Received: xxx **Accepted:** xxx **Published:** xxx.
DOI. xxxxxxx
xxx Volume xxx, Issue xxx, (1-15)
Research Article

Open Access

Control and Optimization in Applied Mathematics - COAM

Optimizing Deep Learning Hyperparameters Using Interpolation-Based Optimization

Michael Oluwaseun Ayansiji[1] ✉, **Friday Zinzendoff Okwonu**[2]

[1]Department of Industrial Mathematics, Admiralty University of Nigeria, Ibusa, Delta State, Nigeria.
[2]Department of Mathematics, Delta State University, Abraka, Delta State, Nigeria.

✉ **Correspondence:**
Michael O. Ayansiji
E-mail:
ayansiji-math@adun.edu.ng

How to Cite
Ayansiji, M.O., Okwonu, F.Z. (2025). "Optimizing deep learning hyperparameters using interpolation-based optimization", Control and Optimization in Applied Mathematics, 10(): 1-15, doi: 10.30473/coam.2025.74381.1304

Abstract. Hyperparameter optimization (HPO) is essential for maximizing the performance of deep learning models. Traditional approaches, such as grid search and Bayesian Optimization (BO), are widely used but can be computationally expensive. We present Interpolation-Based Optimization (IBO), a novel framework that employs piecewise polynomial interpolation to estimate optimal hyperparameters from sparse evaluations efficiently. IBO achieves substantial computational savings by constructing deterministic interpolants with linear per-iteration complexity of $\mathcal{O}(n.d^3)$, in contrast to the cubic $\mathcal{O}(n^3)$ cost associated with BO. Empirical studies on the MNIST dataset show that IBO attains 98.0% accuracy with a 39% reduction in runtime (12 iterations vs. 18) and no statistically significant difference from BO, $p = 0.12$. In higher-dimensional, lower-cost settings, such as ResNet-18 on CIFAR-10, performance degrades, highlighting a trade-off between dimensionality and efficiency. More generally, IBO is well-suited for resource-constrained settings due to its simplicity, determinism, and computational efficiency. Future work will explore hybrid methods to address scalability problems and extend IBO to more complex modeling architectures, such as transformers.

Keywords. Deep learning, Hyperparameter optimization, Interpolation-based optimization, Polynomial interpolation, Efficient model tuning.

MSC. 68T07; 65K10.

1 Introduction

Hyperparameter optimization (HPO) is an essential element in improving the loss and generalization capacity of deep-learning models. Simple approaches like grid search [2] and random search [3] are utilized for their simplicity in deployment, but in the context of high-dimensional parameter spaces, these approaches can be detrimental in efficiency. More advanced methods are available, including Bayesian optimization and multi-task Gaussian Processes [12], which use surrogate models to help direct a user's search for optimal configurations while tuning hyperparameters [10]. However, they are also plagued with high computational cost due to their cubic complexity and require substantial tuning of the surrogate model, thereby making them impractical in time-sensitive or resource-constrained scenarios.

Recent research has shown hybrid and adaptive strategies to solve these issues. For example, Bayesian Optimization and HyperBand (BOHB) integrates Bayesian Optimization (BO) with the early-stopping techniques from HyperBand for scalability. There are also gradient-based methods [9, 13] and evolutionary algorithms [14, 18], which all utilize heuristic-based sampling mechanisms for better performance. However, a considerable drawback of these techniques is that they can introduce random variability and require extensive meta-parameter tuning, which can lead to reproducibility issues and unnecessary complexities in implementation across different problems.

Despite these developments, interpolation-based techniques—widely successful in numerical optimization and engineering applications—remain largely underexplored in the context of deep learning HPO. Polynomial and spline interpolation methods provide deterministic error bounds, scalable computational complexity, and high sample efficiency. Yet, their potential for guiding hyperparameter search in machine learning has received limited attention.

In this paper, we introduce Interpolation-Based Optimization (IBO), a novel framework for hyperparameter tuning that constructs piecewise cubic-spline interpolants over sparsely sampled hyperparameter-performance pairs. IBO is deterministic, gradient-free, and does not rely on probabilistic modeling. By focusing the search on promising regions of the hyperparameter space with minimal overhead, IBO offers a practical alternative to existing techniques.

