mathematics

MDPI

Article

A New Hyperparameter Tuning Framework for Regression Tasks in Deep Neural Network: Combined-Sampling Algorithm to Search the Optimized Hyperparameters

Nguyen Huu Tiep [1,2,*,†], Hae-Yong Jeong [1,†], Kyung-Doo Kim [3], Nguyen Xuan Mung [4], Nhu-Ngoc Dao [5], Hoai-Nam Tran [6], Van-Khanh Hoang [6,†], Nguyen Ngoc Anh [6,†] and Mai The Vu [7]

[1] Department of Quantum and Nuclear Engineering, Sejong University, 209, Neungdong-ro, Gwangjin-gu, Seoul 05006, Republic of Korea; hyjeong@sejong.ac.kr
[2] Institute for Nuclear Science and Technology (INST), Vietnam Atomic Energy Institute (VINATOM), 179 Hoang Quoc Viet, Cau Giay, Hanoi 100000, Vietnam
[3] Korea Atomic Energy Research Institute (KAERI), 111, Daedeok-daero 989beon-gil, Yuseong-gu, Daejeon 34057, Republic of Korea; kdkim@kaeri.re.kr
[4] Faculty of Mechanical and Aerospace Engineering, Sejong University, 209, Neungdong-ro, Gwangjin-gu, Seoul 05006, Republic of Korea; xuanmung@sejong.ac.kr
[5] Department of Computer Science and Engineering, Sejong University, 209, Neungdong-ro, Gwangjin-gu, Seoul 05006, Republic of Korea; nndao@sejong.ac.kr
[6] Phenikaa Institute for Advanced Study (PIAS), PHENIKAA University, Hanoi 12116, Vietnam; nam.tranhoai@phenikaa-uni.edu.vn (H.-N.T.); khanh.hoangvan@phenikaa-uni.edu.vn (V.-K.H.); anh.nguyenngoc1@phenikaa-uni.edu.vn (N.N.A.)
[7] Department of Artificial Intelligence and Robotics, Sejong University, 209, Neungdong-ro, Gwangjin-gu, Seoul 05006, Republic of Korea; maithevu90@sejong.ac.kr
* Correspondence: tiepngh@sejong.ac.kr
† These authors contributed equally to this work.

Citation: 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. A New Hyperparameter Tuning Framework for Regression Tasks in Deep Neural Network: Combined-Sampling Algorithm to Search the Optimized Hyperparameters. *Mathematics* **2024**, *12*, 3892. https://doi.org/10.3390/math12243892

Academic Editors: Ravil Muhamedyev and Evgeny Nikulchev

Received: 31 October 2024
Revised: 7 December 2024
Accepted: 9 December 2024
Published: 10 December 2024

Abstract: This paper introduces a novel hyperparameter optimization framework for regression tasks called the Combined-Sampling Algorithm to Search the Optimized Hyperparameters (CASOH). Our approach enables hyperparameter tuning for deep learning models with two hidden layers and multiple types of hyperparameters, enhancing the model's capacity to work with complex optimization problems. The primary goal is to improve hyperparameter tuning performance in deep learning models compared to conventional methods such as Bayesian Optimization and Random Search. Furthermore, CASOH is evaluated alongside the state-of-the-art hyperparameter reinforcement learning (Hyp-RL) framework to ensure a comprehensive assessment. The CASOH framework integrates the Metropolis-Hastings algorithm with a uniform random sampling approach, increasing the likelihood of identifying promising hyperparameter configurations. Specifically, we developed a correlation between the objective function and samples, allowing subsequent samples to be strongly correlated with the current sample by applying an acceptance probability in our sampling algorithm. The effectiveness of our proposed method was examined using regression datasets such as Boston Housing, Critical heat flux (CHF), Concrete compressive strength, Combined Cycle Power Plant, Gas Turbine CO, and NOx Emission, as well as an 'in-house' dataset of lattice-physics parameters generated from a Monte Carlo code for nuclear fuel assembly simulation. One of the primary goals of this study is to construct an optimized deep-learning model capable of accurately predicting lattice-physics parameters for future applications of machine learning in nuclear reactor analysis. Our results indicate that this framework achieves competitive accuracy compared to conventional random search and Bayesian optimization methods. The most significant enhancement was observed in the lattice-physics dataset, achieving a 56.6% improvement in prediction accuracy, compared to improvements of 53.2% by Hyp-RL, 44.9% by Bayesian optimization, and 38.8% by random search relative to the nominal prediction. While the results are promising, further empirical validation across a broader range of datasets would be helpful to better assess the framework's suitability for optimizing hyperparameters in complex problems involving high-dimensional parameters, highly non-linear systems, and multi-objective optimization tasks.

Keywords: linear regression; hyperparameters optimization; deep neural network; lattice-physics; combined sampling methods

MSC: 68T07

1. Introduction

Artificial intelligence (AI) applications in nuclear reactors and safety analysis enhance the safety and economics of this reactor technology [1–3]. These applications have emerged due to their promising yet challenging tasks. The promise lies in the AI model's ability to reduce computation time significantly. However, a key challenge is that the predictions of the AI model currently for the numerical analysis are highly uncertain due to the diverse dataset and lack of experimental demonstration; almost all the trained datasets come from the numerical analysis [4–6]. Additionally, predicting core and lattice-physics parameters with high accuracy is particularly challenging due to the high dimensionality of variables and the complex physical phenomena resulting from particle interactions in nuclear fission reactions, especially those simulated by the Monte Carlo method. Therefore, ensuring that the AI models used to predict the core and lattice-physics parameters are up-to-date and strongly efficient with low uncertainties is crucial. Consequently, research is focused on enhancing safety features and improving fuel design performance, with specific emphasis on optimizing lattice-physics parameters [7–10].

The provision of high-fidelity predictions for lattice-physics parameters is crucial and has greatly facilitated the optimization of nuclear fuel design. This optimization process involves simulating the interaction of neutrons with fuel and non-fuel materials within a lattice of fuel assemblies. Accurate prediction and low computation costs of these behaviors are essential to ensure the economy, safety, and efficient performance of nuclear fuel design. However, traditional computer codes commonly employed for predicting lattice physics, such as Monte Carlo N-particle neutron transport code (MCNP6) [11], a comprehensive neutronic calculation code system (SRAC2006) [12], standard computational analysis for licensing evaluation (SCALE) [13], and an open-source Monte Carlo particle transport simulation code (OpenMC) [14], tend to be expensive, particularly when conducting an extremely large number of calculations. Furthermore, due to the lack of measured data for these parameters, code-to-code comparison is often adopted to verify predictions. This approach, although useful, can introduce significant uncertainties.

To overcome the above limitations, in the past few decades, machine learning (ML) has been increasingly utilized in the field of lattice-physics prediction due to its ability to provide precise predictions after being trained by massive datasets. For instance, the core parameters of the beginning of the cycle (BOC) were predicted using the backpropagation network (BPN) [15]. They found that this neural network is a few hundred times faster than the reference numerical computer code, but the accuracy is not satisfied due to a lack of systematic selection of training patterns. Moreover, using the data of control rod positions and signals from the ex-core detectors to generate the trained datasets for the neural network, the in-core power peak factor was well predicted [16]. Although the number of these datasets was small, they obtained a similar accuracy compared with in-core detectors for the core power peak factor estimations. Furthermore, a neural network architecture based on computer vision and modular neural network approaches that convert the lattice-physics parameters into an imagelike format to train the neural network is implemented [17]. The results are in good agreement with the prediction of normalized pin powers and k-eff, even with the presence of burnable absorber materials. However, the selection of hyperparameters was primarily based on their experiences and trial-and-error runs, which can significantly affect the accuracy of the deep neural networks (DNNs) model.

In this context, hyperparameter optimization in ML has become essential in the field of machine learning due to its direct influence on model accuracy, efficiency, and general-

ization capabilities across applications. The objective of hyperparameter optimization is to fine-tune parameters that control model behavior, which, when optimized, can significantly improve a model's predictive performance. This process has gained prominence as the complexity of machine learning models and data scales continues to grow, leading to numerous proposed optimization methods, from basic grid and random searches to sophisticated Bayesian, genetic algorithm, and reinforcement learning techniques.

In DNNs, the selection of hyperparameters is crucial for achieving optimal model performance, generalization, robustness, computational efficiency, interpretability, and reproducibility. Properly selecting hyperparameters ensures that the model is fine-tuned to the specific problem at hand, leading to accurate and reliable predictions. Through meticulous fine-tuning of hyperparameters using methodologies such as grid search, random search [18], Bayesian optimization [19,20], and Optuna optimization [21], one can achieve the identification of an optimal configuration that adeptly balances the intricate interplay between model complexity and accuracy. For example, Bayesian optimization is used to automatically optimize the hyperparameters such as the number of hidden layers, number of nodes per layer, batch size, learning rate, and learning rate decay for deep neural networks to predict engine-out NOx emissions [22]. The results of Bayesian optimization are also compared with the grid search and random search methods. It revealed that the accuracy of the model that uses the Bayesian optimization method has increased. However, the implementation of Bayesian optimization in Ref. [22] deployed the number of nodes in every hidden layer to be equal to the number of nodes in the first hidden layer, which can lead to the limited search space of the optimization problem, resulting in insufficient reliability outcome. Consequently, optimizing hyperparameters is crucial for any machine learning model, necessitating the use of efficient methods to sample and find optimal hyperparameter values. In this context, many efficient optimization frameworks have been developed, such as Autotune [23], Google Vizier [24], and Tune [25].

