Journal of International Commercial Law and Technology
Print ISSN: 1901-8401

Website: https://www.jiclt.com/

Article

A Comparative Study of Hyperparameter Optimization Techniques for Machine Learning Models

Article History:

Name of Author:
Dr. Shailendra Kumar Roy

Affiliation:
Radha Govind University, Ramgarh, Jharkhand

Corresponding Author:
Dr. Shailendra Kumar Roy
(Kavitasinghal181191@gmail.com)

How to cite this article:
Dr. Shailendra Kumar Roy, A Comparative Study of Hyperparameter Optimization Techniques for Machine Learning Models. J Int Commer Law Technol. 2026;7(1):1466–1478.

Received: *02-04-2026*
Revised: *24-04-2026*
Accepted: *04-05-2026*
Published: *19-06-2026*

Abstract: Hyperparameter optimization (HPO) remains one of the most critical and computationally demanding challenges in deploying machine learning models effectively. Unlike model parameters that are learned during training, hyperparameters govern the learning process itself and must be configured prior to training, making their selection both consequential and non-trivial. This review article provides a rigorous comparative analysis of established and emerging HPO techniques, spanning traditional exhaustive methods, probabilistic Bayesian approaches, evolutionary and metaheuristic strategies, gradient-based methods, reinforcement learning-driven search, and neural architecture search frameworks. The paper critically evaluates each family of methods with respect to search efficiency, computational cost, convergence behavior, scalability, and suitability across diverse model classes including support vector machines, random forests, gradient-boosted trees, convolutional neural networks, and recurrent architectures. A structured taxonomy is presented to organize the rapidly growing HPO landscape, accompanied by mathematical formulations of core optimization objectives, acquisition functions, and fitness criteria. Three comparative tables synthesize the literature and provide a side-by-side assessment of method capabilities. The review further addresses challenges unique to high-dimensional search spaces, noisy evaluation functions, and distributed optimization settings. Contemporary trends including AutoML pipelines, federated HPO, green AI, and HPO for large language models are analyzed for their trajectory and open research problems. The findings reveal that no single HPO method universally dominates across all settings; rather, method selection should be guided by problem dimensionality, evaluation budget, and the degree of prior knowledge available. Future research directions emphasizing warm-starting, meta-learning, and explainable HPO are identified as essential for advancing the field.

Keywords: Hyperparameter Optimization, Bayesian Optimization, AutoML, Neural Architecture Search, Evolutionary Algorithms, Random Search, Metaheuristic Optimization, Machine Learning, Deep Learning, Grid Search.

INTRODUCTION

The performance of machine learning (ML) models is profoundly sensitive to hyperparameters configuration choices that exist outside the scope of gradient-based learning but directly shape its outcome. The learning rate in a neural network, the number of trees in a random forest, the regularization coefficient of a support vector machine, and the depth of a decision tree are all hyperparameters that must be determined before training commences. Even marginal differences in these settings can translate into significant discrepancies in generalization accuracy, training stability, and inference efficiency [1].

Historically, hyperparameter selection was treated as an art form, largely dependent on practitioner intuition, domain expertise, and manual trial-and-error. The exponential growth in model complexity particularly with deep neural networks has rendered such manual workflows untenable. Modern architectures may expose

tens to hundreds of hyperparameters, and the interaction effects among these parameters create highly non-linear, non-convex, and often discontinuous objective landscapes [2]. Automated hyperparameter optimization has thus evolved from a convenience into a necessity for scalable, reproducible machine learning.

The field of HPO has matured substantially over the past decade. Early systematic approaches grid search and random search gave way to probabilistic surrogate models under the Bayesian optimization paradigm [3]. Concurrently, biologically inspired evolutionary algorithms and swarm intelligence methods offered derivative-free alternatives suited to mixed search spaces. More recently, gradient-based meta-learning approaches, reinforcement learning controllers, and differentiable architecture search methods have extended HPO to the domain of neural architecture design itself [4].

Despite this proliferation of methods, the literature lacks a unified comparative framework that situates techniques relative to one another across a consistent set of evaluation criteria. Surveys that have appeared in the literature tend to focus on specific subdomains Bayesian optimization [5], evolutionary computation [6], or AutoML [7] without providing the cross-paradigm synthesis that practitioners require to make informed method selection decisions.

This review addresses that gap. We present a comprehensive, critical comparison of HPO techniques organized along a taxonomy that spans the full methodological spectrum. We evaluate methods not merely in terms of reported benchmark performance but with respect to structural properties exploration-exploitation balance, scalability with respect to dimensionality and evaluation cost, robustness to noisy objectives, and compatibility with distributed computing environments. In doing so, we aim to provide both researchers and practitioners with a principled foundation for navigating the crowded HPO landscape.

The remainder of the paper is organized as follows. Section 2 provides background on ML models and the importance of HPO. Section 3 reviews the relevant literature. Section 4 presents a taxonomy of HPO methods. Sections 5 through 13 discuss specific method families in detail. Section 14 offers a comparative analysis, and Sections 15 through 18 address complexity, challenges, and future directions before concluding in Section 19.

BACKGROUND

2.1 Machine Learning Models and Their Hyperparameters

Modern ML models span a wide spectrum of architectures, each characterized by its own hyperparameter space. Support Vector Machines (SVMs) are governed by the regularization constant C, the kernel type (linear, RBF, polynomial), and kernel-specific parameters such as γ in the RBF kernel. Random Forests require configuration of the number of trees, maximum tree depth, minimum samples per leaf, and feature subsampling ratio [8]. XGBoost and gradient-boosted trees introduce additional complexity: learning rate, subsample ratio, column sampling parameters, and tree regularization coefficients λ and α all interact in nuanced ways [9].

Deep Neural Networks (DNNs) expose the largest and most challenging hyperparameter spaces. The learning rate, batch size, optimizer choice, weight initialization strategy, dropout rates, number of layers, and layer width each affect training dynamics, and their effects are neither independent nor monotonic. Convolutional Neural Networks (CNNs) add architectural hyperparameters: kernel sizes, stride, padding, pooling strategies, and the number of filters per layer. Long Short-Term Memory (LSTM) networks require tuning of hidden state dimensionality, sequence length, gradient clipping thresholds, and forget gate biases, often compounded by sensitivity to learning rate schedules [10].

2.2 Importance of Hyperparameter Optimization

The importance of systematic HPO can be understood through the lens of the generalization gap. A model trained with suboptimal hyperparameters may exhibit significantly higher variance or bias than the same architecture with well-tuned settings, irrespective of the volume of training data or sophistication of the architecture [11]. Empirical studies have consistently demonstrated that HPO can recover several percentage points of classification accuracy that naive defaults leave on the table [1]. In high-stakes applications medical diagnosis, financial forecasting, autonomous systems this margin is operationally significant.

