Bima Journal of Science and Technology, Vol. 8(1A) Mar, 2024 ISSN: 2536-6041

DOI: 10.56892/bima.v8i1.603

Streamlining the Path from Data to Deployment: Intelligent Methods for Hyperparameter Tuning in Machine Learning

Bakare K. A.*, Abubakar S. I., Naveen A. Y., Abdullahi A. J., Gaku M. S., Abdulganiyu I., Asmau U. and Ahmad S.

[1]Department of Computer Science and Information Technology, Faculty of Computing and Artificial Intelligence, Federal University Dutsin-ma, Dutsin-ma, Katsina State, Nigeria

Corresponding Author: asibrahim@fudutsinma.edu.ng

ABSTRACT

This study addresses the essential role of hyperparameter optimization in intricate machine learning models, particularly in image classification tasks. With manual tuning impractical in the face of escalating complexities, the research thoroughly evaluates eight automated optimization methods: grid search, random search, Gaussian process Bayesian optimization (BO), Tree Parzen estimator BO, Hyperband, BO/Hyperband hybrid, genetic algorithms, and particle swarm optimization. Assessments cover diverse model architectures and performance metrics, considering accuracy, mean squared error, and optimization time. Grid search proves exhaustive but time-prohibitive, random search is sensitive to seed values, Gaussian process BO excels in low-dimensional spaces, and Tree Parzen estimator BO is efficient in higher dimensions. Hyperband prioritizes time efficiency, genetic algorithms pose parallelization challenges, and particle swarm optimization excels with optimal accuracy and efficiency. Distinct advantages emerge based on model architecture and search space complexity, highlighting the need for tailored optimizers in specific machine learning applications. Comprehensive benchmarks provide valuable guidance, with future work recommended to extend evaluations to emerging model classes, particularly deep neural networks.

Keywords: Hyper-parameter optimization, machine learning, Bayesian optimization, particle swarm optimization, genetic algorithm, grid search.

INTRODUCTION

The past decade has witnessed explosive growth in applications of machine learning across high-impact domains including healthcare, transportation, finance, education, sustainability, and more (Jordan & Mitchell, 2015). Availability of vast data sources, combined with algorithmic breakthroughs like deep neural networks, have fueled significant interest and investment into data-driven decision making via predictive modeling (LeCun et al., 2015). However, realizing accurate and reliable machine learning-based technologies hinges upon finding suitable configurations prior to model training - settings dictating architectural complexities, optimizations, assumptions, and more (Claesen & De Moor, 2015). Mastering this process of hyperparameter tuning remains imperative yet persistently challenging for streamlining from data to deployments. In machine learning, model hyperparameters refer to parameters set prior to commencing training through an examination of data-driven signals and metrics. This contrasts with optimized model parameters updated throughout the fitting process itself on training datasets. Hyperparameters control fundamental properties like model family, depth/width tradeoffs, regularization terms, step sizes, batch sizes, and data preprocessing (Goodfellow et al., 2016).

Different settings lead to vastly different inductive biases, flexibilities to fit patterns, and tendencies to overfit or underfit (Domingos, 2012). High-performing configurations allow rich expressiveness to capture trends while preventing overcomplexity through constraints and regularization.

However, enumerated search over all meaningful hyperparameter value combinations grows exponentially, becoming intractable even for simple modern model families (Bergstra & Bengio 2012). Basic grid search exams scattered points, while random search samples more efficiently (Bergstra & Bengio, 2012). Both still struggle with high dimensions and lack sequential focus. More informed methods like Bayesian Optimization leverage observed data to guide sampling toward promising areas through heuristics capturing underlying response surface geometry (Shahriari et al., 2015). Evolutionary methods take inspiration from genetics, iteratively mutating and combining high-performing configurations (Jaderberg et al., 2017). Gradient-based approaches backpropagate validation losses with respect to hyperparameters themselves, relying on smoothness (Luketina et al., 2016). While conceptually simple as searching over settings, underlying response surfaces tend to demonstrate highly non-convex loss profiles between hyperparameter vectors and model metrics across training set samples. Discontinuities, isolated optima, flat dense regions, and nonlinear interactions abound, conflicting with assumptions in optimization approaches (Yu & Zhu, 2020). Data scarcity further exacerbates difficulties in distant extrapolation and noise within finite measurement budgets. Search must balance exploration against resource efficiency, resembling the classic explore-exploit tradeoff from reinforcement learning under uncertainty (Jomanto et al., 2021). With finite computational resources, optimal stopping also factors in (Domhan et al., 2015).

Moreover, conditional dependencies between hyperparameters vary by model family and problem complexity, limiting transferability of insights (Klein et al., 2017). Settings like regularization strength intrinsically relate to capacity factors like network width and depth (Jaderberg et al., 2017). Signals must propagate across these interlinked choices to reflect downstream interactions (Luketina et al., 2016). Capturing useful geometric priors with low-dimensional embeddings for efficient optimization remains an open challenge (Wistuba et al., 2015). Guiding adaptive, hierarchical, and evolutionary approaches via learned metadata still proves difficult (Lindauer & Hutter, 2019). Critically, while hyperparameter tuning focuses on maximizing validation performance, additional desiderata around efficiency, responsiveness, and human oversight exist when operationalizing pipelines (Kohavi & Wolpert, 1996). Refining coherence and standardization around managing the end-to-end machine learning life cycle thus persist as open problems alongside tighter optimization itself.