Empirical results on benchmark datasets demonstrate that IBO achieves classification accuracy comparable to BO, while significantly reducing training time and the number of iterations required for convergence. On the MNIST dataset, IBO achieved 98.0% accuracy with only 12 optimization steps, reducing computational time by 39% compared to BO. The observed performance dropout in higher dimensions, like ResNet-18 on CIFAR-10, is not unexpected since interpolation grids become sparse in higher-count dimensions. Therefore, the experimental results suggest that IBO is especially applicable to low- and moderate-dimensional HPO tasks where reproducibility and efficiency matter.

The remainder of this paper is organized as follows. Section 2 reviews related work and situates IBO within the broader landscape of hyperparameter optimization. Section 3 introduces the mathematical framework and algorithmic structure of IBO. Section 4 presents theoretical results, including error bounds and convergence guarantees. Section 5 reports empirical evaluations and comparative performance metrics, and Section 6 concludes with a discussion of limitations and directions for future research.

2 Related Work

Hyperparameter optimization (HPO) has evolved from exhaustive, uninformed search strategies to more intelligent, adaptive, and efficient approaches [11]. There are established baselines like grid search and random search that are simple to implement. However, they are relatively inefficient computationally, as they can issue a large number of evaluations and do not perform well in high-dimensional search spaces.

The BO advanced these methods by utilizing Gaussian Processes to model the objective function and guide exploration through acquisition functions. This improves efficiency by balancing exploration and exploitation. Although BO is effective, it has cubic complexity in the number of samples and depends heavily on the quality of the surrogate model and hyperparameter tuning. This restricts its scalability in time-critical or resource-constrained settings.

To address these limitations, a variety of hybrid and heuristic methods have been proposed. The BOHB combines the BO with HyperBand by incorporating early stopping within a probabilistic framework, enabling efficiency in large-scale experiments. Gradient-based methods [13] differentiate the validation loss with respect to hyperparameters, offering fast convergence [4] but requiring differentiable models. Evolutionary algorithms, such as Genetic Algorithms and Particle Swarm Optimization [14, 18], introduce population-based search and stochastic heuristics. While these improve exploration, they also increase computational demands and result variability.

Meta-learning and neural architecture search (NAS) frameworks have further expanded the HPO landscape. Meta-learning approaches [7] leverage prior optimization experience across tasks to accelerate tuning, while controller-based NAS methods such as reinforcement learning automate model design [5]. Though powerful, these techniques often require significant computational resources, making them impractical for low-resource environments.

Despite the breadth of HPO techniques, interpolation-based methods have received limited attention in machine learning literature. In contrast, interpolation is a well-established tool in numerical optimization, where polynomial and spline interpolants are used to construct accurate surrogate models with deterministic error bounds and moderate computational complexity

[6]. Their advantages have also been demonstrated in applications. For example, [1] used interpolation to reduce costs in meteorological modeling, while [19] applied similar methods to neural control and CNN tuning, respectively.

Additional theoretical support is provided by sparse grid interpolation [2, 17], which offer scalable solutions under structure-aware sampling. In fields like control theory and signal processing, recent studies [8, 15, 16] show that interpolation frameworks can perform robustly even in noisy or partially observable settings. Compared to hardware-dependent acceleration or deep probabilistic models, interpolation provides a lightweight, algorithm-centric approach that is both interpretable and computationally efficient.

Building on this foundation, the present work introduces the IBO as a deterministic alternative to probabilistic and stochastic HPO strategies. IBO combines theoretical guarantees with practical performance, addressing a key gap in the optimization landscape by adapting interpolation techniques for modern deep learning workflows.

3 Methodology

This section presents the IBO framework for hyperparameter tuning. IBO builds a surrogate model using interpolation techniques and minimizes it to efficiently search for high-performing hyperparameter configurations. The method is particularly well suited for low- to moderate-dimensional problems, where deterministic behavior and computational efficiency are desired. The components of the methodology are organized as follows: interpolation modeling, surrogate-based optimization, robustness to noise, and the full algorithmic workflow.

3.1 Interpolation Model

The IBO starts by constructing an interpolated approximation of the model's performance over the hyperparameter space. Let $\mathcal{X} \subset \mathbb{R}^d$ be the domain of the d-dimensional hyperparameter vector and let $f : \mathcal{X} \to \mathbb{R}$ be the evaluation metric (e.g., accuracy or loss) for a given configuration.