2.3 Challenges in HPO

The HPO problem is structurally adversarial to standard optimization methods. The objective function validation performance as a function of hyperparameters is black-box (no gradient information), expensive to evaluate (each evaluation requires a full model training run), noisy (due to stochastic optimization and data splits), and non-convex with multiple local optima [12]. The search space is frequently mixed-type, combining continuous variables (learning rate), integers (number of layers), and categorical choices (activation function), with conditional dependencies (dropout rate is only relevant if dropout is enabled). High dimensionality further exacerbates the challenge: in deep learning, it is not uncommon to encounter 20–50 meaningful hyperparameters, making the search space astronomically large [13].

LITERATURE REVIEW

The formal study of automated HPO was catalyzed by the work of Bergstra and Bengio [14], who demonstrated

that random search outperforms grid search for most practical HPO problems due to its superior coverage of the effective dimensions of the search space. This foundational insight displaced the naive exhaustive paradigm and motivated the development of more sophisticated adaptive methods.

Bayesian optimization with Gaussian Process surrogates was systematically applied to HPO by Snoek et al. [3], whose Spearmint framework set the standard for probabilistic HPO and introduced the use of Expected Improvement as an acquisition function. Subsequent work refined the acquisition function landscape: Hernández-Lobato et al. [15] introduced Predictive Entropy Search, and Wang and Jegelka [49] proposed Max-value Entropy Search (MES) with improved scalability.

The limitations of Gaussian Processes particularly their cubic computational complexity in the number of observations motivated tree-structured surrogate models. Bergstra et al. [17] introduced the Tree-structured Parzen Estimator (TPE), which models $p(x|y)$ and $p(y)$ separately and scales more gracefully to high-dimensional and conditional search spaces. TPE forms the backbone of the widely used Hyperopt library and has demonstrated strong empirical performance on deep learning benchmarks [18].

Evolutionary approaches to HPO have a longer history, with early genetic algorithm applications to neural network weight and architecture optimization reviewed by Yao [19]. More recently, CMA-ES (Covariance Matrix Adaptation Evolution Strategy) has been applied to HPO by Loshchilov and Hutter [20], demonstrating competitive performance in continuous spaces. Differential Evolution and Particle Swarm Optimization have been applied to SVM and ensemble method tuning with promising results [21].

Population-based Training (PBT), introduced by Jaderberg et al. [22], represented a paradigm shift by simultaneously training and optimizing hyperparameters across a population of models, allowing schedules particularly learning rate schedules to be discovered dynamically rather than fixed at initialization. This approach has proven especially impactful for reinforcement learning and large-scale vision models.

Neural Architecture Search (NAS) extended HPO from scalar hyperparameters to the architecture itself. The seminal NAS work of Zoph and Le [23] used reinforcement learning to search over architecture descriptions, achieving state-of-the-art accuracy at extraordinary computational cost. Subsequent efficiency improvements ENAS [24], DARTS [25], and One-Shot NAS [26] reduced the search cost by orders of magnitude through weight sharing and differentiable relaxation of the architecture choice.

The AutoML paradigm formalized the pipeline from raw data to deployed model as a unified optimization problem. Auto-WEKA [27], Auto-sklearn [28], and H2O AutoML integrate HPO with algorithm selection under the CASH (Combined Algorithm Selection and Hyperparameter optimization) framework. Recent work has further extended AutoML to handle neural architecture design, data preprocessing, and feature engineering jointly [29].

Multi-objective HPO, where accuracy is jointly optimized with model size, inference latency, or energy consumption, has received growing attention [30]. Pareto-front exploration under constrained budgets, combined with surrogate modeling, has emerged as a principled framework for resource-aware model deployment [31].

TAXONOMY OF HYPERPARAMETER OPTIMIZATION TECHNIQUES

The HPO landscape can be organized into a hierarchical taxonomy based on the nature of the search strategy and the information used to guide successive evaluations. Figure 1 presents this taxonomy.

In addition to traditional, Bayesian, evolutionary, and learning-based approaches, recent work has introduced diversity-aware HPO methods that explicitly aim to improve coverage of the search space. ART-HPO is one such method, adapting adaptive random testing to reduce clustering among evaluated hyperparameter configurations [16]. This category is especially relevant when the evaluation budget is limited and broad exploration of the search space is desired.

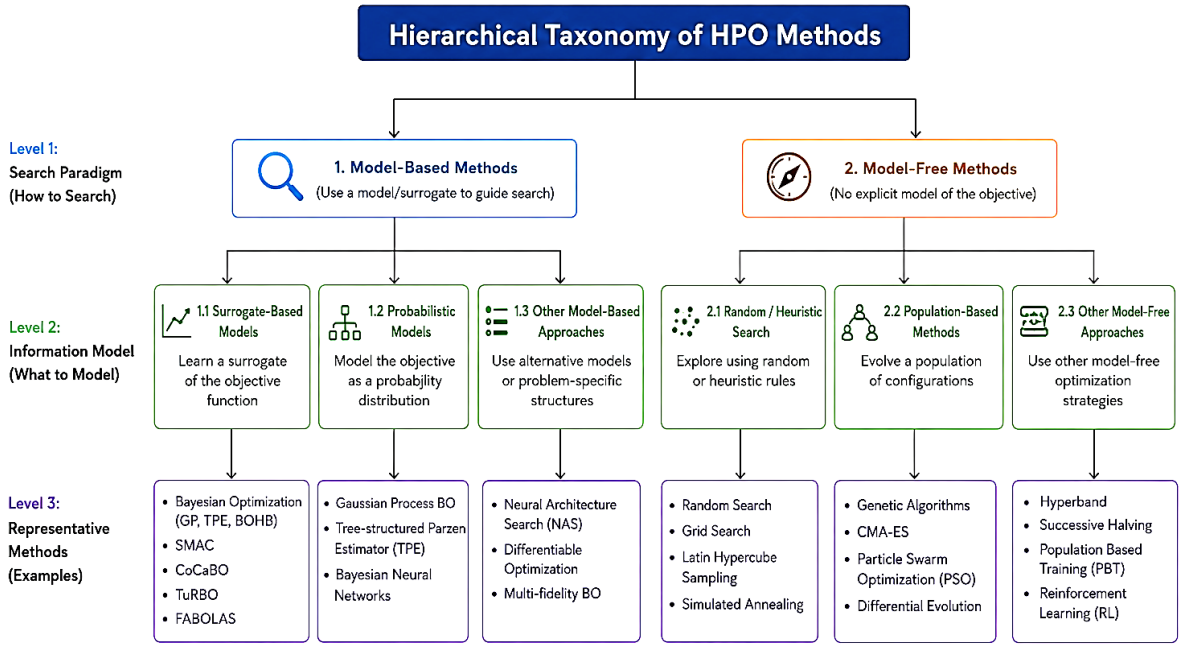

Figure 1: Hierarchical taxonomy of HPO methods, organized by search paradigm and information model.

