9] Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press. https://doi.org/10.7551/mitpress/1090.001.0001

[10] Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Publishing Company. https://www.amazon.com/Genetic-Algorithms-Optimization-Machine-Learning/dp/0201157675

[11] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95-international conference on neural networks* (Vol. 4, pp. 1942-1948). IEEE. https://doi.org/10.1109/ICNN.1995.488968

[12] Shi, Y., Eberhart, R. (1998). A modified particle swarm optimizer. *Evolutionary computation proceedings* (Vol. 890, pp. 69–73). IEEE. https://doi.org/10.1109/ICEC.1998.699146

[13] Dorigo, M. (2007). Ant colony optimization. *Scholarpedia*, *2*(3), 1461. http://www.scholarpedia.org/article/Ant_colony_optimization

[14] Socha, K., & Dorigo, M. (2008). Ant colony optimization for continuous domains. *European journal of operational research*, *185*(3), 1155–1173. https://doi.org/10.1016/j.ejor.2006.06.046

[15] Storn, R., & Price, K. (1997). Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, *11*(4), 341–359. https://doi.org/10.1023/A:1008202821328

[16] Das, S., Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, *15*(1), 4-31. https://doi.org/10.1109/TEVC.2010.2059031

[17] Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67–82. https://doi.org/10.1109/4235.585893

[18] Young, S. R., Rose, D. C., Karnowski, T. P., Lim, S. H., & Patton, R. M. (2015). Optimizing deep learning hyper parameters through an evolutionary algorithm. *Proceedings of the workshop on machine learning in hig performance computing environments* (pp. 1–5). MLHPC. https://doi.org/10.1145/2834892.2834896

[19] Lorenzo, P. R., Nalepa, J., Kawulok, M., Ramos, L. S., & Pastor, J. R. (2017). Particle swarm optimization for hyper parameter selection in deep neural networks. *Proceedings of the genetic and evolutionary computation conference* (pp. 481–488). GECCO. https://doi.org/10.1145/3071178.3071208