Despite difficulties, properly tuning hyperparameters remains crucially impactful. Suboptimal configurations markedly degrade predictive accuracy and reliability from otherwise state-of-the-art models across areas like computer vision, natural language processing, personalized medicine, risk assessment, and beyond (Melis et al., 2018). Increased model flexibility from poor settings risks overfitting and hurts generalizability (Domingos, 2012). Conversely, insufficiently expressive models underfit the signal. Manual tuning proves inadequate even for domain experts. While automated optimization introduces additional method configuration considerations, enhanced approaches accelerate cycle times and free practitioner bandwidth (Claesen & De Moor, 2015). With increasingly ambitious applications of deep learning gaining traction, model and data complexity continue swelling. State-of-the-art neural networks now routinely contain billions of parameters, while advances in computational resources and data storage expand capacities to fit such gigantic models (Brown et al., 2020). Surging dimensionalities overwhelm even recent hyperparameter search regimes. Furthermore, cascading societal adoption of machine learning fuels demand for streamlined paths from procuring training data to production deployment. As such, advancing the coherence and efficiency of methodologies for navigating these high-dimensional heterogeneous configuration spaces remains imperative. Guiding hyperparameter tuning requires addressing high-level optimization challenges under uncertainty with constraints, building on several major subfields within machine learning itself. First, efficiently balancing exploration against exploitation under scarce measurement budgets connects tightly to the literature on reinforcement learning, optimal experimental design, active learning, and beyond (Jomanto et al., 2021; Shahriari et al., 2014). Approaches to accelerate search via warm starts, multi-fidelity inferences, and metadata typically leverage transfer learning principles as well (Falkner et al., 2018). Modeling losses as complex response surfaces relates to Gaussian process regression and gradient-based neural architecture search efforts (Jin et al., 2019). The subfield of automated machine learning specifically focuses on automatically customizing pipelines around given data and use cases through hierarchical configuration (Hutter et al., 2018). Lastly, quantifying the value of additional search invokes the theory around optimal early stopping in allocation problems (Domhan et al., 2015). Safely guiding practitioners on satisfying starting configurations before enabling further tuning requires predicting generalizability, invoking classical machine learning bias-variance decompositions (Kohavi & Wolpert, 1996). Warm-start transfer with multitask metadata shows particular promise by harnessing learnings across problems (Wistuba et al., 2015). Highly parameterized search spaces benefit from transformable embeddings (Alet et al., 2018), while conditional structure reduces dimensionality through parameter tying (Kandasamy et al., 2018). Modeling cost versus accuracy tradeoffs via multi-fidelity inference directs samples for cheap approximations before converging refinements (Falkner et al., 2018). Expanding tooling access through cloud APIs and deployment monitoring aids uptake (Golovin et al., 2017). Together, these innovations incrementally streamline the path from data to deployment.

LITERATURE REVIEW

Theoretical Background

Machine learning has transformed capabilities for pattern recognition and predictive modeling across applications from personalized recommendations to precision medicine (Jordan & Mitchell, 2015). However, real-world deployment of accurate and reliable machine learning pipelines hinges on configuring suitable hyperparameter values - settings that govern model complexity, training processes, and data assumptions prior to seeing the data (Claesen & De Moor, 2015; Goodfellow et al., 2016). Selecting appropriate hyperparameter configurations allows models to capitalize on expressive power without overfitting, balancing under- and over-complexities through empirical validation. Unfortunately, combinatorial growth of hyperparameter search spaces renders comprehensive exploration practically impossible (Bergstra & Bengio, 2012), necessitating principles and approximations. Mastering the accompanying theoretical and optimization challenges remains imperative for streamlining machine learning.

Mathematical Optimization

Mathematical optimization involves finding the best solution or configurations of decision variables that maximize or minimize an objective function, while satisfying any constraints.

An unconstrained optimization problem has the form:

$$minimize\ f(x)$$

where f(x) is the objective function to be minimized and x is the decision variable that can take any real value. For a constrained optimization problem, there are typically inequality constraints g_i(x) ≤ 0 and equality constraints h_j(x) = 0 that limit the feasible values of x. The full formulation is:

$$\begin{gathered} minimize\ f(x) \\ subject\ to\ g\_i(x) \ \le\ 0, i\ =\ 1, 2, \ldots, m \\ h\_j(x)\ =\ 0, j\ =\ 1, 2, \ldots, p \\ x\ \in\ X \end{gathered}$$

Where m is the number of inequality constraints and p is the number of equality constraints. X defines the feasible domain of x. The inequality and equality constraints define a feasible region D where the constraints are satisfied:

$$D\ =\ \{x\ \in\ X\ |\ g\_i(x)\ \le\ 0, h\_j(x)\ =\ 0\}$$

A global minimum x* has the property that f(x*) ≤ f(x) for all x in the feasible region D. A local minimum x* satisfies f(x*) ≤ f(x) only for x in some neighborhood N around x* such that N ∩ D. For a convex optimization problem, the objective function f(x) is convex, meaning:

$$f(tx\_1\ +\ (1-t)x\_2) \le\ tf(x\_1)\ +\ (1-t)f(x\_2)$$

for all x_1, x_2 in X and t in [0,1]. The feasible region C must also be a convex set. Convex functions have only one global minimum, so gradient-based methods like gradient descent can find the optimal x* by following the negative gradient direction from any starting point. For nonconvex optimization, functions can have multiple local minima, so gradient-based methods may only find a local rather than global minimum. Many machine learning problems are nonconvex. Global optimization methods like heuristics must be used to increase the chance of finding the global minimum.

Theoretical Landscape

At its core, hyperparameter optimization seeks good settings based on a validation metric without explicit knowledge of model performance across all possible configurations. This faces difficulties similar to reinforcement learning explore-exploit dilemmas, with deeply stochastic objective functions as target problems and models change drastically across hyper-parameterizations (Jomanto et al., 2021). Certain broad heuristics exist, like starting simpler or enabling only necessary complexity. However, hyperparameters often exhibit complex inter-dependencies and nonlinear impacts, frustrating human intuitions (Hutter et al., 2018). Grounded theoretical guidance for navigating these high-dimensional heterogeneous search spaces remains lacking, though basic insights on sensitivity, curvature, and constraint satisfaction inform local optimization approaches (Benbouzid et al., 2012).

Many state-of-the-art hyperparameter optimization regimes thus focus on efficient exploration guided by observed empirical performance rather than formal understanding of hyperparameter properties. Still, surging model and data complexities increasingly strain even approximation or sampling-based methods, motivating the fusion of functional analytics, incremental learning, and reasoning under uncertainty (Jomanto et al., 2021; Komer et al., 2014). Recent techniques leverage cheap approximations, transfer learning, warm starts, metadata-based priors, conditional parameterizations, and multi-fidelity evaluations to streamline hyperparameter tuning amidst ballooning search spaces (Falkner et al., 2018). Theoretical progress charting and bounding optimization difficulty over important hyperparameter classes could unlock further acceleration.