The optimization of hyperparameters is crucial for machine learning model performance, with techniques ranging from conventional grid and random search to advanced methods like Bayesian optimization, genetic algorithms, and reinforcement learning. Studies have demonstrated various approaches to enhance tuning, such as model-based optimization for random forests, where the benchmark showed better performance than standard methods [26]. Some research has found that default parameters can outperform limited-iteration tuning [27], while others focus on the impact of parameters like momentum on model performance [28]. Surrogate-based ranking methods [29], cost-effective performance measures for support vector machine [30], and the under-reporting of hyperparameter values in research [31] are also significant findings.

Therefore, in this study, we developed a new method to optimize the hyperparameters, the so-called Combined-sampling Algorithm to Search the Optimized Hyperparameters (CASOH) to find the best combinations of the hyperparameters for the DNNs that were deployed. We validated this new framework using the regression conventional datasets in University of California, Irvine (UCI) Machine Learning Repository and Kaggle database such as Boston housing [32], critical heat flux (CHF) in water-cooled tube [33], Concrete compressive strength [34], Combined Cycle Power Plant [35], Gas Turbine CO and NOx Emission [36], and the 'in-house' high-fidelity lattice-physics parameters dataset [9]. The efficiency of our proposed method will be compared with the performance of the random search, the Bayesian optimization, and the Hyperparameter Reinforcement Learning (Hyp-RL) optimization method [37]. The main contributions of this paper are as follows:

1. Development of a sensitivity analysis framework to identify the optimal ranges for hyperparameters in deep learning models, providing critical insights into the structure and behavior of the DNN model. This analysis enhances our understanding of the hyperparameters' impact on model performance;
2. Introduction of the Combined-Sampling Algorithm to Search the Optimized Hyperparameters (CASOH), a novel method for hyperparameter optimization. CASOH significantly improves model accuracy and computational efficiency compared to

traditional methods such as random search and Bayesian optimization. In addition, CASOH is evaluated against the state-of-the-art reinforcement learning-based hyperparameter optimization method, Hyp-RL;

3. Validation of the CASOH method on six diverse datasets, including the conventional regression datasets Boston Housing from Kaggle, the CHF dataset in water-cooled tubes from U.S Nuclear Regulatory Commission, Concrete compressive strength, Combined Cycle Power Plant, Gas Turbine CO and NOx Emission and an in-house dataset for lattice-physics parameters in nuclear fuel analysis. The results demonstrate the superiority of CASOH in optimizing deep learning models across various domains.

2. Related Work

Hyperparameter optimization has become a critical area of research in machine learning, as tuning these parameters significantly impacts model performance across various applications. Numerous techniques have been proposed for hyperparameter optimization, ranging from traditional grid and random search methods to more advanced algorithms such as Bayesian optimization, genetic algorithms, and reinforcement learning-based approaches. For example, in Ref. [26], the authors applied model-based optimization strategies to tune hyperparameters for the random forest (RF) algorithm. They focused on parameters such as the number of observations sampled per tree, split criteria, minimum node samples, and tree count. They indicated that their benchmark study analysis achieved, on average, better performances than the standard random forest and other software that implement tuning for random forest.

In Ref. [27], a methodology was developed to evaluate the importance of hyperparameter tuning using non-inferiority tests and tuning risk, enabling the selection of effective default parameters. Benchmarking 59 datasets from open machine learning, the authors found that, in some cases, default values outperformed limited-iteration tuning processes. Meanwhile, Ref. [28] re-examined common hyperparameter practices in fine-tuning, revealing that parameters like momentum impact performance and are influenced by the similarity between source and target domains.

In reference Ref. [29], they proposed a generic method that leverages previous experimental data to improve hyperparameter tuning on new tasks, combining surrogate-based ranking and optimization to outperform standard techniques and single-problem optimizations. Similarly, Ref. [30] empirically evaluated cost-effective performance measures, finding that k-fold cross-validation reliably estimates support vector machine (SVM) generalization error. In Ref. [31], a survey revealed that only 20.31% of machine learning papers explicitly report final hyperparameter values, with 53% omitting tuning information altogether, highlighting a potential issue of model performance misinterpretation due to inadequate tuning practices.

Addressing complex deep learning models, Ref. [38] highlighted the challenge of hyperparameter optimization for convolutional neural networks (CNNs) and discussed the prevalent "guessing" approach for tuning. Ref. [39] examined spatial autocorrelation's effect on tuning and performance estimation across models like RF, SVM, and boosted regression trees, advocating spatial tuning to avoid over-optimistic predictions, especially in ecological modeling. In Ref. [40], a genetic algorithm was used to automate neural network hyperparameter tuning, accelerating configuration discovery compared to naïve approaches.

Further, Ref. [41] explored enhancements to Bayesian optimization for deep neural networks through their DEEP-BO framework, which incorporates early termination, parallel computation, and cost function transformation, demonstrating robustness across six DNN benchmarks. In Ref. [42], black-box optimization algorithms like Tree-structured Parzen Estimator (TPE), Nelder-Mead, and Particle Swarm Optimization were evaluated, with TPE achieving the best balance in solution quality, improvement speed, and consistency. Moreover, Ref. [43] implemented gradient-based techniques to optimize hyperparameter initialization, offering effective long-term tuning solutions.

Furthermore, in Ref. [44], Bayesian optimization using Gaussian processes was employed for RF and neural networks, achieving high accuracy and reduced runtime compared to manual tuning. Eventually, the authors in Ref. [45] developed an optimization-based nested algorithm, known as the Generalized Weighted Ensemble with Internally Tuned Hyperparameters (GEM-ITH), which optimizes both hyperparameter tuning and the weights used to combine ensemble models. Their results demonstrated that GEM-ITH outperforms state-of-the-art ensemble creation methods. However, the tuning and model-building process remains time-consuming, particularly for ensemble methods that require training multiple models.

3. Materials and Methods

3.1. The Conventional and State-of-the-Art Method to Optimize the Hyperparameters

3.1.1. Random Search

The hyperparameter of a given ML model has direct control of its structure, function, and performance. Therefore, determining optimal values for hyperparameters, also known as hyperparameter tuning, is a crucial step to ensure the success of the ML model. Conventionally, there are several methods for hyperparameter tuning, in which the grid search is a fundamental method. Within the grid search approach, one first defines a space of hyperparameters and then creates a grid for search space to form a multidimensional grid, whose each point represents a unique combination of hyperparameter values. Though this approach works well in some simple cases, it has multiple drawbacks, especially when the dimensional of hyperparameter space is high, as the number of points for a multidimensional grid exponentially expands, consuming extremely expensive computational efforts.

A slightly improved version of the grid search is the random search. This approach defines probability distributions for ranges of hyperparameter values and evaluates the model performance corresponding to sets of hyperparameter values randomly selected based on the defined distributions. The process might stop when the model performance or the number of samples reaches a threshold value provided by users. Though this approach is much more efficient than the grid search, as demonstrated in Ref. [18], it is still extremely expensive in terms of computational costs. In addition, the choice of employed probability distributions is inconsistent, as it might vary from one user to the other without any specific reason. In addition, some of the hyperparameter values can form as the continuous value rather than the point in the grid. Therefore, the random search method seems to be more efficient than the grid search in terms of both the sampling method's efficiency and the computation costs. The difference between grid search and random search was also indicated in Ref. [18].

3.1.2. Bayesian Optimization

Bayesian optimization is a method to optimize the continuous domains with high parameter dimensions. This approach constructs a surrogate model to estimate the objective function and then uses a defined acquisition function to determine the next sample [46].

Recently, Bayesian Optimization has been implemented to turn hyperparameters in machine learning by automating the process of finding optimal configurations of the hyperparameters, especially in deep neural networks [20]. Hyperparameters play a crucial role in model performance but require manual specification and tuning. Bayesian Optimization employs probabilistic models, like Gaussian Processes, to efficiently navigate the hyperparameter space and intelligently select configurations for evaluation. With its ability to balance exploration and exploitation, Bayesian Optimization minimizes the number of model evaluations required, making it an efficient and effective approach for optimizing machine learning models.

The fundamental idea of the Bayesian Optimization approach is to find the optimized value of F value corresponds with the variable X under set A [19]:

$$F = \max_{X \in A \subset R^d} \{f(X)\} \quad (1)$$

where d is the dimensions of the problem domains; $f(X)$ is the continuous function that is normally observed without noise.

In this implementation, $f(X)$ was related to the accuracy function of deep neural network prediction. Therefore, $f(X)$ can be non-convex, highly non-linear, noisy, and expensive to evaluate. The value of $f(X)$ will be estimated using the Bayesian Optimization method to turn the hyperparameters into our deep neural networks model, such as the learning rate, the number of nodes per hidden layer, the optimizer, and the activation function.