Figure 2 presents the general HPO workflow, where the optimizer repeatedly proposes candidate hyperparameters, evaluates model performance, and updates its search strategy until a stopping criterion is met, after which the best configuration is returned.

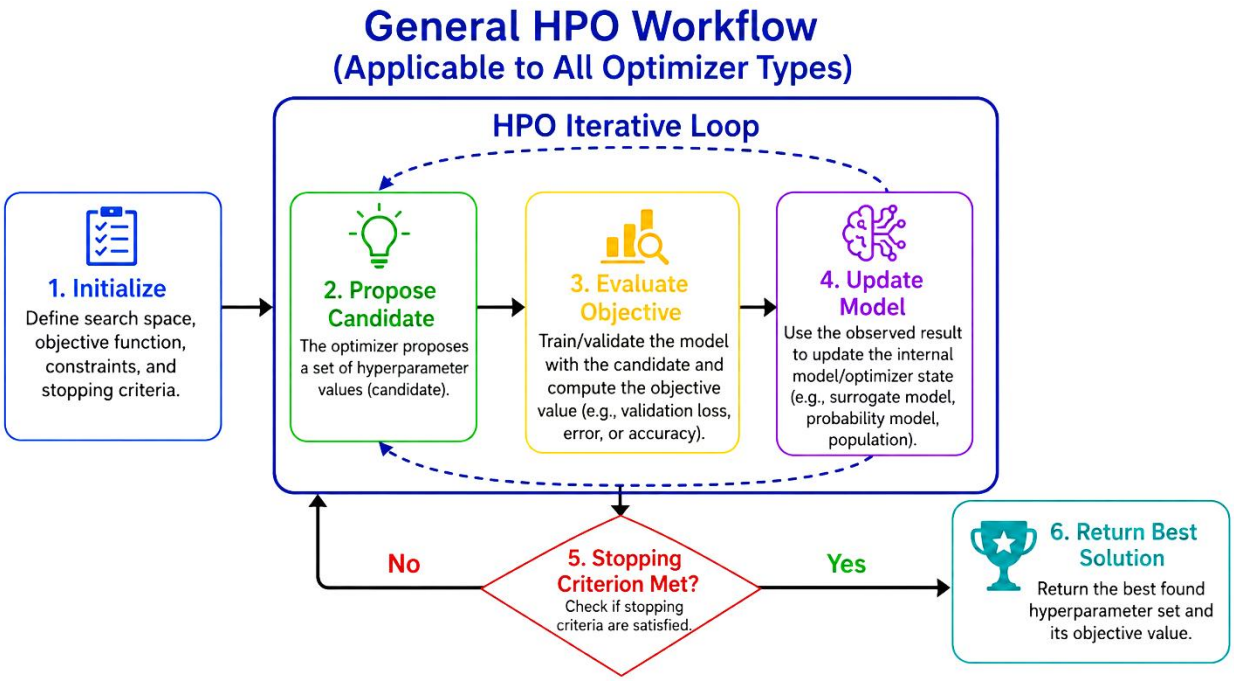

Figure 2: General HPO workflow applicable to all optimizer types

5. Problem Formulation

The HPO problem can be formally stated as follows. Let $\mathcal{H}$ denote the hyperparameter search space, a product space of continuous, integer, and categorical domains:

$$\mathcal{H} = \mathcal{H}_1 \times \mathcal{H}_2 \times \cdots \times \mathcal{H}_d$$

where d is the number of hyperparameters. The goal is to find:

$$\boldsymbol{\lambda}^* = \text{argmin}_{\boldsymbol{\lambda} \in \mathcal{H}} \; \mathcal{L}_{\text{val}}(f_{\boldsymbol{\lambda}}, \mathcal{D}_{\text{val}})$$

where $f_{\boldsymbol{\lambda}}$ denotes a model trained with hyperparameters $\boldsymbol{\lambda}$ on training set $\mathcal{D}_{\text{train}}$, and $\mathcal{L}_{\text{val}}$ is the validation loss evaluated on $\mathcal{D}_{\text{val}}$. The evaluation function $\mathcal{L}_{\text{val}}(f_{\boldsymbol{\lambda}}, \mathcal{D}_{\text{val}})$ is treated as a black-box oracle, as no closed-form gradient with respect to $\boldsymbol{\lambda}$ is generally available.

TRADITIONAL OPTIMIZATION METHODS

6.1 Grid Search

Grid Search (GS) discretizes each hyperparameter dimension into a finite set of candidate values and evaluates all combinations. For d hyperparameters each with k candidate values, GS requires k^d evaluations an exponential growth that renders it computationally infeasible for $d > 4$ or 5 [14]. Despite its simplicity and exhaustiveness within the discretized space, GS suffers from the curse of dimensionality: for a 10-dimensional space with 5 values per dimension, $5^{10} \approx 10^7$ evaluations are required.

The practical use of GS is therefore restricted to low-dimensional problems, coarse initial exploration, or final refinement around a known good region. Its advantage lies in full reproducibility and ease of parallelization, as evaluations are independent. However, GS wastes resources on unimportant dimensions: if only 2 of 10 hyperparameters materially affect performance, GS still exhaustively evaluates all combinations of the remaining 8 [14].

6.2 Random Search

Bergstra and Bengio [14] demonstrated analytically and empirically that Random Search (RS) is more efficient than GS when only a subset of hyperparameters is important. By independently and uniformly sampling from continuous distributions rather than a discrete grid, RS covers the effective dimensions more densely per evaluation. For a budget of N evaluations, the best value found along any single dimension follows the distribution of the maximum of N uniform samples, whereas GS covers only N/k^{d-1} distinct values per dimension.

RS remains a competitive baseline against which more sophisticated methods must justify their additional complexity. It is trivially parallelizable, requires no surrogate model, and carries no risk of surrogate model misspecification. Its primary limitation is the absence of adaptive refinement: RS does not learn from previous evaluations, making it inefficient for problems requiring more than a few dozen evaluations.

A related limitation of Random Search is that independently sampled configurations may cluster in some regions while leaving other parts of the hyperparameter space unexplored. ART-HPO addresses this issue by adapting adaptive random testing principles to hyperparameter optimization, encouraging newly selected configurations to be more spatially diverse from previously evaluated ones [16]. In this sense, ART-HPO can be viewed as a diversity-aware extension of random search rather than a surrogate-based optimizer.