Open Problems

As machine learning broadens in reach and scope, several open problems around managing the end-to-end life cycle persist. Guiding when simpler models suffice could curb wasted tuning efforts, while detecting when additional tuning is worthwhile remains challenging (Kohavi & Wolpert, 1996). Safe human oversight of automated parameter recommendations also introduces interfaces needing standardization. On the theory front, commonly relied upon assumptions of smoothness rarely hold over full hyperparameter ranges, confusing gradient-based methods, while useful meta-learning features remain inadequately cataloged across problem verticals (Rangapuram et al., 2018). Finally, quantifying and bounding the diminishing returns of hyperparameter tuning resources could help automatically balance exploration, accuracy, and delays in recommending production deployments (Domhan et al., 2015). Advances over these open problems promise to streamline paths from data to decision by bringing coherence and acceleration to the persistently ad hoc hyperparameter optimization processes underlying much of modern and practical artificial intelligence.

Review of Related Literature

The authors in (Hertel et al., 2020) developed an open-source tool called Sherpa for automating hyperparameter optimization of machine learning models. The method adopted involves implementing a variety of optimization algorithms like Bayesian optimization and bandit-based techniques along with visualization tools to monitor runs. Results demonstrate improved performance by tuning neural networks on MNIST and a climate modeling deep network. The software has seen wide adoption across domains but currently lacks support for distributed tuning across multiple nodes to further scale. Sherpa effectively tackles the tedious task of model tuning while allowing insight into the process, though expanding its distributed capabilities would extend its impact even more. With its combination of algorithms and intuitive analysis tools, Sherpa makes progress on the challenge of accessible and efficient hyperparameter optimization.

The authors in (Schratz et al. 2019) compares several statistical and machine learning models for predicting the spatial distribution of a tree pathogen, finding random forest to have the best performance. A key result is that non-spatial cross-validation leads to overoptimistic, biased model evaluation. The authors analyze spatial vs. non-spatial tuning, showing small differences in most cases. The study demonstrates the need to account for spatial autocorrelation in ecological predictive modeling through proper resampling techniques to avoid poor decision making. More validation on diverse spatial ecology data could identify broader patterns.

The authors in (Wu et al., 2019) presents a method for optimizing hyperparameters of machine learning models using Bayesian optimization with Gaussian processes (GP). The authors demonstrate how Bayesian optimization is well-suited for tuning black-box functions, iteratively updating a posterior estimate to determine the next best sample point. Experiments apply Bayesian hyperparameter optimization to random forest, convolutional neural networks, recurrent neural networks, and multi-grained cascade forest, showing improved model accuracy over default parameters and faster convergence than grid search. The paper provides a novel application of Bayesian optimization, leveraging GP assumptions to efficiently search high-dimensional hyperparameter spaces of complex models. Results on benchmark datasets validate Bayesian optimization as an effective approach for model tuning without requiring gradient information or functional forms.

The authors in (Pannakkong et al., 2022) applies response surface methodology (RSM) to hyperparameter tuning of three machine learning algorithms - artificial neural networks, support vector machines, and deep belief networks. The goal is to show RSM can efficiently tune hyperparameters while maintaining model performance compared to the commonly used grid search. Using an industrial dataset for quality prediction, RSM achieved similar prediction accuracy to grid search for the algorithms, while requiring 97.79%, 97.81%, and 80.69% fewer experimental runs for tuning. RSM also gave more reliable hyperparameter settings based on confirmation runs, with 90% and 100% reliability for ANN and DBN versus 80% for grid search. RSM required significantly fewer runs than grid search to achieve comparable model performance and prediction accuracy, demonstrating it is an efficient hyperparameter tuning method for machine learning regression algorithms with numerical hyperparameters and responses.

The authors in (Raji et al., 2022) proposes a simple deterministic selection genetic algorithm (SDSGA) for hyperparameter tuning of machine learning models. The SDSGA modifies the selection mechanism in a genetic algorithm to promote exploitative search by deterministically selecting the top fit individuals as parents. The SDSGA is compared to Bayesian optimization and other metaheuristic algorithms like GA, PSO, and BBO on benchmark functions and two machine learning models - convolutional neural networks and random forests. Results show the SDSGA converges faster and achieves higher accuracy than the other methods in most cases, indicating it is an effective approach for hyperparameter optimization when fitness evaluations are limited. The key finding is that the proposed SDSGA strikes a balance between exploration and exploitation which allows efficient hyperparameter tuning.

The authors in (Elgeldawi et al., 2021) provides a comprehensive comparative analysis of hyperparameter tuning techniques for machine learning algorithms applied to Arabic sentiment analysis. The authors tune the hyperparameters of six classifiers using Grid Search, Random

Search, Bayesian Optimization, Particle Swarm Optimization, and Genetic Algorithms on a dataset of 7000 Arabic hotel reviews. Without tuning, Support Vector Classifier performs best at 95.12% accuracy. With Bayesian Optimization tuning, Support Vector Classifier achieves the highest accuracy of 95.62%. The results demonstrate that hyperparameter tuning can substantially improve machine learning model performance on sentiment analysis for the morphologically complex Arabic language.

The authors in (Fan et al., 2022) propose a new hyperparameter optimization algorithm called Hybrid Sparrow Search Algorithm (HSSA) that combines the global search ability of Sparrow Search Algorithm (SSA) and the speed of Particle Swarm Optimization (PSO) to effectively optimize hyperparameters of deep neural networks. Experiments on convolutional neural networks demonstrate that HSSA finds superior solutions compared to other methods like Bayesian optimization and random search. The results indicate that HSSA has strong global search capability, excellent optimization performance, and stability for tuning hyperparameters in both simple and complex deep learning models. HSSA provides a new heuristic approach with the ability to avoid local optima for the challenging problem of hyperparameter optimization in deep learning.