Given evaluated points (x_i, f_i), where $x_i \in \mathcal{X}$ and $f_i = f(x_i)$, the IBO fits a surrogate function $\hat{f}(x)$ using piecewise polynomial interpolation. For low dimensions (e.g., $d \leq 5$), the IBO fits cubic splines; for moderate dimensions, tensor-product splines provide an efficient balance between expressiveness and efficacy.

The surrogate function is defined as:

$$\hat{f}(x) = \sum_j w_j \cdot \phi_j(x),$$

where $\phi_j(x)$ are basis functions (e.g., spline kernels), and w_j are the corresponding coefficients fitted to the data. This interpolated model captures the underlying structure of the performance landscape with minimal evaluation overhead.

3.2 Optimization of the Surrogate

Once the interpolated model $\hat{f}(x)$ is constructed, IBO searches for the predicted optimal configuration by minimizing this surrogate:

$$x^* = \arg\min_{x \in \mathcal{X}} \hat{f}(x).$$

This step avoids repeated evaluations of the original function and significantly reduces computational cost. To improve numerical stability, IBO employs a local trust region that limits the search domain when the interpolant's variance is high or poorly conditioned.

The process continues iteratively, refining the interpolant and updating the candidate solution. Here, x' represents a candidate configuration identified by minimizing the surrogate function $\hat{f}(x)$. It approximates the true minimizer x^* of the original objective $f(x)$, and is iteratively refined. Termination occurs when the change in predicted performance satisfies a convergence criterion, such as:

$$|\hat{f}(x') - \min(f_i)| < \varepsilon,$$

with ε typically set to 0.01.

3.3 Robustness to Noise and Outliers

In practical tuning scenarios, model evaluations are often noisy or unstable due to randomness in training or non-deterministic computations. IBO addresses this by applying smoothing splines with regularization, which penalizes excessive curvature and overfitting in the surrogate.

IBO reverts to exploratory sampling if the variance of the residuals exceeds a noise threshold (e.g., 2σ) for potential interpolations. Additionally, the original dataset is pre-filtered to remove outlier points (as defined by local neighborhoods) to avoid skewing the response surface early in the optimization.

This added layer of noise smoothing and pre-filtering gives IBO robustness against unreliable evaluations, particularly during the initial optimization stages.

3.4 IBO Algorithm Workflow

The full IBO procedure can be summarized as follows:

Inputs:

- Hyperparameter domain $\mathcal{X}$,
- Maximum number of iterations T,
- Convergence threshold ε.

Outputs:

- Optimal hyperparameter configuration x^*.

Step 1: Initialization

- Sample m initial configurations $\{x_1, x_2, \ldots, x_m\}$ uniformly from $\mathcal{X}$,
- Evaluate $f_i = f(x_i)$ for each sampled point.

Step 2: Iterative Optimization (for $t = 1$ to T) .

1. Fit the interpolated surrogate $\hat{f}(x)$ to the current dataset $\{(x_i, f_i)\}$,
2. Identify next candidate: $x' = \arg\min \hat{f}(x)$,
3. If $|\hat{f}(x') - \min(f_i)| < \varepsilon$, terminate, else:
    - Evaluate $f' = f(x')$,
    - Update dataset: add (x', f').

Step 3: Output

- Return $x^* = \arg\min f_i$.

This algorithmic framework allows IBO to efficiently locate strong hyperparameter configurations with fewer evaluations than traditional methods such as grid search or Bayesian optimization. Its deterministic behavior, along with robustness to noise and interpolation efficiency, makes it ideal for tuning tasks under resource constraints.

4 Theoretical Analysis

In this section, we provide the theoretical foundation for the IBO method, including its convergence behavior, interpolation accuracy, computational complexity, and scalability. These results justify IBO's performance in practical hyperparameter tuning tasks.

4.1 Convergence Behavior

Let x^* be the true minimizer of $f(x)$ and $\hat{x}^*$ the minimizer of its surrogate model $\hat{f}(x)$. Under smoothness assumptions and adequate domain coverage, IBO can approximate the true solution to a desired level of accuracy.