3.1.3. Hyp-RL Optimization Method

The Hyp-RL framework applies reinforcement learning principles to optimize hyperparameters for deep learning models [37]. It operates within a continuous action space, where an agent samples hyperparameter values such as learning rate, epochs, batch size, and layer sizes from predefined ranges. The framework employs an exploration-exploitation strategy, iteratively refining hyperparameter selection over successive episodes. Negative mean squared error (MSE) serves as the reward signal, guiding the agent toward configurations that minimize prediction error while balancing computational efficiency. This approach is particularly effective for navigating high-dimensional hyperparameter spaces and solving complex, non-linear regression tasks.

In this method, the hyperparameter optimization problem is modeled as a sequential decision-making process. Hyp-RL adjusts its sampling strategy dynamically, selecting the next hyperparameter configurations based on reinforcement learning. Unlike Bayesian optimization, which relies on a heuristic acquisition function, Hyp-RL learns directly from the observed reduction in validation loss. This allows the framework to identify prospective hyperparameter configurations by analyzing trends in validation loss reduction, further improving its ability to find optimal solutions in challenging optimization landscapes.

3.2. The Proposed Method

In this Section, we describe our combined sampling method for implementation in the optimization process. We implemented uniform samples and continuous samples, which can include all types of hyperparameters, such as categorical and non- categorical value hyperparameters.

We utilized three sample types: uniform samples, continuous samples, and categorical samples denoted as x, y, and z. For example, x can be randomly adjusted within a range from a minimum value (a) to a maximum value (b) as follows:

$$x \sim U(a, b) \tag{2}$$

where $U(a, b)$ is the random number from a to b with uniform distribution. The values of a or b can encompass either irrational or integer values, corresponding respectively to continuous or discrete ranges.

The continuous sample will be as follows:

$$y = x + U(-\delta, \delta) \tag{3}$$

where δ is the small value, the so-called step size. Therefore, y can be very close to the x position if the value of x is identified. The term 'continuous' characterizes samples intentionally fashioned to closely align with a reference value of uniform samples x while also introducing controlled and minor random variations defined within the range $U(-\delta, \delta)$. This terminology effectively emphasizes proximity to the original value x while also indicating the justified and stochastic nature of the introduced variations. In this context, the continuous sampling method is not applicable to hyperparameters such as optimizers and activators, which are characterized by textual rather than numerical values. This discrepancy prompted the introduction of categorical samples, as outlined below:

$$z = \{S\} \tag{4}$$

The categorical sample z is represented by set S, where S is the set of predefined character combinations. The uniform distribution is also employed to obtain the value of z in the set S. These three variables, x, y, and z, are used to assign values to different types of hyperparameters.

We categorized the hyperparameters into two different types: (1) non-categorical hyperparameters and (2) categorical hyperparameters (see Table 1). The non-categorical hyperparameters are the learning rate and number of nodes per hidden layer, which can be obtained as a real or integer value. These hyperparameters are adjusted within a very small step size δ that facilitates finding the better system states around the current system state position.

Table 1. The structure and range of hyperparameters.

No.	Quantity	Range	Type
1	Learning rate	L_1 to L_2	non-categorical
2	Number of hidden layers	$\{S_h\}$	categorical
3	Number of nodes per layer	N_1 to N_2	non-categorical
4	Activator	$\{S_a\}$	categorical
5	Optimizer	$\{S_o\}$	categorical

In contrast, the categorical hyperparameters are the number of hidden layers, activation function, and optimizer, which is referred to the selection of a fixed integer number or a categorical variable with choices such as 'relu' or 'linear' in a user-defined set. A specific combination of the categorical hyperparameters can be assumed as a system starting point of a multidimensional convex function in which adjacent positions of the optimal value of the loss function, a so-called local minimum peak, can be found [47] using continuous samples. This, in turn, highlights the necessity for combining sampling algorithms in the CASOH framework.

Mathematically, in our sampling algorithm, a sample does not fit the traditional definition of a vector or a matrix due to its varying dimensions and structures. It is more like a collection of vectors, where each element can have a different type or structure. This type of data structure is common in situations where diverse types of data or measurements do not have a consistent dimensionality. In this context, a uniform sample X can have the structures as follows:

$$X = \left(x_L, z_H, \vec{x_N^k}, \vec{z_A^k}, z_O \right) \tag{5}$$

where each element represents different hyperparameters or structures in the system, such as

Learning rate: $x_L = U(L_1, L_2)$ is learning rate with $U(L_1, L_2)$ denotes a uniform distribution within the specified bounds; L_1 and L_2 are the user-defined learning rate range ($L_1, L_2 \in \mathbb{R}^+$).

Number of hidden layers: $z_H = \{S_h\} = k$ is the number of hidden layers chosen from the set $\{S_h\}$ and k is a selected value for the number of hidden layers.

Vector of nodes per layer: $\vec{x_N^k} = \left(U^1(N_1, N_2),\ U^2(N_1, N_2) \ldots, U^k(N_1, N_2) \right)$ represents the number of nodes per layer, where each element is independently sampled from a uniform distribution $U(N_1, N_2)$; Here $N_1, N_2 \in \mathbb{Z}^+$ specify the range for the number of nodes per layer.

Activation functions per layer: $\vec{z_A^k} = \left(\{S_a\}^1,\ \{S_a\}^2 \ldots, \{S_a\}^k \right)$ is a vector with activation functions in all hidden layers.

Optimizer: $z_O = \{S_o\}$ is optimizer and chosen from a set $\{S_o\}$ of valid optimizers.

Subsequently, the continuous sample Y would be:

$$Y = \left(y_L, z_H, \vec{y_N^k}, \vec{z_A^k}, z_O \right) \tag{6}$$

$$y_L = x_L + U(-\delta_L, \delta_L) \tag{7}$$

$$\vec{y}_N^k = \vec{x}_N^k + U^k(-\vec{\delta}_N, \delta_N) \tag{8}$$

where δ_L is the learning rate step size; δ_N is the number of node step sizes. From this perspective, when a uniform sample is generated, a continuous sample can be defined based on this current sample, and only the non-categorical hyperparameters can be adjusted. The types of hyperparameters are displayed in Table 1.

In this sampling algorithm, on the one hand, the continuous samples facilitate the discovery of local minimum peaks by transitioning to neighboring better system states within a small step size. On the other hand, uniform samples are utilized to identify superior local minimum peaks. By implementing this combination, it was anticipated that the accuracy would be significantly enhanced after a few iterations (see Figure 1). A detailed explanation of the CASOH combined sampling algorithm is provided in Supplementary Materials File S1.

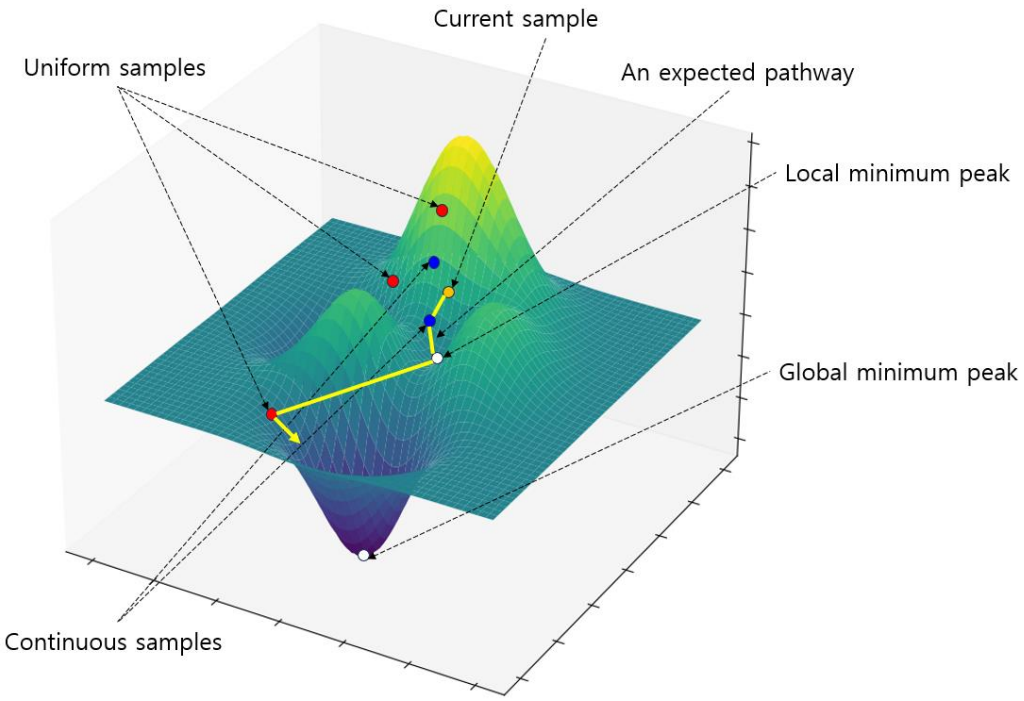

Figure 1. A virtual sampling process of the CASOH methodology.