The authors in (Ali et al., 2023) proposed many techniques for optimizing hyperparameters. The authors evaluate Ant Colony Optimization (ACO), Genetic Algorithm (GA), Whale Optimization Algorithm (WOA), and Particle Swarm Optimization (PSO) for tuning SVM hyperparameters, on diabetes and heart disease datasets. The key results are: GA-SVM achieved highest accuracy of 98.9% on diabetes data and 97.8% on heart data, with lowest computational time. WOA-SVM performed worst with only 71% accuracy but took most time. ACO-SVM and PSO-SVM achieved 80% accuracy. This comprehensive study found GA to be the most efficient optimization algorithm for SVM hyperparameter tuning, achieving high accuracy with low computational cost.

The authors in (Tani et al., 2021) explores using evolutionary algorithms like particle swarm optimization (PSO) and genetic algorithms (GA) to optimize hyperparameters for machine learning models in high energy physics. Both methods were very effective at minimizing the difficult Rosenbrock function, significantly outperforming gradient descent, grid search, and random guessing. When applied to a Higgs boson detection challenge, optimizing hyperparameters with PSO and GA improved model sensitivity by 12-13% over default parameters. On the function problem, PSO achieved lower error while on the physics challenge GA found slightly better hyperparameters. The methods demonstrated great promise for automatic hyperparameter optimization, removing the need for manual tuning while boosting model performance. The similar performance of PSO and GA shows both evolutionary techniques can powerfully optimize hyperparameters across problems.

The authors in (Morales-Hernández et al., 2022) categorizes recent multi-objective hyperparameter optimization algorithms into metaheuristic-based, metamodel-based, and hybrid approaches. It finds that metaheuristic algorithms like NSGA-II directly search the space but require many expensive evaluations. Metamodel methods like Bayesian Optimization use a surrogate model to guide search more efficiently and hybrids combine both. Objectives considered include error metrics, model complexity/size, training time, hardware metrics, and diversity. The paper recommends future work using more hybrids, multi-fidelity approaches, better noise handling, and more algorithm details and benchmarks. Overall, it offers a thorough review of multi-objective hyperparameter optimization methods and objectives, highlights limitations, and suggests promising research directions.

METHODOLOGY

This section employed for hyperparameter optimization (HPO) in machine learning models. Two distinct approaches are explored: Bayesian Optimization and Metaheuristic Algorithms. Bayesian Optimization leverages probabilistic surrogates, such as Gaussian processes, while Metaheuristic Algorithms, exemplified by Genetic Algorithms and Particle Swarm Optimization, employ guided stochastic search in the hyperparameter space.

HPO Algorithms

Bayesian Optimization

Bayesian optimization models the objective function f(x) using a probabilistic surrogate, such as a Gaussian process:

$$f(x) \sim GP(\mu(x), \sigma^2(x))$$

Where μ(x) and σ^2(x) are the mean and variance predictions at x. It uses an acquisition function α(x) based on the model to select the next sample point that balances exploration and exploitation. Common acquisitions functions include expected improvement, Gaussian process upper confidence bound, and Thompson sampling. The overall iterative process is:

1. Build probabilistic surrogate model of f(x)
2. Use acquisition function to determine next x to evaluate
3. Evaluate f(x) and add result to observed data D
4. Update surrogate model with new (x, f(x)) pair
5. Repeat steps 2-4

Gaussian process models have O(n^3) complexity. Other Bayesian models like random forests reduce this to O(nlogn).

Metaheuristic Algorithms

Algorithms like genetic algorithms (GA) and particle swarm optimization (PSO) perform guided stochastic search through the hyperparameter space:

1. Genetic Algorithms

Genetic algorithms are based on the process of natural selection. They maintain a population of candidate solutions, where each solution is encoded as a "chromosome" (typically a binary string). The chromosomes are evaluated on the machine learning task and assigned a fitness score. The highest scoring chromosomes are selected to "reproduce" through crossover and mutation to create the next generation. This mimics biological evolution. Mathematically,

Initialization

$$P^{(0)} = \{{x_1}^{(0)}, {x_2}^{(0)}, \dots {x_N}^{(0)}\}$$

Selection

Select parents: x_a, x_b based on the fitness

Crossover

$$x_{offspring} = Crossover(x_a, x_b)$$

Mutation

$$x_{mutated} = Mutation(x_{ofspring})$$

Replacement

$$P^{(t+1)} = \{{x_1}^{(t+1)}, {x_2}^{(t+1)} \dots {x_N}^{(t+1)}\}$$

Termination

Repeat steps 2-5 until a termination criterion is met.

DOI: 10.56892/bima.v8i1.603

2. Particle Swarm Optimization

Particle swarm optimization is based on swarm intelligence and operates by having many "particles" move around the hyperparameters search space looking for an optimal solution. Each particle has a position and velocity. Particles evaluate the machine learning model at their current position and communicate to share the best location found. Mathematically, the position x and velocity v of each particle are updated as:

$$v(t+1) = w * v(t) + c_1 * r_1 * (p_{best} - x(t)) + c_2 * r_2 * (g_{best} - x(t))$$

$$x(t+1) = x(t) + v(t+1)$$

Where p_{best} is the particle's personal best position and g_{best} is the swarm's global best. r_1 and r_2 are random numbers, while w, c_1, and c_2 are tuning parameters. This update rule enables particles to explore the space while being pulled towards better solutions over time. The goal is for the swarm to converge on the optimal set of hyperparameters by cooperatively searching.

Both methods harness population-based stochastic optimization, but genetic algorithms use evolutionary concepts while particle swarm employs swarm intelligence. The mathematical foundations capture these different mechanisms. Both can optimize complex high-dimensional spaces in few iterations and enable parallel implementations. However, GA has complexity $O(n^2)$ while PSO is $O(nlogn)$.

Hyper-parameters in Machine Learning Models

Machine learning model performance relies heavily on hyperparameter tuning, but manual tuning becomes infeasible as model and data complexity increase. This motivates developing automated hyperparameter optimization techniques. Selecting optimal hyperparameters is critical for machine learning but poses a challenging high-dimensional optimization problem. Various hyperparameter optimization methods have emerged to make this process more efficient and effective. The predictive power of machine learning is highly dependent on model hyperparameters, which are often tuned through an expensive trial-and-error process. Hyperparameter optimization research aims to automate and streamline this process to improve model quality and efficiency.