BAYESIAN OPTIMIZATION APPROACHES

Bayesian Optimization (BO) is the dominant paradigm for sample-efficient HPO. It constructs a probabilistic surrogate model $\hat{f}(\lambda)$ of the objective function and uses an acquisition function $\alpha(\lambda)$ to determine the next candidate configuration to evaluate by trading off exploration of uncertain regions against exploitation of known good regions [5].

7.1 Gaussian Process Surrogate

The standard BO framework models the objective as a sample from a Gaussian Process:

$$f(\boldsymbol{\lambda}) \sim \mathcal{GP}\big(\mu(\boldsymbol{\lambda}),\, k(\boldsymbol{\lambda}, \boldsymbol{\lambda}')\big)$$

where $\mu(\boldsymbol{\lambda})$ is the mean function (often set to zero) and $k(\boldsymbol{\lambda}, \boldsymbol{\lambda}')$ is a covariance kernel, typically the Matérn 5/2 kernel:

$$k(\boldsymbol{\lambda}, \boldsymbol{\lambda}') = \left(1 + \frac{\sqrt{5}\, r}{\ell} + \frac{5r^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}\, r}{\ell}\right), \quad r = \| \boldsymbol{\lambda} - \boldsymbol{\lambda}' \|$$

After observing n evaluations $\mathcal{D}_n = \{(\boldsymbol{\lambda}_i, y_i)\}_{i=1}^n$, the posterior predictive distribution at a new point $\boldsymbol{\lambda}$ is Gaussian with mean $\mu_n(\boldsymbol{\lambda})$ and variance $\sigma_n^2(\boldsymbol{\lambda})$ obtained via standard GP conditioning formulae.

7.2 Acquisition Functions

The Expected Improvement (EI) acquisition function, introduced for HPO by Snoek et al. [3], is defined as:

$$\alpha_{\text{EI}}(\boldsymbol{\lambda}) = \mathbb{E}[\max(f^* - f(\boldsymbol{\lambda}),\, 0)]$$

where $f^* = \min_i y_i$ is the best observed value. Under the GP posterior, this has a closed form:

$$\alpha_{\text{EI}}(\boldsymbol{\lambda}) = \big(f^* - \mu_n(\boldsymbol{\lambda})\big)\, \Phi(Z) + \sigma_n(\boldsymbol{\lambda})\, \phi(Z), \quad Z = \frac{f^* - \mu_n(\boldsymbol{\lambda})}{\sigma_n(\boldsymbol{\lambda})}$$

The Upper Confidence Bound (UCB) acquisition:

$$\alpha_{\text{UCB}}(\boldsymbol{\lambda}) = \mu_n(\boldsymbol{\lambda}) - \kappa\, \sigma_n(\boldsymbol{\lambda})$$

with $\kappa > 0$ controlling the exploration-exploitation trade-off. Max-value Entropy Search (MES) [49] selects points that maximally reduce uncertainty about f^*:

$$\alpha_{\text{MES}}(\boldsymbol{\lambda}) = H[f^*] - \mathbb{E}_{f(\boldsymbol{\lambda})}\big[H[f^* \mid f(\boldsymbol{\lambda})]\big]$$

7.3 Tree-Structured Parzen Estimator (TPE)

The TPE, introduced by Bergstra et al. [17], bypasses the GP entirely by modeling the density of hyperparameter configurations as:

$$p(\boldsymbol{\lambda} \mid y) = \begin{cases} \ell(\boldsymbol{\lambda}) & \text{if } y < y^* \\ g(\boldsymbol{\lambda}) & \text{if } y \geq y^* \end{cases}$$

where $\ell(\boldsymbol{\lambda})$ is the density of "good" configurations, $g(\boldsymbol{\lambda})$ is the density of "bad" configurations, and y^* is the γ-quantile of observed losses. TPE then selects $\boldsymbol{\lambda}$ that maximizes $\ell(\boldsymbol{\lambda})/g(\boldsymbol{\lambda})$, which is proportional to EI. TPE natively handles conditional search spaces through its tree-structured factorization, making it particularly effective for neural architecture HPO [18].

7.4 SMAC

Sequential Model-based Algorithm Configuration (SMAC) [32] employs a Random Forest surrogate instead of a GP or TPE model, making it scalable to high-dimensional categorical and mixed-type spaces. SMAC has demonstrated strong performance on algorithm configuration benchmarks and is widely used in the AutoML literature.

8. Evolutionary and Metaheuristic Optimization

8.1 Genetic Algorithms

Genetic Algorithms (GA) encode hyperparameter configurations as chromosomes and apply selection, crossover, and mutation operators to evolve a population over generations. The fitness function is the validation performance:

$$\text{fitness}(\boldsymbol{\lambda}_i) = -\mathcal{L}_{\text{val}}\big(f_{\boldsymbol{\lambda}_i}, \mathcal{D}_{\text{val}}\big)$$

GAs are well-suited for mixed discrete-continuous search spaces and impose no assumptions on the objective landscape's smoothness. The primary limitation is the large number of objective evaluations required to maintain a sufficiently diverse population typically $O(P \cdot G)$ where P is population size and G is the number of generations [19]. Population-Based Training [22] can be viewed as a modern parallel GA variant that incorporates Lamarckian evolution (inheriting trained weights) for significant efficiency gains.

8.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) maintains a swarm of N particles, each with position $\boldsymbol{\lambda}_i$ and velocity $\mathbf{v}_i$. The update rule is:

$$\mathbf{v}_i^{(t+1)} = w\,\mathbf{v}_i^{(t)} + c_1 r_1\big(\mathbf{p}_i - \boldsymbol{\lambda}_i^{(t)}\big) + c_2 r_2\big(\mathbf{g} - \boldsymbol{\lambda}_i^{(t)}\big)$$
$$\boldsymbol{\lambda}_i^{(t+1)} = \boldsymbol{\lambda}_i^{(t)} + \mathbf{v}_i^{(t+1)}$$

where $\mathbf{p}_i$ is the particle's personal best, $\mathbf{g}$ is the global best, w is the inertia weight, and c_1, c_2 are cognitive and social coefficients. PSO has been applied to SVM hyperparameter tuning [21] and CNN architecture search, demonstrating competitive accuracy with lower computational overhead than GAs due to its gradient-free velocity mechanism.

8.3 Differential Evolution

Differential Evolution (DE) generates trial vectors by adding scaled difference vectors to population members:

$$\mathbf{u}_i = \boldsymbol{\lambda}_{r_1} + F\big(\boldsymbol{\lambda}_{r_2} - \boldsymbol{\lambda}_{r_3}\big)$$

followed by crossover with the target vector. DE is particularly effective for continuous hyperparameter spaces with strong inter-parameter correlations and has been applied to XGBoost and deep learning HPO with favorable convergence properties [33].