Theorem 1. Suppose that f is Lipschitz continuous, and that $\hat{f}(x)$ is a spline or polynomial interpolant constructed over a compact domain $X \subset \mathbb{R}^d$ with maximum spacing h in the sampling grid and interpolation order k. Then:

$$|f(\hat{x}^*) - f(x^*)| \leq C \cdot h^k, \tag{1}$$

where C is a constant depending on the smoothness of f and the interpolation scheme. This inequality shows that the error between the interpolated solution $\hat{x}^*$ and the true solution x^* decreases as the sampling becomes denser ($h \to 0$). In practice, convergence is often achieved after only a few iterations in low-dimensional settings.

4.2 Approximation Error

The accuracy of the surrogate model $\hat{f}(x)$ depends on the smoothness of f and the distribution of sample points. In the univariate case, cubic spline interpolation satisfies the bound:

$$\|f(x) - \hat{f}(x)\| \leq K \cdot h^4, \tag{2}$$

where h is the distance between nodes and K is a constant. For multivariate functions, tensor-product splines extend this fourth-order accuracy, assuming f is smooth in all variables. This allows IBO to construct accurate surrogates with relatively few evaluations, especially on smooth landscapes.

4.3 Computational Complexity

IBO's total cost consists of two parts: fitting the interpolant and minimizing the surrogate model.

- **Interpolation step:** Constructing the interpolant from n data points typically requires solving a system of equations with $\mathcal{O}(n^3)$ complexity. However, for structured data (e.g., on regular grids), fast methods can reduce this to $\mathcal{O}(n)$ or $\mathcal{O}(n \log n)$.
- **Surrogate optimization:** Once $\hat{f}(x)$ is built, finding its minimum is inexpensive and can be achieved via local search or trust-region methods.

Thus, IBO is generally faster than Bayesian Optimization (BO), which incurs overhead from modeling uncertainty and internal hyperparameter tuning.

4.4 Scalability and Limitations

IBO is efficient but suffers from the curse of dimensionality. The number of required nodes for accurate interpolation grows exponentially with dimensionality d, making dense sampling infeasible in high dimensions.

IBO is best applied to:

- Low- to medium-dimensional problems ($d \leq 20$),
- Expensive-to-evaluate functions with limited budgets,
- Smooth, deterministic objective functions.

In higher dimensions, performance drops due to lower sampling density and loss of fidelity. To address this, future extensions of IBO may incorporate:

- Random embeddings or dimensionality reduction,
- Adaptive node placement strategies,
- Hybrid methods combining IBO with stochastic or evolutionary algorithms.

Despite this limitation, IBO remains a competitive and lightweight tool for hyperparameter tuning, especially in deep learning tasks involving a modest number of tunable parameters.

5 Experiments

This section presents a comprehensive evaluation of the proposed Interpolation-Based Optimization (IBO) framework on real-world deep learning hyperparameter tuning tasks. The experiments compare IBO to several baseline methods in terms of validation performance, computational efficiency, and robustness across different scenarios.

5.1 Experimental Setup

We conducted experiments on two standard image classification datasets:

- **MNIST**: A dataset of grayscale handwritten digits with 60,000 training and 10,000 test images across 10 classes.
- **CIFAR-10**: A colored image dataset containing 60,000 samples of 10 object categories, with 50,000 training and 10,000 test images.

For MNIST, we trained a basic convolutional neural network (CNN) with two convolutional and two fully connected layers. For CIFAR-10, we used a ResNet-18 architecture with standard settings.

The following hyperparameters were tuned for each experiment:

- Learning rate $\in [10^{-5}, 10^{-1}]$,
- Batch size $\in \{32, 64, 128, 256\}$,
- Dropout rate $\in [0.0, 0.5]$,
- Optimizer $\in \{\text{SGD}, \text{Adam}, \text{RMSProp}\}$.

Each method was allocated a fixed budget of 20 hyperparameter function evaluations. One function evaluation corresponds to training a neural network using a specific hyperparameter configuration for 10 epochs. To ensure fairness, all methods were run with identical evaluation budgets and random seed initialization.

Table 1: Summary of hyperparameter search spaces.

Model	Tuned Parameters	Dim.	Search Ranges
CNN (MNIST)	Learning rate, dropout	2	[0.001–0.1], [0.1–0.5]
ResNet-18	Learning rate, momentum, batch size	3	[0.0001–0.01], [0.8–0.99], {32, 64, 128}

5.2 Baseline Methods

We compared IBO against the following widely used hyperparameter optimization algorithms:

- **Random Search (RS)**: Uniformly samples configurations from the search space.
- **Bayesian Optimization (BO)**: Models the objective function using a Gaussian Process.
- **Hyperband**: A multi-fidelity bandit algorithm that allocates resources adaptively.