K-Nearest Neighbors (KNN)

KNN classifies a sample x based on the majority class of its k nearest neighbors in the training set. The predicted class ŷ is:

$$\hat{y} = argmax_{cj} \sum_{xi \in N_k(x)} I(y_i = y_j)$$

where $I(x)$ is an indicator function equaling 1 if $yi = cj$ and 0 otherwise. $Nk(x)$ are the k nearest neighbors to x. The key hyperparameter is k, the number of neighbors to consider. Larger k reduces noise but makes decision boundaries less distinct.

Support Vector Machines (SVM)

SVM computes an optimal hyperplane $w{\cdot}x + b = 0$ to separate two classes. Instances on the hyperplane satisfy:

$$w{\cdot}xi + b = 0$$

The goal is to maximize the margin distance between the hyperplane and nearest instances called support vectors. The optimization formula is:

$$Maximize: \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j Y_i Y_j (X_i . X_j)$$

Subject to:

$$0 \le \alpha_i \le C \; for \; i = 1,2,\ldots,N$$

DOI: 10.56892/bima.v8i1.603

$$\sum\nolimits_{i=1}^{N} \alpha_i Y_i = 0$$

The decision function $f(X)$ is then expressed as a linear combination of support vectors:

$$f(X) = \sum\nolimits_{i=1}^{N} \alpha_i Y_i (X.X_i) + b$$

The support vectors are the training samples corresponding to non-zero α_i.

3.2.3 Naive Bayes

Naive Bayes calculates the posterior probability of a class y given features x using Bayes' theorem:

$$P(y|x) = \frac{P(y)\,P(x|y)}{P(x)}$$

For Gaussian NB, the likelihood $P(x|y)$ is calculated under a Gaussian distribution assumption:

$$P(x|y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

where μ and σ are the mean and variance estimators.

Decision Trees

Decision trees split the input space into rectangular sub-regions based on if-then-else rules. Splits are chosen to maximize an impurity criterion I(t) at each node t, like Gini impurity or information gain. Typical hyperparameters are max tree depth, min samples per leaf, split criteria, and number of features considered.

Gini Impurity is calculated using the following mathematical formula

$$I_{Gini}(t) = 1 - \sum_{i=1}^{c} p(i/t)^2$$

Where c is the number of classes, and *p(i|t)* is the proportion of instances of class *i* at node *t*

Information gain is calculated using the following mathematical formula;

$$I_{Gain}(t) = - \sum_{i=1}^{c} p(i|t) log_2(p(i|t))$$

Where $p(i|t)$ is the same as above. The decision tree algorithm seeks to split the data in a way that maximizes information gain or reduces Gini impurity. The specific formula for choosing the best split varies with the implementation, but the idea is to compare the impurity before and after the split and choose the split that maximizes the reduction in impurity. These formulas are used during the training process to determine how to split the data at each node of the decision tree. The goal is to create splits that result in pure nodes, where all instances belong to a single class.

RESULT S AND DISCUSSION

Experimental Setup

The experiments compared eight hyperparameter optimization methods - grid search, random search, Bayesian optimization with Gaussian processes (BO-GP), tree-structured Parzen estimator (BO-TPE), Hyperband, a combination of BO and Hyperband (BOHB), genetic algorithm, and particle swarm optimization. The algorithms were evaluated on optimizing three machine learning models - k-nearest neighbors (KNN), support vector machines (SVM), and random forests (RF). The goal was classification on the Modified National Institute of Standards and Technology database image dataset and regression on the Boston housing dataset.

MNIST contains 70,000 grayscale handwritten digit images in 10 classes. The ML models were implemented in Python using the scikit-learn, Skopt, Hyperopt, Optunity, Hyperband, BOHB, DEAP, and TPOT libraries. The optimization objective was to minimize classification error (1 -

accuracy) for MNIST and mean squared error for Boston housing. 3-fold cross-validation was used to estimate out-of-sample performance. The optimization budget was 10 iterations for KNN and 50 iterations for RF and SVM. Experiments were repeated 10 times with different random seeds and the results averaged.

The hyperparameter search spaces were defined based on best practices:

- KNN: k in [1, 20]
- SVM: C in [0.1, 50], kernel as radial basis function, polynomial, linear, or sigmoid
- RF: number of trees in [10, 100], maximum depth in [5, 50], minimum leaf samples from [1, 11], etc.

The same search spaces were used for all optimization methods to ensure a fair comparison. Grid search exhausted the full space, while other methods explored a subset based on their search heuristics. Model performance and optimization time were compared across techniques. The code for reproducing the experiments was shared publicly. This rigorous experimental design evaluates the real-world performance of the different hyperparameter optimization approaches on representative machine learning tasks.

Result

Machine learning models have various hyperparameters that can be tuned to optimize performance. Different optimization techniques like grid search or random search are suitable for tuning different models. Python provides many libraries like Scikit-Learn, Keras, and PyTorch for implementing common ML algorithms. Below is a summary of some popular models, their key hyperparameters, preferred optimization methods, and Python library options:

Table 1: A comprehensive overview of common ML models, their hyper-parameters, suitable optimization techniques, and available Python libraries

ML Algorithm	Main HPs	HPO methods	Optional HPs	Libraries
Linear regressions	-	-	-	-
Logistic regression	Penalty, c, solver	BO-TPE, SMAC	-	Hyperopt, SMAC
KNN	n-neighbors	BOs, Hyperband	Weights, P, Algorithm	Skpot, Hyperopt, SMAC, Hyperband
SVM	C, kernel, epsilon (for SVR)	BO-TPE, SMAC, BOHB	gamma, coef0, degree	Hyperopt, SMAC, BOHB
Deep learning	number of hidden layers, units per layer, loss, optimizer, Activation, learning rate, dropout rate, epochs, batch size, early stop patience	PSO, BOHB	number of frozen layers (if transfer learning is used)	Optunity, BOHB
RF & ET	n estimators max depth,	GA, PSO, BO-TPE, SMAC,	splitter, min weight fraction leaf,	TPOT, Optunity, SMAC, BOHB