8.4 Ant Colony Optimization

Ant Colony Optimization (ACO) builds solutions probabilistically on a discretized pheromone graph. Its extension to continuous domains (ACOR) deposits pheromone on Gaussian kernels centered at previous solutions. ACO has found application in feature selection combined with HPO, and in configuring decision tree ensembles, where the discrete nature of tree parameters aligns well with ACO's graph-based formulation [34].

9. Gradient-Based and Learning-Based HPO

9.1 Differentiable Architecture Search (DARTS)

DARTS [25] reformulates the discrete architecture search problem as a continuous optimization by introducing architecture parameters $\boldsymbol{\alpha}$ that represent mixture weights over candidate operations:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o' \in \mathcal{O}} \exp\left(\alpha_{o'}^{(i,j)}\right)} \cdot o(x)$$

The architecture $\boldsymbol{\alpha}$ and model weights $\mathbf{w}$ are jointly optimized via bilevel optimization:

$$\min_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \quad \text{s.t.} \quad \mathbf{w}^*(\boldsymbol{\alpha}) = \text{argmin}_{\mathbf{w}} \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\alpha})$$

This allows gradient descent over architecture parameters, reducing NAS costs from thousands of GPU days to a few GPU days. However, DARTS is known to collapse toward parameter-free operations (skip connections) under certain conditions, a problem addressed by subsequent variants [35].

9.2 Implicit Differentiation Methods

Gradient-based HPO methods such as DrMAD and T1-T2 optimization [36] compute approximate gradients of validation loss with respect to hyperparameters by differentiating through the training process using implicit differentiation or truncated backpropagation. These methods are efficient for continuous hyperparameters but require careful treatment of numerical stability and are difficult to apply to discrete or categorical hyperparameter spaces.

10. Reinforcement Learning for HPO

RL-based HPO frames the search as a sequential decision-making problem. The NAS-RL framework of Zoph and Le [23] trains an RNN controller that outputs architecture tokens; each sampled architecture is trained to convergence, and its validation accuracy is used as a reward signal for REINFORCE:

$$\nabla_{\theta_c} J(\theta_c) = \mathbb{E}_{\boldsymbol{\lambda} \sim \pi_{\theta_c}}\left[R(\boldsymbol{\lambda})\, \nabla_{\theta_c} \log \pi_{\theta_c}(\boldsymbol{\lambda})\right]$$

This formulation is general but notorious for requiring tens of thousands of GPU hours for convergence. More efficient variants ENAS [24], which shares weights across architectures reduced cost by approximately $1000\times$.

Population-Based Training (PBT) [22] frames HPO as an online learning problem: a population of agents trains simultaneously, and those performing poorly are periodically replaced by perturbed copies of high-performing agents. PBT can discover hyperparameter schedules (not just fixed values) and has been applied successfully to reinforcement learning and generative model training.

11. Multi-Objective Hyperparameter Optimization

Real-world deployments frequently require simultaneous optimization of multiple, often conflicting objectives accuracy versus inference latency, model size versus robustness, training cost versus generalization. Multi-objective HPO seeks the Pareto-optimal front:

$$\mathcal{P}^* = \left\{\boldsymbol{\lambda} \in \mathcal{H} : \nexists\ \boldsymbol{\lambda}' \text{ s.t. } f_i(\boldsymbol{\lambda}') \leq f_i(\boldsymbol{\lambda})\ \forall i,\ \exists j : f_j(\boldsymbol{\lambda}') < f_j(\boldsymbol{\lambda})\right\}$$

Multi-objective evolutionary algorithms such as NSGA-II and MOEA/D are naturally suited to this problem and have been combined with Bayesian surrogates in frameworks such as PESMO [37] and MOBO (Multi-Objective Bayesian Optimization). Hardware-aware NAS methods including MNasNet and Once-for-All apply multi-objective optimization to jointly minimize inference latency on target hardware alongside accuracy [38]. Energy-aware HPO has emerged as a response to the growing computational footprint of large model training, penalizing high-FLOPs configurations even when they achieve marginally superior accuracy [30].

12. Neural Architecture Search and AutoML

12.1 Neural Architecture Search

NAS represents the logical extension of HPO to the complete architectural design problem. Early NAS work [23] required 800 GPUs running for 28 days to discover competitive image classification architectures. The subsequent development of one-shot methods wherein all candidate architectures share a single super network with shared weights dramatically reduced this cost. ENAS [24] achieved NAS in approximately 12 GPU hours, while DARTS [25] reduced search time to under 4 GPU hours on CIFAR-10.

Progressive NAS (PNAS) [39] employs a sequential model-based search that expands the architecture from simple to complex, using a learned predictor to rank candidate cells. Hardware-aware NAS has extended these methods to constrained deployment environments, discovering architectures specifically optimized for mobile, FPGA, and edge hardware.

12.2 AutoML Pipelines

AutoML formalizes the Combined Algorithm Selection and Hyperparameter Optimization (CASH) problem:

$$\mathbf{A}^*, \boldsymbol{\lambda}^* = \text{argmin}_{A^{(j)} \in \mathcal{A},\ \boldsymbol{\lambda} \in \boldsymbol{\Lambda}^{(j)}} \mathcal{L}_{\text{val}}\left(f_{\boldsymbol{\lambda}}^{(j)}, \mathcal{D}\right)$$

where $\mathcal{A}$ is the set of candidate algorithms and $\mathbf{\Lambda}^{(j)}$ is the hyperparameter space of algorithm $A^{(j)}$.

Auto-sklearn [28] solves the CASH problem using SMAC with meta-learning warm-starts and ensemble construction. It has achieved competitive performance across hundreds of benchmark datasets. Optuna [40] introduced define-by-run API semantics, enabling dynamic search spaces that adapt to the current trial's results. Ray Tune [41] provides a distributed HPO framework with support for Hyperband, PBT, and BO backends. SMAC3 provides a mature implementation of Random Forest-based Bayesian optimization with strong theoretical guarantees. Hyperopt implements TPE and is widely deployed in industrial ML pipelines.

13. HPO Tools and Frameworks

The ecosystem of HPO software has matured considerably. Table 1 synthesizes existing literature surveying these frameworks and their applications.