The generation procedure of the uniform and continuous samples in the CASOH framework (see Algorithm 1):

1. Generating a uniform sample

STEP 1. Draw the learning rate;
STEP 2. Draw the number of hidden layers;
STEP 3. Draw the optimizer;
STEP 4. Based on the number of hidden layers generated in STEP 2, draw the number of nodes and activators per hidden layer.

2. Generating a continuous sample

STEP 1. Identify the learning rate, the number of hidden layers, and the corresponding number of nodes per layer of the previously accepted sample;
STEP 2. Draw a new learning rate based on the previous-accepted value—using variable y in Equation (3);
STEP 3. Draw a new number of nodes per hidden layer based on the previous-accepted sample—using variable y in Equation (3).

Figure 2 illustrates the CASOH sampling algorithm. In this algorithm, we combined the Metropolis–Hasting algorithm and the Random search sample, which corresponded to the continuous sample and uniform sample [47]. After generating the first sample, the cost function (the accuracy of the ML prediction) for this sample will be calculated (F_a); this first sample will be considered as an accepted sample $F_a(x_a)$. Subsequently, we simultaneously draw both kinds of samples and estimate their corresponding cost functions (F_i). These cost functions will be used to compare with the currently accepted cost function ($\alpha_i = F_i/F_a$). Then, the minimum value $\alpha_{min} = \min\{1, \alpha_i\}$ is identified, which can be used to compare with a random number u = rand(β, 1). If α_{min} is equal or greater than u, it would be accepted; otherwise, it would be rejected. Thereby, the new cost function value (F_a) can be updated after accepting a sample.

Algorithm 1. CASOH Algorithm

1. Draw and investigate the first accepted sample
2. For i = 1, 2, . . ., n do
   Read the current accepted sample $F_a(X_a)$
   Draw the continuous samples based on the accepted sample $F_a(X_a)$ -> Equations (7) and (8)
   Draw the uniform samples -> Equation (5)
   Calculate their corresponded loss functions
   Identify the minimum loss functions and denote as $F_a(X_a)$
   Calculate the $\alpha_i = \frac{F_i}{F_a}$ and identify the $\alpha_{min} = \min\{1, \alpha_i\}$
   Draw the u = rand(β, 1)
   If $\alpha_{min} \geq u$, accept this sample and update the current accepted sample
   Else, continue $i = i + 1$
3. End for

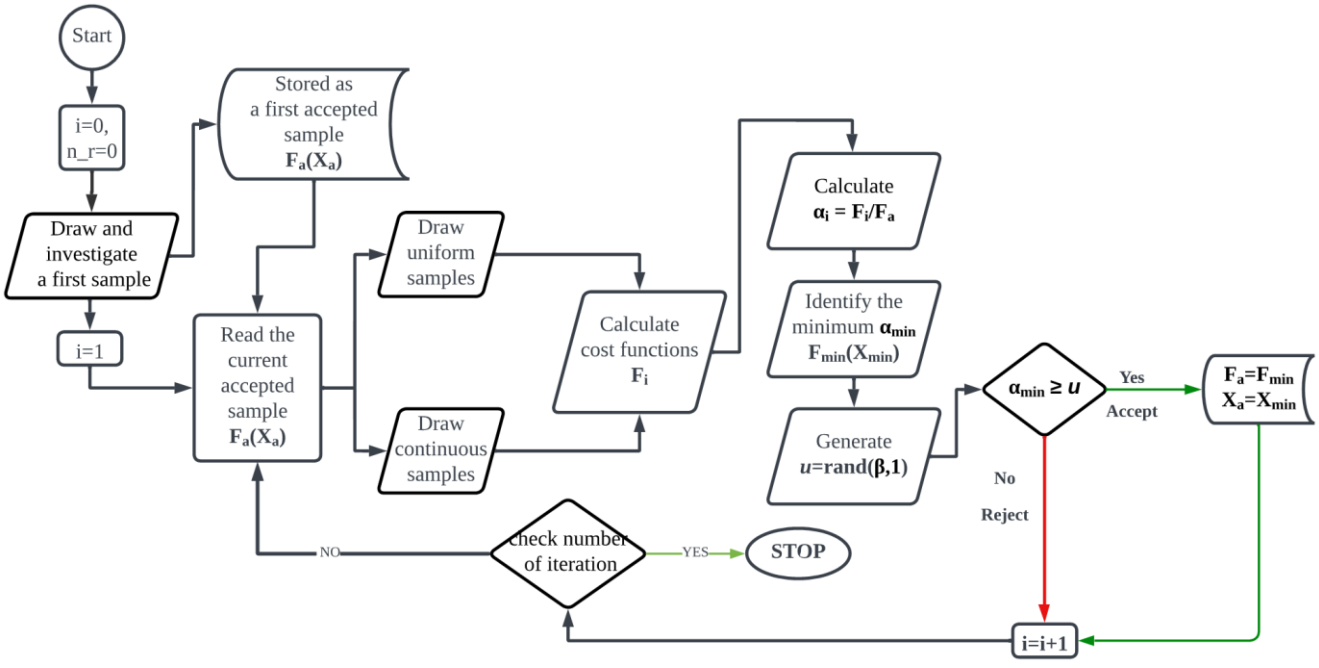

Figure 2. The CASOH sampling algorithm.

One of the advantages of our sampling algorithm is that the acceptance rate can be justified by changing the value of β. However, if the value of β is too small, the results can fluctuate due to the acceptance of bad candidates. Conversely, if the value of β is very close to 1, along with a very small step size of the continuous sample, the system may either slowly converge or cannot escape the local minimum peak. Therefore, selecting β needs several trials and errors of testing to preliminary estimate the system behaviors.

3.3. Mathematical Model of Proposed Method

In this optimization problem, let $f_0 :\rightarrow \mathbb{R}$ represent the objective function that needs to be minimized. We defined $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ as the vector of optimization variables; $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ as the constraint functions, where each function satisfies the inequality:

$$f_i(x) \leq b_i \text{ for } i = 1, \ldots, m \tag{9}$$

where b_i denotes the boundary of each $f_i(x)$. If a vector x^* is called optimal if it minimizes the objective function $f_0(x)$ subject to the constraints $f_i(x) \leq b_i$ for all i. Mathematically, x^* is optimal if, for any vector z satisfying $f_i(z) \leq b_i$ for all $i = 1, \ldots, m$, we have:

$$f_0(z) \geq f_0(x^*) \tag{10}$$

If x^* is feasible and $f_0(x^*) = p^*$, where p^* is the minimum value, the set of all optimal solutions, or the optimal set, is given by the following:

$$X_{optimal} = \{x \in \mathbb{R}^n \mid f_i(x) \leq b_i \text{ for } i = 1, \ldots, m \text{ and } f_0(x^*) = p^*\} \tag{11}$$

Therefore, if there exists an optimal value p^*, then the optimized value can be searchable, meaning $f_i(x)$ has a feasible minimum subject to the constraints; otherwise, the optimization problem is deemed infeasible or unbounded.

Given the setup of a continuous sample Y combined with a uniform distribution around it, as described in Equations (7) and (8), we can demonstrate the convergence of CASOH sampling algorithms.

Since the continuous variables are uniformly adjusted within step sizes δ, the next sample $x^{(t)}$ is generated based on the previous sample x, and the adjacent objective functions are convex. Due to the properties of convex functions or distributions, the objective function $f_0(x)$ may possess a unique local or global minimum within this uniform sample space.

Assume $\pi(x) \propto e^{-\mu f_0(x)}$ as the target distribution, where μ (analogous to inverse temperature) controls concentration around minima as $\mu \rightarrow \infty$. Let the proposal distribution $q(x^t|x)$ allow movement between any two states x and x' within the feasible region. We introduce the acceptance criteria $\alpha(x, x^t)$ as follows:

$$\alpha(x, x^t) = \min\left(1, \frac{\pi(x^t)q(x|x^t)}{\pi(x)q(x^t|x)}\right) \tag{12}$$

CASOH always saves the current accepted state for sampling the next movement, applying the acceptance criteria to jump to the next position justifiably. In particular, the chains will satisfy the detailed balance:

$$\pi(x)P(x \rightarrow x^t) = \pi(x^t)P(x^t \rightarrow x) \tag{13}$$

This balance condition implies that the chain has $\pi(x)$ as its stationary distribution, which is crucial for convergence. By the ergodic theorem for Metropolis–Hastings algorithm, if the chain is irreducible and aperiodic, then the distribution of x^t converges to $\pi(x)$ as $t \rightarrow \infty$.

The distribution $\pi(x)$ becomes increasingly concentrated around points where $f_0(x)$ is minimized. Therefore, let x^* is a local minimum value of $f_0(x)$. Then the $\pi(x)$ will concentrate around x^* because for any $x \neq x^*$:

$$\pi(x) = e^{-\mu f_0(x)} \ll e^{-\mu f_0(x^*)} \tag{14}$$

as $\mu \rightarrow \infty$ due to the exponential decay. Thus, as we increase μ, samples generated by the Metropolis–Hastings algorithm must be concentrated around x^* approximating the local minimum. However, if there are multiple distinct local minima widely spaced apart, a

single Metropolis–Hastings sample may become trapped at one peak, thereby limiting the opportunity to find a better solution.