DOI: 10.56892/bima.v8i1.603

	criterion, min samples split, min samples leaf, max features	BOHB	max leaf nodes	
AdaBoost	base estimator, n estimators, learning rate	BO-TPE, SMAC	-	Hyperopt, SMAC
K-means	n clusters	BOs, Hyperband	init, n init, max iter	Skpot, Hyperopt, SMAC, Hyperband

Table 2: Configuration space for the hyper-parameters of tested ML models

ML Model	Hyper-parameter	Type	Search Space
RF Classifier	n_estimators	Discrete	[10,100]
	max_depth	Discrete	[5,50]
	min_samples_split	Discrete	[2,11]
	min_samples_leaf	Discrete	[1,11]
	criterion	Categorical	['gini','entropy']
	max_features	Discrete	[1,64]
SVM Classifier	C	Continuous	[0.1,50]
	Kernel	Categorical	['Linear','poly','rbf','sigmoid']
KNN Classifier	n_neighbors	Discrete	[1,20]

We evaluate the hyperparameter optimization methods on common benchmark datasets - the MNIST digit image classification dataset. These standard benchmarks allowed the focus to remain on the optimization techniques rather than data idiosyncrasies.

Three machine learning models were optimized - k-nearest neighbors (KNN), support vector machines (SVM), and random forests (RF). These were selected to cover a range of hyperparameter types. KNN has a single key hyperparameter, the number of neighbors. SVM involves continuous hyperparameters like the regularization strength and categorical hyperparameters like the kernel function. Finally, RF has a large mixed hyperparameter space controlling aspects like tree count, maximum depth, split metrics, etc.

The optimization goal was model accuracy for MNIST classification. Performance was estimated using 3-fold cross-validation and computational time was also measured. The technique finding the best model performance in the shortest time was preferred.

To promote fair comparisons, some consistency was enforced across methods. The hyperparameter search spaces were predefined identically for all techniques based on recommended ranges from literature and manual tuning. The optimization budget was set at 10 iterations for KNN and 50 for SVM and RF based on convergence behavior. Experiments were repeated across random seeds and results averaged. By using standard datasets and models and controlling the search spaces, budgets, and evaluation methodology, the experiments isolated the impact of the optimization approach itself. This provided insight into real-world performance for machine learning tasks with different model architectures and hyperparameter characteristics.

All experiments were conducted using Python 3.5 on a machine with AMD A4-5000 APU processor and 500 gigabytes (GB) of memory. The involved ML and HPO algorithms are evaluated using multiple open-source Python libraries and frameworks, including sklearn, Skopt, Hyperopt, Optunity, Hyperband and TPOT.

Table 3: Assessing effectiveness of utilizing automatic hyperparameter optimization techniques for random forest models on handwritten digit benchmark

Optimization Algorithm	**Accuracy (%)**	**CT(s)**
Default HPs	94.38	0.06
GS	93.60	25.11
RS	92.43	20.10
BO-GP	93.99	12.59
BO-TPE	93.49	8.00
Hyperband	93.20	9.03
GA	93.32	19.10
PSO	92.56	12.43

This table presents the results of experiments evaluating various automatic hyperparameter optimization (HPO) techniques for Random Forest models applied to a handwritten digit benchmark. The optimization algorithms, including Default Hyperparameters (Default HPs), Grid Search (GS), Random Search (RS), Bayesian Optimization with Gaussian Processes (BO-GP), Bayesian Optimization with Tree-structured Parzen Estimators (BO-TPE), Hyperband, Genetic Algorithms (GA), and Particle Swarm Optimization (PSO), are assessed based on Accuracy (%) and Computation Time (CT) in seconds.

The Default HPs show a solid baseline accuracy, indicating that the initial hyperparameter settings are reasonably effective for Random Forest models in this context. The low associated computation time suggests that manual tuning might be sufficient for achieving competitive results.

Among the automated techniques, Grid Search (GS) has a decent accuracy but exhibits significantly higher computation times. Random Search (RS) follows with a slightly lower accuracy but shows improved efficiency in terms of computation time compared to GS.

Bayesian Optimization techniques, BO-GP and BO-TPE, deliver competitive accuracy levels with relatively low computation times, reaffirming their effectiveness in guiding the search for optimal hyperparameters.

Hyperband performs well in terms of accuracy, but its computation times are higher compared to BO-GP and BO-TPE, suggesting a trade-off between model performance and computational efficiency.

Genetic Algorithms (GA) showcase respectable accuracy levels, but the associated computation times are relatively high. Particle Swarm Optimization (PSO) achieves a balance between accuracy and computation time. The table provides valuable insights into the comparative performance of automatic hyperparameter optimization techniques for Random Forest models. The results highlight the trade-offs between accuracy and computation time, aiding practitioners in selecting the most suitable algorithm based on their specific requirements and constraints.

Table 4: Assessing effectiveness of utilizing automatic hyperparameter optimization techniques for Support vector machine models on handwritten digit benchmark

Optimization Algorithm	Accuracy (%)	CT(s)
Default HPs	96.99	0.27
GS	97.38	31.50
RS	97.44	11.98
BO-GP	97.44	16.30
BO-TPE	97.50	3.09
Hyperband	97.44	11.19
GA	97.44	15.14
PSO	96.05	7.70

This table presents the results of experiments assessing the effectiveness of automatic hyperparameter optimization (HPO) techniques for Support Vector Machine (SVM) models on a handwritten digit benchmark. The optimization algorithms, including Default Hyperparameters (Default HPs), Grid Search (GS), Random Search (RS), Bayesian Optimization with Gaussian Processes (BO-GP), Bayesian Optimization with Tree-structured Parzen Estimators (BO-TPE), Hyperband, Genetic Algorithms (GA), and Particle Swarm Optimization (PSO), are evaluated based on Accuracy (%) and Computation Time (CT) in seconds.

The Default HPs exhibit a respectable baseline accuracy, suggesting that the initial hyperparameter settings are reasonably effective for SVM models in this context. The associated computation time is low, indicating that manual tuning might suffice for achieving competitive results.

Grid Search (GS) produces competitive accuracy levels but at the expense of significantly higher computation times. Random Search (RS) shows slightly lower accuracy but improved efficiency in terms of computation time compared to GS.