Table 1: Comparative Review of Existing Literature on HPO Techniques and Applications

Reference	Year	Method	Model(s) Tuned	Dataset(s)	Key Contribution
Bergstra & Bengio [14]	2012	Random Search vs Grid Search	DNNs	MNIST, CIFAR	Showed RS outperforms GS for high-dim HPO
Snoek et al. [3]	2012	GP-based BO (Spearmint)	DNNs, SVMs	MNIST, DBN	Formal BO for HPO; introduced EI acquisition
Bergstra et al. [17]	2013	TPE (Hyperopt)	DNNs	Multiple	Tree-structured Parzen estimator for conditional spaces
Hutter et al. [32]	2011	SMAC	SAT solvers, SVMs	Algorithm benchmarks	RF surrogate for CASH; meta-learning warm-start
Zoph & Le [23]	2017	NAS-RL	CNN	CIFAR-10, PTB	RL-based NAS, 800 GPU days
Liu et al. [25]	2019	DARTS	CNN, RNN	CIFAR-10, PTB	Differentiable architecture search
Pham et al. [24]	2018	ENAS	CNN	CIFAR-10	Weight sharing for 1000x speedup
Jaderberg et al. [22]	2017	PBT	DNNs	Multiple	Parallel online HPO with adaptation
Feurer et al. [28]	2015	Auto-sklearn	Multiple classifiers	OpenML	CASH + meta-learning + ensembling
Falkner et al. [42]	2018	BOHB	DNNs	Multiple	Bayesian BO + Hyperband multi-fidelity
Akiba et al. [40]	2019	Optuna	General ML	Multiple	Define-by-run API, pruning, TPE
Liaw et al. [41]	2018	Ray Tune	Deep learning	Multiple	Distributed HPO with Hyperband and PBT
White et al. [35]	2021	DARTS stabilization	CNNs	NAS benchmarks	Collapse prevention in differentiable NAS
Loshchilov & Hutter [20]	2016	CMA-ES for HPO	DNNs	CIFAR	Competitive evolution strategy for HPO
Real et al. [43]	2019	Aging Evolution	NAS	ImageNet	Tournament selection for scalable NAS

14. Comparative Analysis of HPO Techniques

14.1 Evaluation Framework

To provide a systematic comparison, we evaluate HPO methods across six dimensions: (1) search efficiency (quality of configurations found per evaluation), (2) computational cost (wall-clock time and resources required), (3) convergence speed (evaluations to reach near-optimal performance), (4) scalability (behavior in high-dimensional spaces), (5) robustness (sensitivity to noise and initialization), and (6) exploration-exploitation balance (ability to avoid premature convergence).

Table 2: Comparison of HPO Methods Across Key Evaluation Criteria

Method	Search Efficiency	Computational Cost	Convergence Speed	Scalability (High-dim)	Robustness	Exploration-Exploitation	Deep Learning Suitability

Grid Search	Very Low	Very High	Slow	Very Poor	High	Exploration only	Poor
Random Search	Low–Medium	Low	Moderate	Moderate	High	Exploration only	Moderate
Gaussian Process BO	High	Medium	Fast	Poor (d > 20)	High	Tunable (EI/UCB)	Moderate
TPE (Hyperopt)	High	Low–Medium	Fast	Good	Medium	EI-based	Good
SMAC (RF surrogate)	High	Medium	Fast	Good	Medium	EI-based	Good
Genetic Algorithm	Medium	High	Slow–Medium	Medium	Medium	Controllable	Moderate
PSO	Medium	Medium	Medium	Medium	Medium	Inertia-controlled	Moderate
Differential Evolution	Medium–High	Medium	Medium	Medium	High	F-controlled	Moderate
CMA-ES	High	Medium	Fast	Medium	High	Covariance-based	Good
DARTS	High	Low (GPU grad)	Very Fast	High	Low	Gradient bias	Excellent
NAS-RL	Very High	Very High	Very Slow	High	Low	Policy entropy	Excellent
PBT	High	Medium	Fast (online)	Medium	High	Population diversity	Excellent
BOHB	Very High	Medium	Fast	Good	High	BO + multi-fidelity	Excellent
Hyperband	Medium	Low	Fast	Good	High	Bandit arms	Very Good
ART-HPO	Medium–High	Low–Medium	Moderate	Good	High	Diversity-first exploration	Good

14.2 Performance on Specific Model Classes

SVM tuning involves a small number of continuous hyperparameters (C, γ, degree), making GP-based BO and TPE highly effective in fewer than 50 evaluations. Random search is competitive for binary kernels. GA-based methods have been applied to multi-kernel SVMs with success [21].

Random Forest and XGBoost tuning involve moderate-dimensional spaces (6–12 hyperparameters) with important categorical choices. SMAC and TPE excel here due to their handling of conditional and categorical variables. PBT has shown promise for XGBoost with learning rate warm-up schedules.

Deep Neural Networks present the most challenging HPO scenario. BOHB [42] is currently the strongest general-purpose method, combining the sample efficiency of Bayesian optimization with the computational savings of successive halving. PBT is preferred when training dynamics matter (e.g., cyclical learning rates). DARTS and ENAS are the methods of choice when the architecture itself is variable.

15. HPO Frameworks: Practical Comparison

Table 3: Advantages, Limitations, and Practical Applications of Major HPO Techniques

Technique / Framework	Core Algorithm	Key Advantages	Key Limitations	Best Applications
Grid Search	Exhaustive enumeration	Simple, reproducible, fully parallel	Exponential complexity, wasteful	Small search spaces, final grid refinement
Random Search	Uniform sampling	Simple, parallel, competitive baseline	No adaptation, sample-inefficient	First-pass exploration, high-dim spaces
Spearmint / GP-BO	GP + EI	Sample-efficient, principled uncertainty	$O(n^3)$ GP cost, poor scaling $d > 20$	Low-dim, expensive evaluations (e.g., SVM)
Hyperopt / TPE	Tree-structured Parzen	Conditional spaces, scalable, fast	No explicit exploration guarantee	DNNs, NLP, conditional pipelines
SMAC3	RF surrogate + EI	Mixed types, robust, strong CASH support	Higher overhead than TPE	Algorithm configuration, AutoML

Technique / Framework	Core Algorithm	Key Advantages	Key Limitations	Best Applications
Optuna	TPE + CMA-ES	Define-by-run, pruning, distributed	Newer, less theoretical backing	General ML, PyTorch, LightGBM
Ray Tune	Distributed BO/PBT	Scalable, multi-GPU, backend-agnostic	Complexity, infrastructure overhead	Large-scale deep learning
BOHB	BO + Hyperband	Best of BO + multi-fidelity, robust	Requires fidelity proxy	DNNs, NAS, resource-constrained settings
Auto-sklearn	SMAC + meta-learning	End-to-end pipeline, ensemble output	Slow on new datasets without meta-data	Tabular data AutoML
NAS-RL (NASNet)	RL controller	State-of-the-art architectures	800+ GPU days, impractical	Architecture discovery research
DARTS	Gradient-based NAS	Fast, differentiable, low cost	Operation collapse, instability	CNN/RNN architecture search
PBT	Evolutionary + RL	Online schedule learning, efficient	Non-reproducible, stochastic	Reinforcement learning, GAN training
PSO	Swarm intelligence	Derivative-free, conceptually simple	Premature convergence, many evaluations	Continuous HPO, ensemble tuning
Differential Evolution	Population + mutation	Robust, good for correlated parameters	Many evaluations, manual tuning F, CR	XGBoost, continuous spaces

16. Computational Complexity and Scalability Analysis

The computational complexity of HPO methods varies dramatically. Grid Search has complexity $\mathcal{O}(k^d)$ where k is the grid resolution per dimension and d is the number of hyperparameters—clearly intractable for modern neural networks. Random Search has $\mathcal{O}(N)$ complexity where N is the evaluation budget, making it trivially scalable. GP-based BO incurs $\mathcal{O}(n^3)$ cost per surrogate update due to GP regression, limiting it to $n \lesssim 1000$ observations in practice; sparse GP approximations and scalable kernels extend this range but introduce approximation error [5].