In this context, by combining a Metropolis–Hastings sample with a random uniform sample (Equation (5)) which represents all possible solutions or samples, we ensure that the updated local minimum is consistently identified, ultimately converging to the optimal value within the search space. In the next Section, we demonstrate this convergence by proposing a multi-peak analytical function to evaluate the proposed sampling process.

3.4. Testing Proposed Sampling Method with a Multi-Peak Analytical Function

In this Section, we experimentally demonstrated the efficiency of the CASOH sampling algorithm using a multi-peak analytical target function (Equation (15)). We compared the results with those obtained using only one sample, the Metropolis–Hastings algorithm. The assumed analytical function is as follows:

$$y = 1 + e^{(-\frac{(x-0.8)^2}{0.1}-1)} + e^{(-\frac{(0.5x-0.8)^2}{0.0005}-1)} + e^{(-\frac{(0.3x-0.8)^2}{0.015}+0.2)} + e^{(-\frac{(0.1x-0.5)^2}{0.02}-1)} + e^{(-\frac{(0.09x-0.8)^2}{0.0005}+1.5)} \quad (15)$$

The analytical form of this target function is presented as the solid black line in Figure 3.

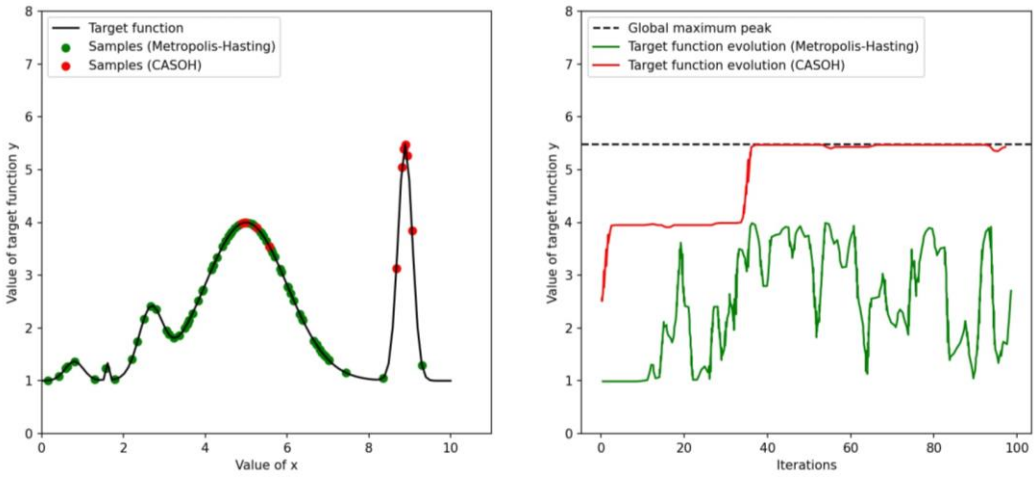

Figure 3. The comparison of sampling methods.

Consider the objective of determining the maximum value of 'y' across a range of 'x' values adjusting from 0 to 10. Through analytical analysis, it can be revealed that there are four local maximum peaks with $x_1 \approx 0.82$, $x_2 \approx 1.58$, $x_3 \approx 2.7$, $x_4 \approx 5.0$, respectively; and the global maximum of 'y' as 5.483, occurring at 'x' ≈ 8.889, as depicted in Figure 3.

In this investigation, we provided a multi-peak analytical function to compare the sampling algorithm of CASOH and a single Metropolis-Hasting sample, which generated a Markov chain by invoking a proposal density for the generation of novel samples and a mechanism for the rejection of suboptimal proposed samples. Given that $y(x)$ denotes the target distribution, and $P(x^* \mid x)$ represents the proposal distribution for the stochastic generation of a new variable x^* conditioned on the extant value x, the MH algorithm is amenable to endorsing the prospective candidate x^* in accordance with the acceptance probability clarified by Equation (12), now expressed as follows:

$$\alpha(x, x^*) = \min\left\{1, \frac{y(x^*).P(x \mid x^*)}{y(x).P(x^* \mid x)}\right\} \quad (16)$$

where $\frac{P(x|x^*)}{P(x^*|x)} = 1$ due to the symmetric proposal assumption. It will return the current 'x' value if α is less than the random variable 'u' drawn from a uniform distribution $U(0, 1)$. It

is worth noting that $U(0, 1)$ represents a uniform random variable ranging from 0 to 1, and the initial sample corresponds to 'x' = 0.

Figure 3 illustrates the comparison of the CASOH sampling method and an MH sample. It can be seen that after 100 iterations, the single Metropolis-Hasting sample is trapped at the fourth local maximum peak (x = 5), and the convergence fluctuates. In contrast, the CASOH method demonstrated that the optimized value of the target function can be achieved within 40 iterations. This leads to a robust conclusion that the CASOH approach exhibits greater efficiency when compared to the Metropolis–Hastings algorithm. However, it is worth noting that the computational cost of CASOH is higher due to the combination of sampling methods, which necessitates the generation and simultaneous evaluation of numerous samples. Nonetheless, this limitation can be overcome by deploying parallel computations.

3.5. Datasets

3.5.1. Boston Housing Dataset (BH)

The Boston Housing dataset [32] consists of 506 instances and 13 features, commonly used for regression tasks predicting housing prices in Boston's suburbs. Each feature captures a specific attribute of the area, influencing the housing price. This dataset is often employed for testing algorithms and learning regression techniques, as well as exploring the relationships between various socio-economic factors and housing prices. Several versions of this dataset are available on Kaggle, offering easy access for practitioners and researchers interested in real-world applications of predictive modeling.

3.5.2. Critical Heat Flux (CHF) Dataset

This CHF dataset presents a collection of more than 10,000 CHF data points from water-cooled tubes, which served as the foundation for developing the 2006 Groeneveld CHF lookup table [33]. The data compilation spans 62 datasets gathered over the past six decades. The NUREG report provides detailed descriptions of the experimental setups and highlights potential issues associated with these datasets. Additionally, it evaluates the applicability and reliability of the CHF lookup table for reactor conditions of interest and includes graphical comparisons illustrating the ranges covered by both the primary datasets and supplementary data collected later. This dataset includes seven features, and CHF is the output measured data of the experiment.

3.5.3. Concrete Compressive Strength (CCS)

This Concrete Compressive Strength dataset [34], available in the UCI Machine Learning Repository, is designed for regression tasks. It contains 1030 instances and nine attributes, representing the relationships between concrete's compressive strength (measured in MPa) and its components. The input variables include quantities of cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, fine aggregate, and the concrete's age (in days). The output variable is the measured compressive strength. This dataset provides insights into the non-linear effects of concrete composition on its strength, making it useful for predictive modeling and machine learning studies in civil engineering applications.

3.5.4. Combined Cycle Power Plant (CCPP)

The Combined Cycle Power Plant dataset [35] contains 9568 data points collected over six years (2006–2011) from a power plant operating at full load. The dataset is designed for regression tasks and includes four input features: ambient temperature, ambient pressure, relative humidity, and exhaust vacuum, alongside the target variable: net hourly electrical energy output of the plant. This dataset is frequently used in energy modeling and optimization studies, particularly to predict power output based on environmental and operating conditions.

3.5.5. Gas Turbine CO and NO_x Emission (NOX)

The Gas Turbine CO and NO_x Emission Dataset [36] are focused on modeling and predicting pollutant emissions—specifically carbon monoxide (CO) and nitrogen oxides (NO_x)—from gas turbines. This dataset, available on the UCI Machine Learning Repository, is widely used to develop predictive emission monitoring systems (PEMS). Such systems help reduce the financial costs of continuous emission monitoring while adhering to strict environmental regulations.

This dataset consists of 36,733 hourly instances collected from a gas turbine in Turkey's northwestern region. It includes 11 sensor measurements used to study flue gas emissions, such as CO and NO_x ($NO + NO_2$), along with turbine parameters like Turbine Inlet Temperature and Compressor Discharge Pressure. These features make it suitable for regression tasks, especially in analyzing the relationship between operating conditions and pollutant emissions. The data support research into developing interpretable models, such as symbolic regression and improving emission prediction accuracy using machine learning techniques.

3.5.6. The Lattice-Physics Dataset

In this Section, we provide a detailed description of the proprietary dataset used to train the deep neural network, as these data are not yet publicly available. We plan to make this dataset accessible through the UCI Machine Learning Repository in the future. The dataset included the lattice-physics parameters such as k-inf and pin power peaking factor (PPPF) depending on the adjustments of fuel pin enrichments of the NuScale US600 fuel assembly type C-01 (NFAC-01) (see Figure 4) [48]. The k-inf and PPPF were evaluated using the MCNP6 code for the criticality calculation. The propagated fuel enrichments of all fuel pins were generated by a uniform sampling method within the range 0.7–5.0 w/o of U-235.