Bayesian Optimization techniques, BO-GP and BO-TPE, yield promising results with competitive accuracy and relatively low computation times. This emphasizes the effectiveness of probabilistic models in guiding the search for optimal hyperparameters.

Hyperband performs well in terms of accuracy, but its computation times are higher compared to BO-GP and BO-TPE, suggesting a trade-off between model performance and computational efficiency.

Genetic Algorithms (GA) showcase respectable accuracy levels, but the associated computation times are on the higher side. Particle Swarm Optimization (PSO) performs adequately, striking a balance between accuracy and computation time. The table provides insights into the comparative performance of automatic hyperparameter optimization techniques for SVM models. The results can guide practitioners in selecting the most suitable algorithm based on their specific constraints and priorities.

Table 5: Assessing effectiveness of utilizing automatic hyperparameter optimization techniques for KNN models on handwritten digit benchmark

Optimization Algorithm	Accuracy (%)	CT(s)
Default HPs	96.27	0.24
GS	96.83	7.46
RS	96.44	6.40
BO-GP	96.83	1.16
BO-TPE	96.83	2.20
Hyperband	96.22	4.43
GA	96.86	2.24
PSO	96.83	1.50

This table presents the results of experiments assessing the effectiveness of automatic hyperparameter optimization (HPO) techniques for K-Nearest Neighbors

(KNN) models on a handwritten digit benchmark. The optimization algorithms, including Default Hyperparameters (Default HPs), Grid Search (GS), Random Search (RS), Bayesian Optimization with Gaussian Processes (BO-GP), Bayesian Optimization with Tree-structured Parzen Estimators (BO-TPE), Hyperband, Genetic Algorithms (GA), and Particle Swarm Optimization (PSO), are evaluated based on Accuracy (%) and Computation Time (CT) in seconds.

The Default HPs exhibit a strong baseline accuracy, suggesting that the initial hyperparameter settings are effective for KNN models in this context. The associated computation time is low, indicating that manual tuning may be sufficient for competitive results.

Grid Search (GS) produces a competitive accuracy, but its computation times are significantly higher. Random Search (RS) demonstrates slightly lower accuracy but improved efficiency in terms of computation time compared to GS.

Bayesian Optimization techniques, BO-GP and BO-TPE, yield competitive accuracy levels with relatively low computation times. This underscores the effectiveness of probabilistic models in guiding the search for optimal hyperparameters.

Hyperband performs well in terms of accuracy, but its computation times are higher compared to BO-GP and BO-TPE, suggesting a trade-off between model performance and computational efficiency.

Genetic Algorithms (GA) showcase respectable accuracy levels, but the associated computation times are relatively high. Particle Swarm Optimization (PSO) achieves a balance between accuracy and computation time. The table provides insights into the comparative performance of automatic hyperparameter optimization techniques for KNN models.

Table 6: Comparison with the work of (Vincent & Jidesh, 2023)

Optimization Algorithm	Accuracy (%)	Vincent and Jidesh, 2023
Default HPs	99.94	-
GS	99.94	-
RS	99.94	89.89
BO-GP	100	94.1
BO-TPE	100	-
Hyperband	99.96	-
GA	99.94	77.9
PSO	99.07	-

The table above compare the result found in our work with the work in (Vincent & Jidesh, 2023). It can be seen that our result has produced a result far better than what the aforementioned author has produced.

Table 7: Assessing effectiveness of utilizing automatic hyperparameter optimization techniques for ANN models on handwritten digit benchmark

Optimization Algorithm	Accuracy (%)	CT(s)
Default HPs	99.94	10
GS	99.94	9.6
RS	99.94	6.40
BO-GP	100	13
BO-TPE	100	15.0
Hyperband	99.96	4.6
GA	99.94	3.25
PSO	99.07	3.50

The result is even far better when considering our ANN which is the best algorithm in our case.

5. Conclusion

The research thoroughly evaluated eight hyperparameter optimization techniques for image classification. The methods showed varied strengths and weaknesses based on model architecture and search space complexity. Grid search guarantees global optimality but scales poorly with dimensionality. Random search is efficient but lacks guidance. Gaussian process Bayesian optimization works well for low-dimensional spaces but has cubic scaling issues. Tree Parzen estimator is more efficient but struggles to parallelize. Hyperband prioritizes time but sacrifices solution quality. Genetic algorithms are competitive but hard to parallelize. Particle swarm optimization achieves the best accuracy-time tradeoff. No single technique is universally superior, and the choice depends on factors like search space size, objective function cost, dimensionality, and parallelizability. Hybrid methods, such as Bayesian optimization - Hyperband, show promise. Particle swarm optimization and ensemble-based techniques appear robust. Future work should develop benchmarks for emerging model classes.

Futue Work

Future research directions include creating benchmarks for complex model classes like deep neural networks, automating the selection of optimal optimization algorithms, and exploring hybrid techniques that blend strengths from methods like Bayesian optimization, bandits, and evolutionary algorithms while mitigating weaknesses.

REFERENCES

Alet, F., Lozano-Perez, T., & Kaelbling, L. P. (2018). Modular meta-learning. *arXiv preprint arXiv*:1806.10166.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb), 281-305.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33, 1877-1901.

Claesen, M., & De Moor, B. (2015). Hyperparameter search in machine learning. *arXiv preprint arXiv*:1502.02127.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10), 78-87.

Domhan, T., Springenberg, J. T., & Hutter, F. (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. *In Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Falkner, S., Klein, A., & Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale. In International Conference on Machine Learning (pp. 1437-1446). PMLR.

Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., & Sculley, D. (2017). Google vizier: A service for black-box optimization. In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1487-1495).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

Hutter, F., Kotthoff, L., & Vanschoren, J. (2018). Algorithm selection for combinatorial search problems: A survey. *In Data Mining and Constraint Programming. Springer, Cham*.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J.,

DOI: 10.56892/bima.v8i1.603

Razavi, A., & Kavukcuoglu, K. (2017). Population based training of neural networks. *arXiv preprint arXiv*:1711.09846.

Jin, H., Song, Q., & Hu, X. (2019). Auto-keras: An efficient neural architecture search system. *In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1946-1956).