TPE scales as $\mathcal{O}(n\log n)$ for KDE-based density estimation, making it tractable for thousands of trials. SMAC with Random Forest surrogates has $\mathcal{O}(n\log n)$ training and $\mathcal{O}(T \cdot n\log n)$ prediction cost (where T is the number of trees), providing a good balance of scalability and accuracy [32].

Evolutionary methods scale as $\mathcal{O}(P \cdot G \cdot C_{\text{eval}})$ where C_{eval} is the cost of a single evaluation. Their primary bottleneck is the large number of expensive function evaluations required to evolve effective populations. Multi-fidelity methods such as Hyperband and BOHB address evaluation cost directly by allocating minimal resources to poor-performing configurations, achieving significant total speedups at the cost of fidelity assumptions.

In the context of distributed HPO, Ray Tune enables asynchronous parallel evaluation on multi-GPU clusters, effectively reducing wall-clock time by a factor approaching the number of parallel workers. Distributed BOHB extends this to the Bayesian setting through asynchronous surrogate updates [41].

17. Challenges and Limitations

17.1 High-Dimensional Search Spaces

Most HPO methods degrade in high-dimensional spaces. GP-based BO suffers from the curse of dimensionality in both the surrogate fitting and acquisition function optimization steps. Additive models of the objective where performance decomposes as a sum over subsets of hyperparameters can partially mitigate this, but the decomposition must be inferred from data [13]. Subspace selection methods that identify the effective low-dimensional manifold have shown promise but remain an active research area.

17.2 Noise and Non-Stationarity

Validation performance estimates are noisy due to random initialization, data shuffling, and mini-batch stochasticity. GP-based BO can model noise explicitly, but surrogate fitting under high noise requires careful regularization. Non-stationarity where the objective landscape changes as the optimization progresses (e.g., due to learning rate warmup) violates the stationary kernel assumptions of standard GPs [12].

17.3 Transfer and Meta-Learning

Warm-starting HPO using knowledge from related datasets can dramatically reduce the required evaluation budget. However, identifying the appropriate notion of dataset similarity and ensuring that transferred knowledge is helpful rather than misleading remains non-trivial [28]. Task2Vec and dataset2vec representations have been proposed as meta-feature encoders, but their generalization across domain shifts is not well characterized.

17.4 Reproducibility and Fairness in Comparison

The HPO literature suffers from inconsistent evaluation protocols: different papers use different budgets, hardware, and validation methodologies, making direct

comparisons unreliable. HPOBench [44] provides a standardized benchmarking suite with tabular results from pre-computed grids, enabling fair comparison under identical computational budgets.

17.5 Federated and Privacy-Preserving HPO

Federated learning introduces additional complexity to HPO: the hyperparameter landscape depends on the data distribution across clients, which is heterogeneous and unavailable to a central server. Federated HPO must navigate the communication overhead of transmitting model performance results alongside gradients, and must handle non-IID data distributions that make standard surrogate models unreliable [45].

18. Future Research Directions

18.1 Explainable HPO

The field of **explainable HPO** seeks to quantify the contribution of individual hyperparameters to model performance through tools such as functional ANOVA decomposition [46] and fANOVA-based sensitivity analysis. Understanding which hyperparameters matter most and under what conditions enables practitioners to prune the search space intelligently and builds trust in the optimization process. This direction intersects with the broader push for interpretable AI.

18.2 HPO for Large Language Models

Training large language models (LLMs) such as GPT-4, LLaMA, and Mistral involves hyperparameter choices learning rate schedules, warmup steps, batch size, gradient clipping at a scale where each evaluation costs millions of GPU hours. Standard HPO methods are entirely inapplicable at this scale. **Scaling laws** [47] empirical power-law relationships between model size, compute, data volume, and loss partially address this by enabling extrapolation from small-scale experiments. However, the interaction between architectural choices and optimization hyperparameters at frontier scale remains poorly understood and represents a high-impact open problem.

18.3 Green AI and Sustainable HPO

The environmental cost of HPO is substantial: searching over large spaces with expensive evaluations contributes significantly to the carbon footprint of ML research [30]. Carbon-aware HPO frameworks that incorporate energy consumption as an explicit optimization objective, schedule evaluations during low-carbon-intensity periods, and apply aggressive early stopping represent an important emerging research direction. BOHB and multi-fidelity methods represent early steps in this direction, but energy-aware acquisition functions and hardware-specific efficiency modeling remain open problems.

18.4 HPO for Edge and Federated AI

Deploying ML models on edge devices imposes strict constraints on model size, latency, and energy. Hardware-aware NAS and multi-objective HPO methods that incorporate device-specific benchmarks are essential for this domain. Federated HPO where hyperparameter search is conducted across distributed, privacy-sensitive data requires communication-efficient protocols and robustness to client heterogeneity and dropout [45].

18.5 Meta-Learning and Warm-Starting

Meta-learning for HPO leverages performance data from previous tasks to initialize surrogate models with informed priors, dramatically accelerating search on new tasks. The RGPE (Ranking-Weighted Gaussian Process Ensemble) model [48] combines predictions from task-specific GPs weighted by their performance ranking, providing a principled warm-start. Learning to optimize (L2O) frameworks train the HPO algorithm itself as a recurrent network across a distribution of tasks, potentially enabling amortized HPO at inference time.

18.6 Zero-Shot and Few-Shot HPO

For practitioners with very tight evaluation budgets, zero-shot HPO selecting a single configuration based on meta-features without any evaluations is an emerging paradigm. Portfolio methods that maintain a fixed set of configurations with strong average performance across a task distribution represent one approach; meta-learned default configurations derived from extensive offline search represent another.

CONCLUSION

This review has provided a comprehensive comparative analysis of hyperparameter optimization techniques spanning traditional exhaustive methods, probabilistic Bayesian frameworks, evolutionary and metaheuristic approaches, gradient-based architecture search, reinforcement learning controllers, multi-objective optimization, and AutoML pipelines. The analysis reveals several overarching conclusions.