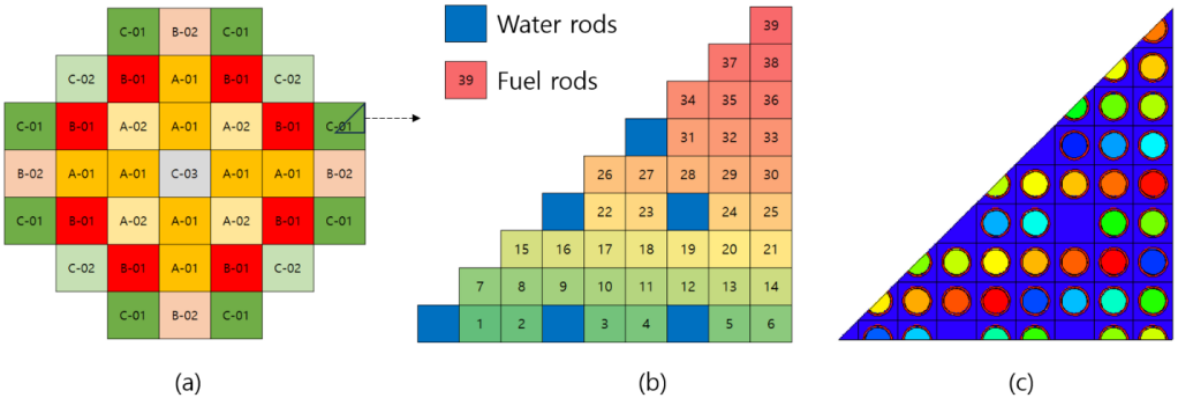

Figure 4. The Equilibrium core and fuel assembly of NuScale US600: (**a**) NuScale reactor core loading pattern for the equilibrium cycle [48]; (**b**) Cell numeration of 1/8 NFAC-01; (**c**) MCNP6 model of 1/8 NFAC-01.

For a single calculation, the criticality is run with 100,000 neutrons per cycle, including 250 active and 50 skipped cycles; the computation time is approximately 20 min for an AMD Ryzen Threadripper 3970X computer for a single calculation. The total number of samples collected was 24,360 calculations that consumed more than 8000 h of computation, which, however, was approximately reduced to 1600 h due to parallel computation. Consequently, we obtained the estimated standard deviation of k-inf calculation is about 12 pcm, and the tally relative error of the PPPF calculation is approximately 0.001, which is a significant small uncertainty in the criticality calculation of Monte Carlo code.

To generate a dataset for the machine learning model, we assumed that every fuel rod would have an adjustable enrichment of 235-U, which can generate the corresponding values of k-inf and PPPF for each combination of enrichments. For instance, the pin-by-pin

enrichments were denoted as $C_i = \{C_1, C_2, \ldots, C_n\}$, with n being the total number of fuel rods ($n = 39$), were considered as the adjustable variables to calculate the lattice-physics parameters; all the fuel pins were allowed to have their unique enrichments with the range from 0.7–5.0 w/o. It is worth noting that the total number of fuel rods in fuel assembly type C-01 that needed to be considered is $17 \times 17 = 289$, which, however, can be deducted to 39 due to the 1/8 symmetry of the configuration (see Figure 4).

We implemented uniform sampling for all the fuel pin enrichments under their uncertainty range to avoid the missing data zone, leading to a biased distribution of the MCNP6 predictions. For instance, Figure 5 indicates the pairwise correlation of C_i and C_j along with i and j from 1 to 39. It can be observed that all these distributions were uniform due to random variables for each sample also having uniform distribution from 0.7 to 5.0 (see Figure 6). Therefore, the histogram of C1 to C39 is approximately uniform. Although the distribution of variables is uniform, the values of k-inf and PPPF are observed in the form of the Gaussian distribution (see Figures 7 and 8). These data were plotted using 24,000 uniform samples and their corresponding value of k-inf and PPPF. There are 360 test samples that were used to evaluate the accuracy of the DNN prediction for the lattice-physics parameters, such as k-inf and PPPF (see Figures 9 and 10). The value of PPPF was evaluated based on all the pin power factors in the 1/8 NFAC-01. Assume $f_i = \{f_1, f_2, \ldots, f_n\}$ is the relative pin power factor for the i-th fuel rod. The PPPF can be determined as follows:

$$\text{PPPF} = \max\left\{ \frac{f_1}{\left(\frac{f_1+f_2+\ldots+f_n}{n}\right)}, \frac{f_2}{\left(\frac{f_1+f_2+\ldots+f_n}{n}\right)}, \ldots, \frac{f_n}{\left(\frac{f_1+f_2+\ldots+f_n}{n}\right)} \right\} \tag{17}$$

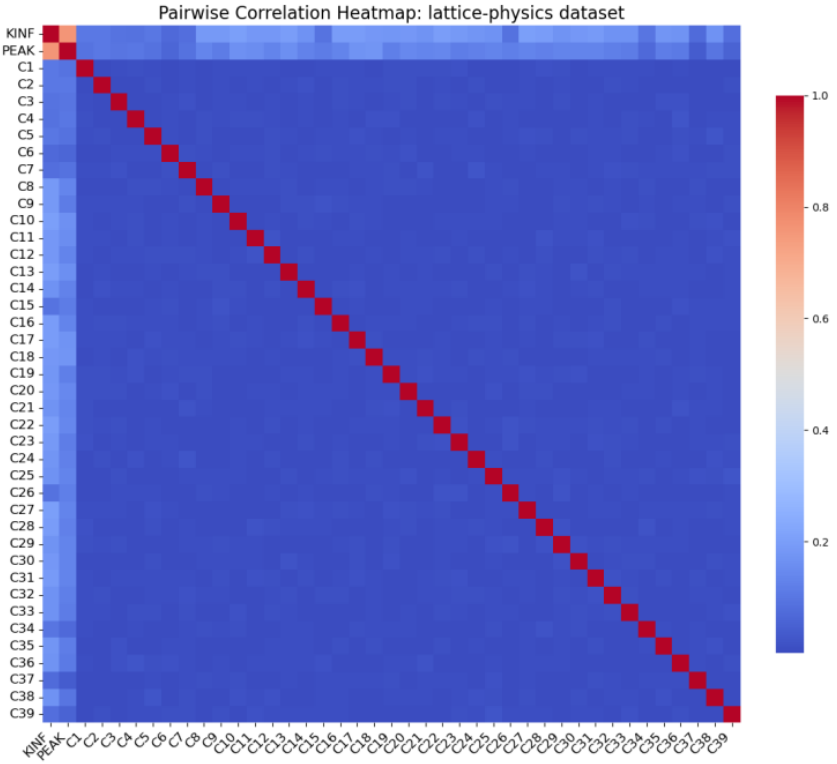

Figure 5. The pairwise correlation heatmap of all input parameters.

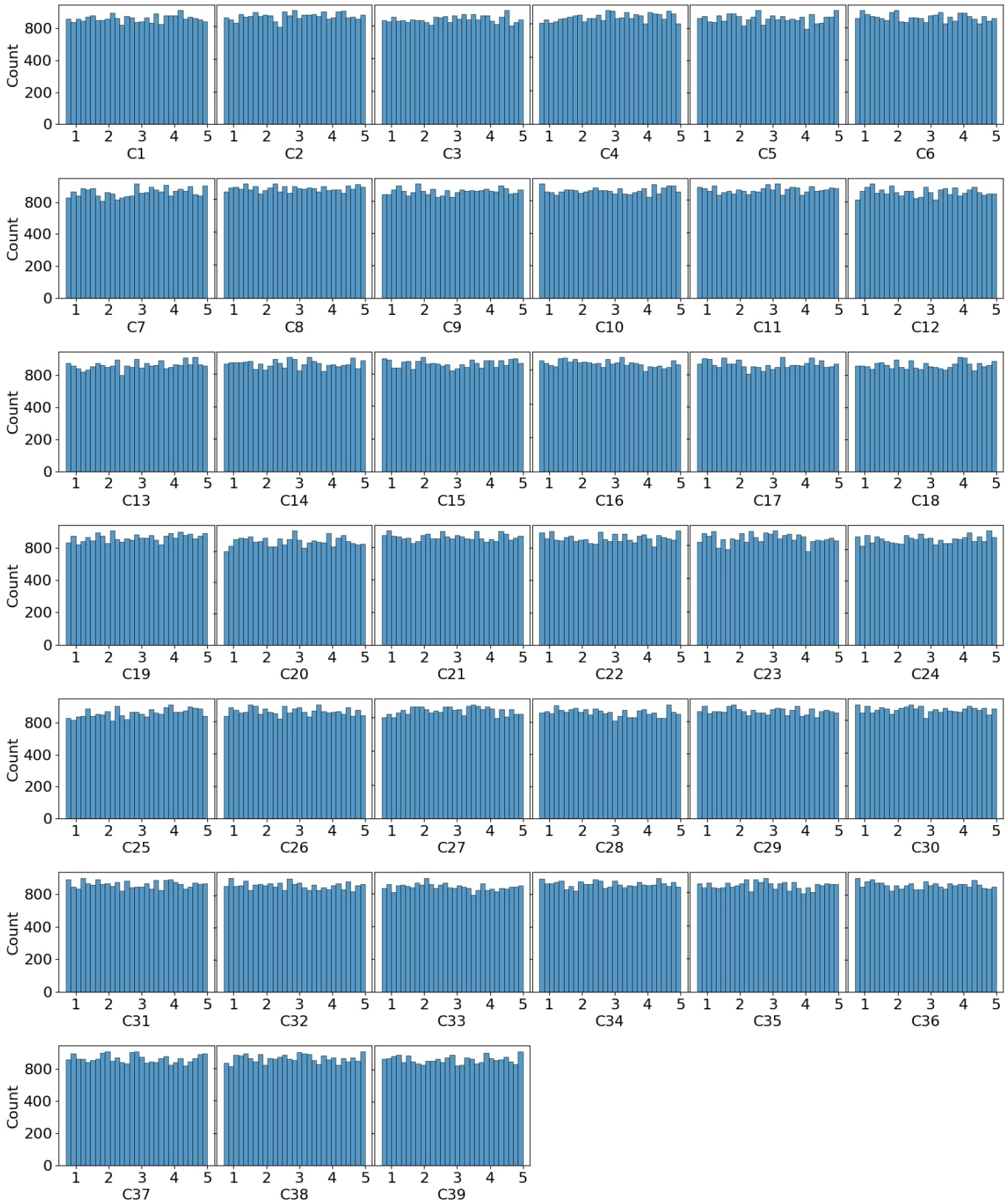

Figure 6. The histogram of all features.