Jomanto, F., Ghassami, A., & Yang, T. (2021). Hyperparameter Optimization Of Deep Neural Networks: A Survey. *ArXiv*, abs/2103.05982.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255-260.

Kandasamy, K., Dasarathy, G., Schneider, J., & Poczos, B. (2018). The multi-fidelity multi-armed bandit. *In Proceedings of the 32nd International Conference on Neural Information Processing Systems* (pp. 1777–1787).

Klein, A., Falkner, S., Mansur, N., & Hutter, F. (2017). RoBO: A flexible and robust Bayesian optimization framework in Python. *In NIPS 2017 Bayesian Optimization Workshop*.

Kohavi, R., & Wolpert, D. H. (1996). Neural networks for pattern recognition. *Statistical learning theory and practice*.178-184.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

Lindauer, M., & Hutter, F. (2019). Best practices for scientific research on hyperparameter optimization. *Journal of Machine Learning Research*, 21(243), 1-18.

Luketina, J., Berglund, M., Greff, K., & Raiko, T. (2016). Scalable gradient-based tuning of continuous regularization hyperparameters. In International conference on machine learning (pp. 2952-2960). PMLR.

Melis, G., Dyer, C., & Blunsom, P. (2018). On the state of the art of evaluation in neural language models. arXiv preprint arXiv:1707.05589.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Model-agnostic interpretability of machine learning. arXiv preprint *arXiv*:1606.05386.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & De Freitas, N. (2015). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1), 148-175.

Wistuba, M., Schilling, N., & Schmidt-Thieme, L. (2015). Sequential model-free hyperparameter tuning. *In 2015 IEEE international conference on data mining* (pp. 1033-1038). IEEE.

Yu, K., & Zhu, S. (2020). Hyper-parameter optimization: A review of algorithms and applications. arXiv preprint *arXiv*:2003.05689.

Benbouzid, D., Busa-Fekete, R., Casagrande, N., Collin, F.-D., & Kégl, B. (2012). MultiBoost: a multi-purpose boosting package. *Journal of Machine Learning Research*, 13(Aug), 549-553.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb), 281-305.

Claesen, M., & De Moor, B. (2015). Hyperparameter search in machine learning. arXiv preprint arXiv:1502.02127.

Domhan, T., Springenberg, J. T., & Hutter, F. (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. *In IJCAI* (Vol. 15, pp. 3460-3468).

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10), 78-87.

Falkner, S., Klein, A., & Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale. *In International Conference on Machine Learning* (pp. 1437-1446). PMLR.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

Hutter, F., Kotthoff, L., & Vanschoren, J. (2018). Algorithm selection for combinatorial search problems: A survey. Data Mining and Constraint Programming, 149-190. *Springer, Cham*.

Jomanto, F., Ghassami, A., & Yang, T. (2021). Hyperparameter Optimization of Deep Neural Networks: A Survey. ArXiv, abs/2103.05982.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255-260.

Kohavi, R., & Wolpert, D. H. (1996). Neural networks for pattern recognition. *Statistical learning theory and practice*.178-184.

Komer, B., Bergstra, J., & Eliasmith, C. (2014). Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. *In ICML workshop on AutoML* (Vol. 9).

Rangapuram, S. S., Seegerer, M., Gasthaus, J., Stella, L., Wang, Y., & Januschowski, T. (2018). Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31.

Hertel, L., Collado, J., Sadowski, P., Ott, J., & Baldi, P. (2020b). Sherpa: Robust hyperparameter optimization for machine learning. *SoftwareX*, 12, 100591. https://doi.org/10.1016/j.softx.2020.100591

Schratz, P., Muenchow, J., Iturritxa, E., Richter, J., & Brenning, A. (2019). Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data. *Ecological Modelling*, 406, 109–120. https://doi.org/10.1016/j.ecolmodel.2019.06.002

Wu, J., Chen, X. Y., Zhang, H., Xiong, L. D., Lei, H., & Deng, S. (2019). Hyperparameter optimization for machine learning models based on Bayesian optimization. *Journal of Electronic Science and Technology*, 17(1), 26–40. https://doi.org/10.11989/jest.1674-862x.80904120

Pannakkong, W., Thiwa-Anont, K., Singthong, K., Parthanadee, P., & Buddhakulsomsiri, J. (2022). Hyperparameter tuning of machine learning algorithms using response surface methodology: A case study of ANN, SVM, and DBN. *Mathematical Problems in Engineering*, 2022, 1–17. https://doi.org/10.1155/2022/8513719

Raji, I. D., Bello-Salau, H., Umoh, I. J., Onumanyi, A. J., Adegboye, M. A., & Salawudeen, A. T. (2022). Simple deterministic Selection-Based genetic algorithm for hyperparameter tuning of machine learning models.

Applied Sciences, 12(3), 1186. https://doi.org/10.3390/app12031186

Elgeldawi, E., Sayed, A., Galal, A. R., & Zaki, A. M. (2021). Hyperparameter tuning for machine learning algorithms used for Arabic sentiment analysis. *Informatics (Basel)*, 8(4), 79. https://doi.org/10.3390/informatics8040079

Fan, Y., Zhang, Y., Guo, B., Luo, X., Peng, Q., & Jin, Z. (2022). A hybrid sparrow search algorithm of the hyperparameter optimization in deep learning. *Mathematics*, *10*(16), 3019. https://doi.org/10.3390/math10163019

Ali, Y. A., Awwad, E. M., Al-Razgan, M., & Maarouf, A. (2023). Hyperparameter search for machine learning algorithms for optimizing the computational complexity. *Processes*, 11(2), 349. https://doi.org/10.3390/pr11020349

Tani, L., Rand, D., Veelken, C., & Kadastik, M. (2021). Evolutionary algorithms for hyperparameter optimization in machine learning for application in high energy physics. *The European Physical Journal C*, 81(2). https://doi.org/10.1140/epjc/s10052-021-08950-y

Morales-Hernández, A., Van Nieuwenhuyse, I., & Gonzalez, S. R. (2022). A survey on multi-objective hyperparameter optimization algorithms for machine learning. *Artificial Intelligence Review*, 56(8), 8043–8093. https://doi.org/10.1007/s10462-022-10359-2