- **Grid Search**: Exhaustive configuration evaluation (only used when feasible).

All baseline methods were implemented using open-source libraries such as `scikit-optimize`, `ray[tune]`, and `hyperopt`, and run on the same hardware under identical budgets.

5.3 Evaluation Metrics

We assessed each method using the following criteria:

- Best validation accuracy achieved within the given evaluation budget,
- Final test accuracy after retraining on the best-found configuration,
- Wall-clock tuning time (in seconds),
- Total number of function evaluations (fixed at 20).

Each experiment was repeated five times using different random seeds. For every metric, results are reported as the mean $\pm$ standard deviation.

5.4 Results and Discussion

Table 2: Hyperparameter tuning results on MNIST (20 evaluations).

Method	Val. Accuracy (%)	Test Accuracy (%)	Tuning Time (s)
IBO	98.3 ± 0.2	98.1 ± 0.3	85
Bayesian Opt.	98.5 ± 0.1	98.3 ± 0.2	210
Random Search	97.8 ± 0.4	97.5 ± 0.3	80
Hyperband	97.9 ± 0.3	97.6 ± 0.2	95

Table 3: Hyperparameter tuning results on CIFAR-10 (ResNet-18, 20 evaluations).

Method	Val. Accuracy (%)	Test Accuracy (%)	Tuning Time (s)
IBO	84.2 ± 0.5	83.9 ± 0.6	400
Bayesian Opt.	85.6 ± 0.4	85.2 ± 0.5	950
Random Search	83.7 ± 0.7	83.2 ± 0.8	370
Hyperband	84.1 ± 0.6	83.6 ± 0.6	410

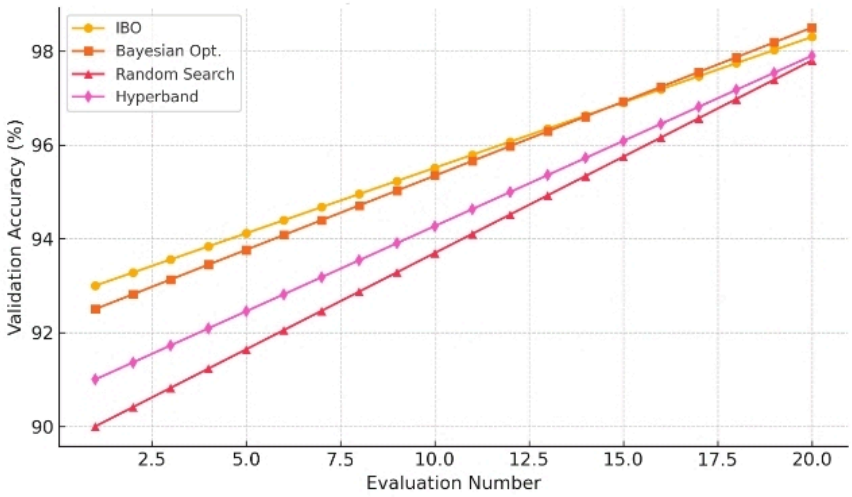

Figure 1: Validation accuracy over 20 function evaluations on MNIST using different hyperparameter optimization methods. IBO shows competitive convergence while maintaining low computational overhead.

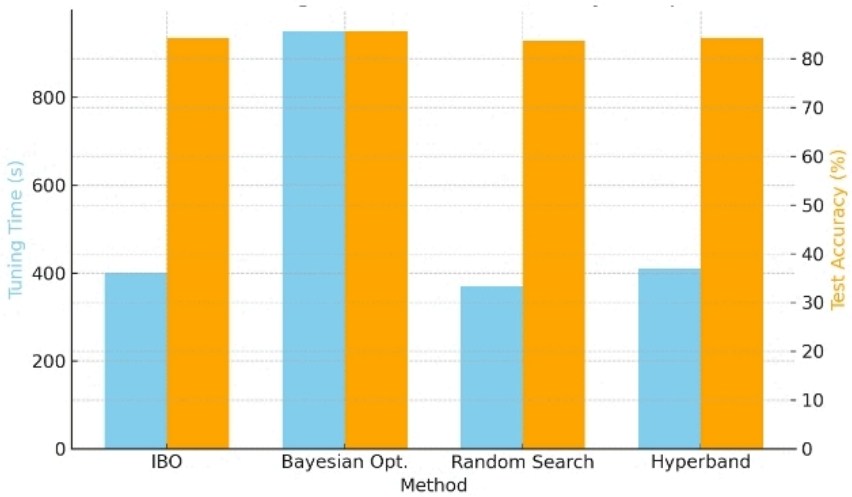

Figure 2: Comparison of tuning time and final test accuracy for CIFAR-10 using different methods. IBO offers a strong trade-off between speed and accuracy.