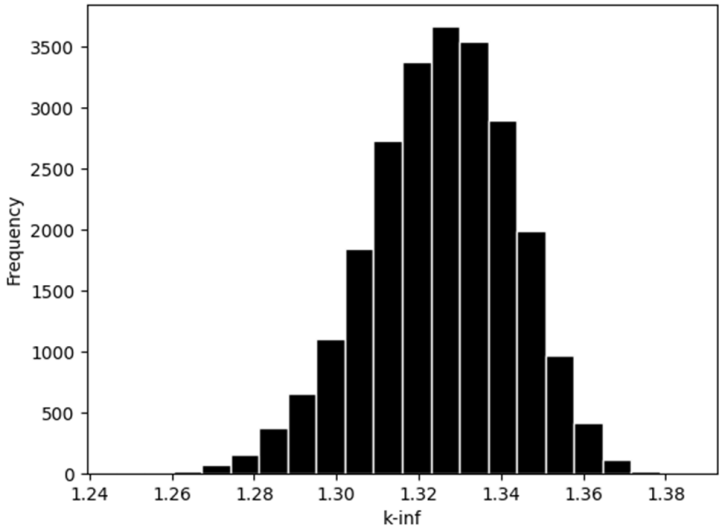

Figure 7. The histogram of k-inf values.

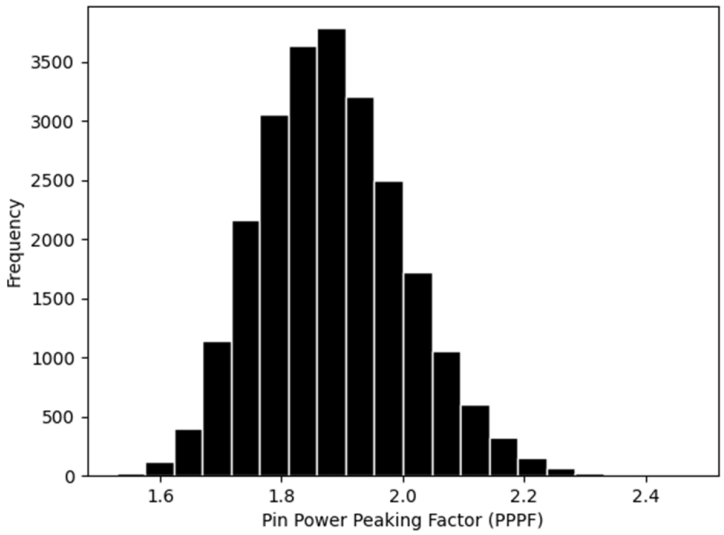

Figure 8. The histogram of PPPF values.

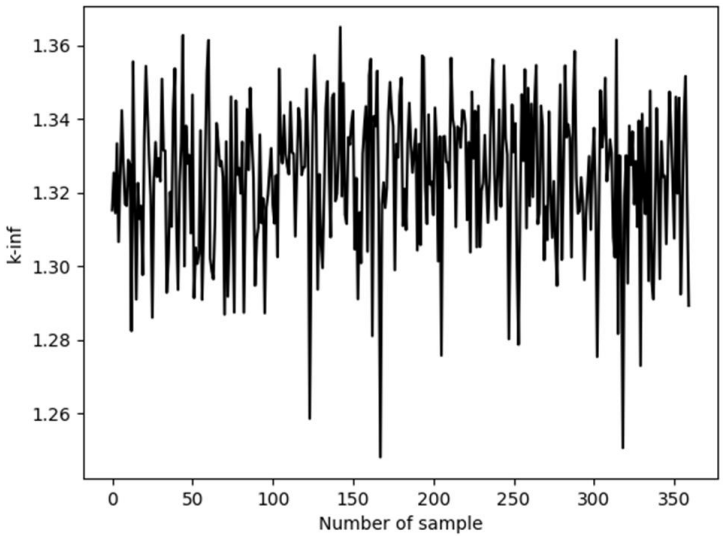

Figure 9. The k-inf values for the test dataset.

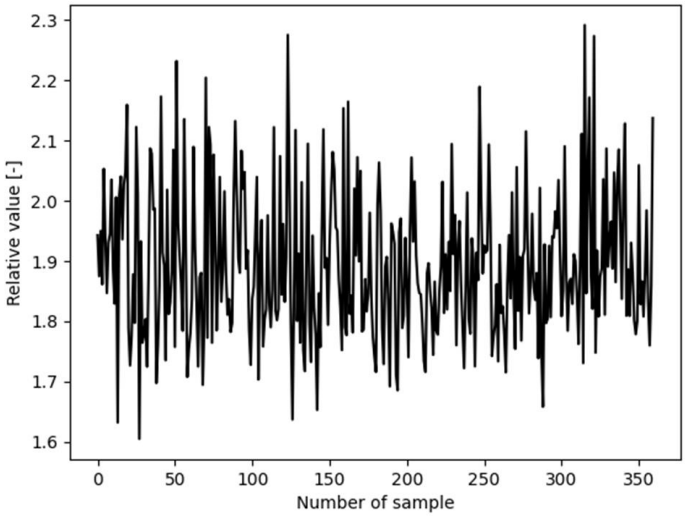

Figure 10. The PPPF values for the test dataset.

4. Results and Discussion

This Section presents the results in three main parts. The first part focuses on the confidence interval (CI) tests, evaluating model accuracy using metrics such as R^2 and MSE across all public datasets. The second part discusses statistical analysis. The third part discusses the performance of publicly available datasets, providing a benchmark for comparison. Finally, the fourth part highlights the results obtained from our proprietary 'in-house' dataset, which serves as the primary focus of this study.

4.1. Confidence Intervals

In the confidence interval analysis, we evaluated the nominal predictions of the DNN model across all datasets using a consistent configuration of initial hyperparameters for the public datasets. The configuration included 50 nodes per layer, 20 epochs, a learning rate of 0.01, and a batch size of 5. To compute the confidence intervals (CI), the experiment was repeated over 20 iterations, with the accuracy metric based on the R^2 method (see Equation (18)). The results, summarized in Table 2, indicate variations in MSE values across datasets, while R^2 values remained relatively consistent. Thus, R^2 was chosen as the primary metric to evaluate improvements in optimization results for the regression tasks in this study.

Table 2. The R^2 and MSE values and their 95% confidence intervals.

No.	Dataset	R^2_Mean	R^2_95% CI	MSE_Mean	MSE_95% CI
1	BH	0.820	0.811–0.828	13.17	12.63–13.82
2	CHF	0.988	0.986–0.990	27,056.38	22,859.4–32,010.1
3	CCS	0.869	0.859–0.878	33.83	31.52–36.33
4	CCPP	0.913	0.758–0.942	24.88	16.66–69.28
5	NOX	0.851	0.843–0.862	19.71	18.35–20.77

4.2. Statistical Analyses

In the statistical analysis, we employed the Wilcoxon Signed-Rank test [49] and paired *t*-test [50] across all datasets. Each optimization method was run 10 times per dataset. The null hypothesis (H_0) states that CASOH's performance is not significantly different from the baselines (Random Search, Bayesian Optimization, or Hyp-RL), while the alternative hypothesis (H_1) asserts that CASOH's performance is significantly different. As shown in Tables 3 and 4, both the paired *t*-test and Wilcoxon Signed-Rank test strongly reject the null hypothesis (H_0) with extremely low *p*-values for the comparison between CASOH and Random Search, indicating that CASOH's performance is significantly different and likely better. Similarly, for the comparison between CASOH and Bayesian Optimization, both

tests reject H_0, demonstrating a significant performance difference in favor of CASOH. In contrast, the comparison between CASOH and Hyp-RL shows no significant differences in either test, as the *p*-values exceed the 0.05 threshold, suggesting that CASOH and Hyp-RL perform comparably.

Table 3. The statistical test results.