First, no single HPO method universally dominates. The appropriate method depends critically on the dimensionality of the search space, the cost of objective evaluation, the nature of the hyperparameter types (continuous, integer, categorical, conditional), and the computational budget available. For low-dimensional, expensive evaluations, GP-based Bayesian optimization remains the gold standard. For high-dimensional, moderate-cost settings, TPE and SMAC provide an effective balance of scalability and efficiency. For deep learning at scale, BOHB and PBT represent the state of the practice.

Second, multi-fidelity methods represent the most important practical advance of the past decade. By intelligently allocating evaluation budgets based on intermediate performance, methods like Hyperband and BOHB achieve order-of-magnitude speedups over single-fidelity approaches without sacrificing solution quality.

Third, the AutoML paradigm has democratized HPO by packaging sophisticated optimization within end-to-end pipelines that non-experts can apply directly. However, the black-box nature of AutoML reduces interpretability and complicates debugging when pipelines fail.

Fourth, critical open problems remain at the frontier of HPO research: scaling to LLMs, federated and privacy-preserving settings, energy-aware optimization, and meta-learning for rapid adaptation to new tasks. These directions will define the trajectory of the field over the coming decade.

As machine learning continues its expansion into high-stakes domains and resource-constrained environments, the importance of principled, efficient, and interpretable HPO will only grow. This review aims to serve as a navigational reference for researchers and practitioners entering or advancing within this foundational area of the machine learning ecosystem.

REFERENCES

1. J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proc. 30th Int. Conf. Mach. Learn. (ICML)*, Atlanta, GA, 2013, pp. 115–123.
2. F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., *Automated Machine Learning: Methods, Systems, Challenges*. Cham, Switzerland: Springer, 2019.
3. J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012, pp. 2951–2959.
4. T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.
5. P. I. Frazier, "A tutorial on Bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.
6. R. Miikkulainen et al., "Evolving deep neural networks," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, R. Kozma et al., Eds. Academic Press, 2019, pp. 293–312.
7. L. He, Q. Ao, L. Shi, and J. Lin, "AutoML: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.
8. L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
9. T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, San Francisco, CA, 2016, pp. 785–794.
10. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
11. L. Probst, B. Bischl, and A. Boulesteix, "Tunability: Importance of hyperparameters of machine learning algorithms," *J. Mach. Learn. Res.*, vol. 20, no. 53, pp. 1–32, 2019.
12. M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated Machine Learning*. Cham: Springer, 2019, pp. 3–33.
13. Z. Li, E. Andreeto, M. A. Ranzato, and P. Perona, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *J. Mach. Learn. Res.*, vol. 18, no. 185, pp. 1–52, 2018.
14. J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.
15. J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani, "Predictive entropy search for efficient global optimization of black-box functions," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 27, 2014, pp. 918–926.
16. Thapa, P., Sharma, P., Khadka, S., K C, S., Yadav, A. C., & Gupta, S. (2026). *ART-HPO: Adaptive random testing for efficient hyperparameter optimization*. In *Artificial Intelligence and Machine Learning* (Proceedings of the Third International Artificial Intelligence Conference, IAIC 2026, Singapore, February 6–9, 2026). Springer Nature Singapore.
17. J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 24, 2011, pp. 2546–2554.
18. J. Bergstra, B. Kégl, and Y. Bengio, "Hyperopt: A Python library for optimizing the hyperparameters of machine learning algorithms," in *Proc. 12th Python in Science Conf.*, 2013, pp. 13–20.
19. X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, no. 9, pp. 1423–1447, Sep. 1999.
20. I. Loshchilov and F. Hutter, "CMA-ES for hyperparameter optimization of deep neural networks," in *Proc. Int. Conf. Learn. Representations (ICLR) Workshop*, 2016.
21. M. Martínez-Estudillo, C. Hervás-Martínez, P. A. Gutiérrez, and F. J. Martínez-Estudillo, "Evolutionary product unit based neural networks for regression," *Neural Networks*, vol. 19, no. 4, pp. 477–486, 2006.
22. M. Jaderberg et al., "Population based training of neural networks," *arXiv preprint arXiv:1711.09846*, 2017.
23. B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2017.
24. H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search

via parameter sharing," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 4095–4104.
25. H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2019.
26. G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. V. Le, "Understanding and simplifying one-shot architecture search," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 549–558.
27. L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA," *J. Mach. Learn. Res.*, vol. 18, no. 25, pp. 1–5, 2017.
28. M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015, pp. 2962–2970.
29. X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, Jan. 2021.
30. E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proc. 57th Annu. Meeting Assoc. Computational Linguistics (ACL)*, Florence, Italy, 2019, pp. 3645–3650.
31. K. Daulton, M. Balandat, and E. Bakshy, "Parallel Bayesian optimization of multiple noisy objectives with expected hypervolume improvement," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021.
32. F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Proc. 5th Int. Conf. Learning and Intelligent Optimization (LION 5)*, 2011, pp. 507–523.
33. R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.
34. M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.
35. C. White, M. Safari, R. Sukthanker, B. Ru, T. Elsken, A. Zela, D. Dey, and F. Hutter, "Neural architecture search: Insights from 1000 papers," *arXiv preprint arXiv:2301.08727*, 2023.
36. Y. Franceschi, M. Donini, P. Frasconi, and M. Pontil, "Forward and reverse gradient-based hyperparameter optimization," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 1165–1173.
37. D. Hernández-Lobato, J. Hernández-Lobato, A. Shah, and R. Adams, "Predictive entropy search for multi-objective Bayesian optimization," in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 1492–1501.
38. M. Tan et al., "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE Conf. Computer Vision Pattern Recognition (CVPR)*, 2019, pp. 2820–2828.
39. C. Liu et al., "Progressive neural architecture search," in *Proc. European Conf. Computer Vision (ECCV)*, 2018, pp. 19–34.
40. T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, 2019, pp. 2623–2631.
41. R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," in *Proc. ICML AutoML Workshop*, 2018.
42. S. Falkner, A. Klein, and F. Hutter, "BOHB: Robust and efficient hyperparameter optimization at scale," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1437–1446.
43. E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artificial Intelligence*, vol. 33, 2019, pp. 4780–4789.
44. K. Eggensperger et al., "HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO," in *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.
45. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 1273–1282.
46. F. Hutter, H. Hoos, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in *Proc. 31st Int. Conf. Mach. Learn. (ICML)*, 2014, pp. 754–762.
47. J. Hoffmann et al., "Training compute-optimal large language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022.
48. M. Feurer, B. Letham, and E. Bakshy, "Scalable meta-learning for Bayesian optimization using ranking-weighted Gaussian process ensembles," in *Proc. AutoML Workshop at ICML*, 2018.
49. Z. Wang and S. Jegelka, "Max-value entropy search for efficient Bayesian optimization," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 3627–3635.