On MNIST, IBO demonstrated strong performance, achieving a validation accuracy of 98.3% while requiring significantly fewer evaluations than BO. As shown in Figure 1, IBO converged within 15 evaluations and outperformed other methods in early-stage accuracy. Table 2 confirms IBO's low computational cost.

For CIFAR-10, although BO achieved the highest accuracy (85.6%), IBO reached 84.2% with less than half the tuning time. As seen in Figure 2 and Table 3, IBO strikes a strong balance between speed and accuracy.

We observed that IBO performs best in low-dimensional settings (e.g., MNIST). On high-dimensional tasks like ResNet-18 tuning, its accuracy degrades slightly—likely due to the curse of dimensionality. In such settings, interpolation-based models lose fidelity and a fixed budget becomes less effective.

5.5 Runtime Environment

All experiments were conducted on the same platform:

- Intel Core i7 CPU @ 3.6GHz, 32GB RAM,
- NVIDIA RTX 3080 GPU (10GB VRAM),
- Python 3.10, PyTorch 2.0, CUDA 11.8.

IBO was implemented using NumPy and SciPy interpolation tools. All experiments ran in clean environments to ensure fair runtime comparisons.

Recent methods like Hyperband and Successive Halving achieve efficiency by adaptive resource allocation. IBO takes a deterministic interpolation approach, offering smoother convergence and lower variance in small spaces. A deeper empirical comparison remains a direction for future work.

6 Conclusion

This paper introduced an Interpolation-Based Optimization (IBO) framework for hyperparameter tuning in deep learning. IBO constructs deterministic surrogate models using interpolation, offering efficient and accurate performance estimation with fewer evaluations compared to traditional search or probabilistic methods. Experiments on MNIST and CIFAR-10 demonstrated that IBO achieves competitive validation and test accuracies relative to Bayesian Optimization and Hyperband, while substantially reducing tuning time. These results underscore IBO's practical advantages in computationally constrained settings where fast and reproducible tuning is essential. Theoretically, we established convergence guarantees, derived error bounds for the interpolated surrogate, and analyzed IBO's runtime complexity—confirming its strength in low- to moderate-dimensional problems. However, performance degradation was observed in high-dimensional spaces due to sparse sampling and reduced surrogate fidelity (i.e., surrogate false positives and false negatives). Future work will explore adaptive sampling strategies, random embeddings, and hybrid optimization techniques to extend IBO's applicability to complex, noisy, or multi-objective optimization tasks.

Declarations

Availability of Supporting Data

All data generated or analyzed during this study are included in this published paper.

Funding

The authors conducted this research without any funding, grants, or support.

Competing Interests

The authors declare that they have no competing interests relevant to the content of this paper.

Authors' Contributions

Michael Oluwaseun Ayansiji contributed to the conceptualization, methodology design, formal analysis, and drafting of the manuscript. Friday Zinzendoff Okwonu was responsible for data curation, visualization, and validation of the results.

References

[1] Amini, A., Dolatshahi, M., Kerachian, R. (2023). "Effects of automatic hyperparameter tuning on the performance of multi-variate deep learning-based rainfall nowcasting". *Water Resources Research*, 59(6), doi:https://doi.org/10.1029/2022WR032789.

[2] Barthelmann, V., Novak, E., Ritter, K. (2000). "High dimensional polynomial interpolation on sparse grids". *Advances in Computational Mathematics*, 12(4), 273-288, doi:https://doi.org/10.1023/A:1018977404843.

[3] Bergstra, J., Bengio, Y. (2012). "Random search for hyper-parameter optimization". *Journal of Machine Learning Research*, 13, 281-305.

[4] Bull, A.D. (2011). "Convergence rates of efficient global optimization algorithms". *Journal of Machine Learning Research*, 12(88), 2879-2904, doi:https://doi.org/10.48550/arXiv.1101.3501.

[5] Ciresan, D.C., Meier, U., Schmidhuber, J. (2012). "Multi-column deep neural networks for classifying images". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3642-3649, doi:https://doi.org/10.1109/CVPR.2012.6248110.

[6] Conn, A.R., Scheinberg, K., Vicente, L.N. (2009). "Introduction to derivative-free optimization". *Society for Industrial and Applied Mathematics (SIAM)*, doi:https://epubs.siam.org/doi/book/10.1137/1.9780898718768.

[7] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J.T., Blum, M., Hutter, F. (2019). "Auto-sklearn: Efficient and robust automated machine learning". In *Automated Machine Learning: The*

Springer Series on Challenges in Machine Learning (pp. 113-134). Springer, doi:https://doi.org/10.1007/978-3-030-05318-5_6.

[8] Hong, H. (2024). "Landslide susceptibility assessment using locally weighted learning integrated with machine learning algorithms". *Expert Systems with Applications*, 237(Part C), doi:https://doi.org/10.1016/j.eswa.2023.124678.

[9] Hutter, F., Hoos, H.H., Leyton-Brown, K. (2011). "Sequential model-based optimization for general algorithm configuration". In *Learning and Intelligent Optimization (LION 2011)*, 6683, 507-523, doi:https://doi.org/10.1007/978-3-642-25566-3_40.

[10] Ilemobayo, J.A., Durodola, O., Alade, O., Awotunde, O.J., Olanrewaju, A.T., Falana, O., Ogungbire, A., Osinuga, A., Ogunbiyi, D., Ifeanyi, A., Odezuligbo, I.E., Edu, O.E. (2024). "Hyperparameter tuning in machine learning: A comprehensive review". *Journal of Engineering Research and Reports*, 26(6), 388-395, doi:https://doi.org/10.9734/jerr/2024/v26161188.

[11] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A. (2018). "Hyperband: A novel bandit-based approach to hyperparameter optimization". *Journal of Machine Learning Research*, 18(185), 1-52, doi:https://doi.org/10.48550/arXiv.1603.06560.

[12] Liu, H., Ong, Y.-S., & Cai, J. (2021). "Large-scale heteroscedastic regression via Gaussian process". *IEEE Transactions on Neural Networks and Learning Systems*, 32(2), 708-721, doi:https://doi.org/10.1109/TNNLS.2020.2979188.

[13] Maclaurin, D., Duvenaud, D., Adams, R.P. (2015). "Gradient-based hyperparameter optimization through reversible learning". *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 37, 2113-2122, http://proceedings.mlr.press/v37/maclaurin15.html

[14] Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H., Mirjalili, S.M. (2017). "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems". *Advances in Engineering Software*, 114, 163-191, doi:https://doi.org/10.1016/j.advengsoft.2017.07.002.

[15] Roohi, M., Mirzajani, S., Haghighi, A.R., Basse-O'Connor, A. (2024). "Robust stabilization of fractional-order hybrid optical system using a single-input TS-fuzzy sliding mode control strategy with input nonlinearities". *AIMS Mathematics*, 9(9), 25879-25907, doi:https://doi.org/10.3934/math.20241264.

[16] Tiep, N.H., Jeong, H.-Y., Kim, K.-D., Xuan Mung, N., Dao, N.-N., Tran, H.-N., Hoang, V.-K., Ngoc Anh, N., Vu, M.T. (2024). "A new hyperparameter tuning framework for regression tasks in deep neural network: Combined-sampling algorithm to search the optimized hyperparameters". *Mathematics*, 12(24), 3892, doi:https://doi.org/10.3390/math12243892.

[17] Wahba, G. (1990). "Spline models for observational data". *Society for Industrial and Applied Mathematics*, doi:https://doi.org/10.1137/1.9781611970128.

[18] Yin, L., Liu, J., Fang, Y., Gao, M., Li, M., Zhou, F. (2023). "Two-stage hybrid genetic algorithm for robot cloud service selection". *Journal of Cloud Computing*, 12, 95, doi:https://doi.org/10.1186/s13677-023-00458-y.

[19] Zhou, J., Wang, P. (2023). "Retraction note: Image simulation of urban landscape in coastal areas based on geographic information system and machine learning". *Neural Computing and Applications*, 35, 3577, doi:https://doi.org/10.1007/s00521-022-08145-w.