No.	Dataset	Method	Run1	Run2	Run3	Run4	Run5	Run6	Run7	Run8	Run9	Run10	Average
1	BH	RS	0.8564	0.8641	0.8737	0.8596	0.8473	0.8463	0.8470	0.8530	0.8925	0.8433	0.8583
		BO	0.8622	0.8708	0.8524	0.8659	0.8598	0.8526	0.8569	0.8723	0.8622	0.8519	0.8607
		Hyp-RL	0.8660	0.8680	0.8740	0.8630	0.8750	0.8770	0.8700	0.8800	0.8810	0.8700	0.8724
		CASOH	0.8753	0.8940	0.8821	0.8863	0.8770	0.8657	0.8725	0.8777	0.8677	0.8774	0.8776
2	CHF	RS	0.9961	0.9964	0.9961	0.9960	0.9964	0.9974	0.9968	0.9971	0.9958	0.9965	0.9965
		BO	0.9968	0.9971	0.9970	0.9963	0.9971	0.9967	0.9968	0.9963	0.9953	0.9966	0.9966
		Hyp-RL	0.9970	0.9960	0.9960	0.9970	0.9960	0.9970	0.9970	0.9970	0.9970	0.9960	0.9966
		CASOH	0.9968	0.9967	0.9971	0.9970	0.9971	0.9970	0.9969	0.9969	0.9971	0.9969	0.9970
3	CCS	RS	0.8773	0.8673	0.8789	0.8817	0.8736	0.8940	0.8770	0.8924	0.8809	0.8853	0.8808
		BO	0.8907	0.8784	0.8801	0.8865	0.8798	0.8845	0.8874	0.8832	0.8747	0.8875	0.8833
		Hyp-RL	0.9020	0.9110	0.9080	0.9000	0.9030	0.9030	0.9140	0.9070	0.9100	0.9050	0.9063
		CASOH	0.8893	0.8884	0.8893	0.9050	0.9081	0.8930	0.8899	0.8974	0.8928	0.8890	0.8942
4	CCPP	RS	0.9397	0.9402	0.9385	0.9384	0.9405	0.9401	0.9405	0.9389	0.9395	0.9382	0.9395
		BO	0.9395	0.9416	0.9400	0.9406	0.9402	0.9391	0.9393	0.9405	0.9411	0.9411	0.9403
		Hyp-RL	0.9410	0.9420	0.9410	0.9400	0.9390	0.9380	0.9410	0.9420	0.9380	0.9380	0.9400
		CASOH	0.9415	0.9415	0.9404	0.9400	0.9411	0.9407	0.9407	0.9411	0.9404	0.9427	0.9410
5	NOX	RS	0.9131	0.9023	0.9059	0.9043	0.9057	0.9102	0.9001	0.9068	0.9058	0.9122	0.9066
		BO	0.9065	0.9033	0.9055	0.9075	0.9136	0.9090	0.9135	0.9094	0.9033	0.9102	0.9082
		Hyp-RL	0.9090	0.9100	0.9120	0.9110	0.9130	0.9120	0.9090	0.9120	0.9100	0.9110	0.9109
		CASOH	0.9131	0.9148	0.9192	0.9137	0.9140	0.9113	0.9156	0.9120	0.9117	0.9108	0.9136

Table 4. The hypothesis test results.

Test	Comparison	Statistic	*p*-Value	Conclusion
Paired *t*-test	RS vs. CASOH	$t = -4.775$	1.67×10^{-5}	Reject H_0
Paired *t*-test	BO vs. CASOH	$t = -5.055$	6.40×10^{-6}	Reject H_0
Paired *t*-test	Hyp-RL vs. CASOH	$t = 1.268$	2.11×10^{-1}	Fail to Reject H_0
Wilcoxon	RS vs. CASOH	W = 87.0	1.07×10^{-7}	Reject H_0
Wilcoxon	BO vs. CASOH	W = 116.0	4.80×10^{-7}	Reject H_0
Wilcoxon	Hyp-RL vs. CASOH	W = 616.0	8.36×10^{-1}	Fail to Reject H_0

4.3. Results for Public Datasets

In this investigation, the hyperparameters selected for optimization are the learning rate, number of epochs, batch size, and number of nodes in the hidden layers of the DNN model. The selected ranges of hyperparameters in Table 5 are carefully evaluated based on the sensitivity tests presented in the Supplementary Materials File S2 for all public datasets.

To observe the accuracy of prediction, the R^2 value, or the coefficient of determination, which is a statistical measure to indicate how well the predicted values from a regression model align with the actual values observed in the dataset, is implemented. The R^2 values range from 0 to 1. An R^2 value of 0 indicates that the model explains none of the variability in the target variable, while an R^2 value of 1 indicates perfect prediction. The Equation (18) of the R^2 value is as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2} \tag{18}$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, $\overline{y}$ is the mean value, and n is the number of observations.

Table 5. The selected ranges of hyperparameters for all public datasets (File S2).

No.	Quantity	BH	CHF	CCS	CCPP	NOX
1	Learning rate	1×10^{-5} to 1×10^{-2}	5×10^{-3} to 2×10^{-2}	1×10^{-2} to 4×10^{-2}	2×10^{-3} to 2×10^{-2}	5×10^{-3} to 2×10^{-2}
2	Number of hidden layers	2	2	2	2	2
3	Number of nodes per layer	25 to 200	100 to 200	100 to 200	25 to 200	25 to 200
4	Epochs	50 to 200	50 to 200	25 to 200	50 to 200	100 to 200
5	Activator	ReLU	ReLU	ReLU	ReLU	ReLU
6	Optimizer	Adam	Adam	Adam	Adam	Adam
7	Batch size	1 to 50	1 to 50	1 to 25	100 to 150	25 to 50

The total number of iterations is set to 25 for all optimization methods. The optimization results are summarized in Table 6. In this Table, NO denotes the nominal prediction of the DNN model (i.e., no optimization applied), RS represents random search, and BO stands for Bayesian optimization. The variable L represents the learning rate, while N1 and N2 indicate the number of neurons in the first and second hidden layers, respectively. Certain hyperparameters, such as the number of hidden layers (set to two), the Adam optimizer, and the ReLU activation function, remained fixed during optimization. Improvements were calculated using the following formula:

$$I\ (\%) = 100 * \frac{R_X^2 - R_N^2}{R_N^2} \tag{19}$$

where R_N^2 is the R^2 value of the nominal prediction, R_X^2 is referred to as the prediction of methods X, such as random search, Bayesian Optimization, Hyp-RL, and CASOH.

Table 6. The optimized parameters and improvements.

No.	Datasets	Methods	L	N1	N2	Epochs	Batch Size	R^2	I (%) *
1	BH	NO	1.00×10^{-2}	50	50	20	5	0.802	0
		RS	1.46×10^{-4}	168	168	110	10	0.864	7.7
		BO	2.16×10^{-4}	152	25	99	10	0.858	7.0
		Hyp-RL	9.64×10^{-2}	46	46	176	38	0.866	8.0
		CASOH	1.45×10^{-2}	103	159	112	11	0.894	11.5
2	CHF	NO	1.00×10^{-2}	50	50	20	5	0.983	0
		RS	1.81×10^{-2}	179	157	152	6	0.992	0.9
		BO	1.22×10^{-2}	165	128	132	8	0.994	1.1
		Hyp-RL	1.10×10^{-2}	159	171	143	19	0.997	1.4
		CASOH	1.14×10^{-2}	135	119	184	22	0.997	1.4
3	CCS	NO	1.00×10^{-2}	50	50	20	5	0.835	0
		RS	3.06×10^{-2}	173	166	156	20	0.898	7.5
		BO	2.15×10^{-2}	159	119	184	16	0.886	6.1
		Hyp-RL	2.10×10^{-2}	121	192	111	13	0.905	8.4
		CASOH	3.91×10^{-2}	125	191	177	22	0.905	8.4
4	CCPP	NO	1.00×10^{-2}	50	50	20	5	0.844	0
		RS	1.12×10^{-2}	108	159	119	118	0.918	8.8
		BO	1.46×10^{-2}	47	57	125	107	0.932	10.4
		Hyp-RL	8.50×10^{-3}	82	120	122	105	0.941	11.5
		CASOH	5.05×10^{-3}	96	67	115	133	0.941	11.5
5	NOX	NO	1.00×10^{-2}	50	50	20	5	0.819	0
		RS	5.60×10^{-3}	170	148	122	32	0.913	11.5
		BO	5.93×10^{-3}	155	192	148	40	0.909	11.0
		Hyp-RL	9.41×10^{-3}	151	79	122	37	0.909	11.0
		CASOH	5.62×10^{-3}	165	159	164	30	0.914	11.6

* Refers to the enhanced value of R^2, which is illustrated in Equation (18).

For the Boston Housing dataset, CASOH achieves a marginally higher R^2 value of 0.894, outperforming Bayesian Search (0.858) and Random Search (0.864) (see Figure 11). Similarly, for the CHF dataset, CASOH demonstrates comparable performance, with R^2 values of 0.997, 0.994, and 0.992 for CASOH, Bayesian Search, and Random Search, respectively (see Figure 12). In contrast, CASOH and Hyp-RL yield similar results.

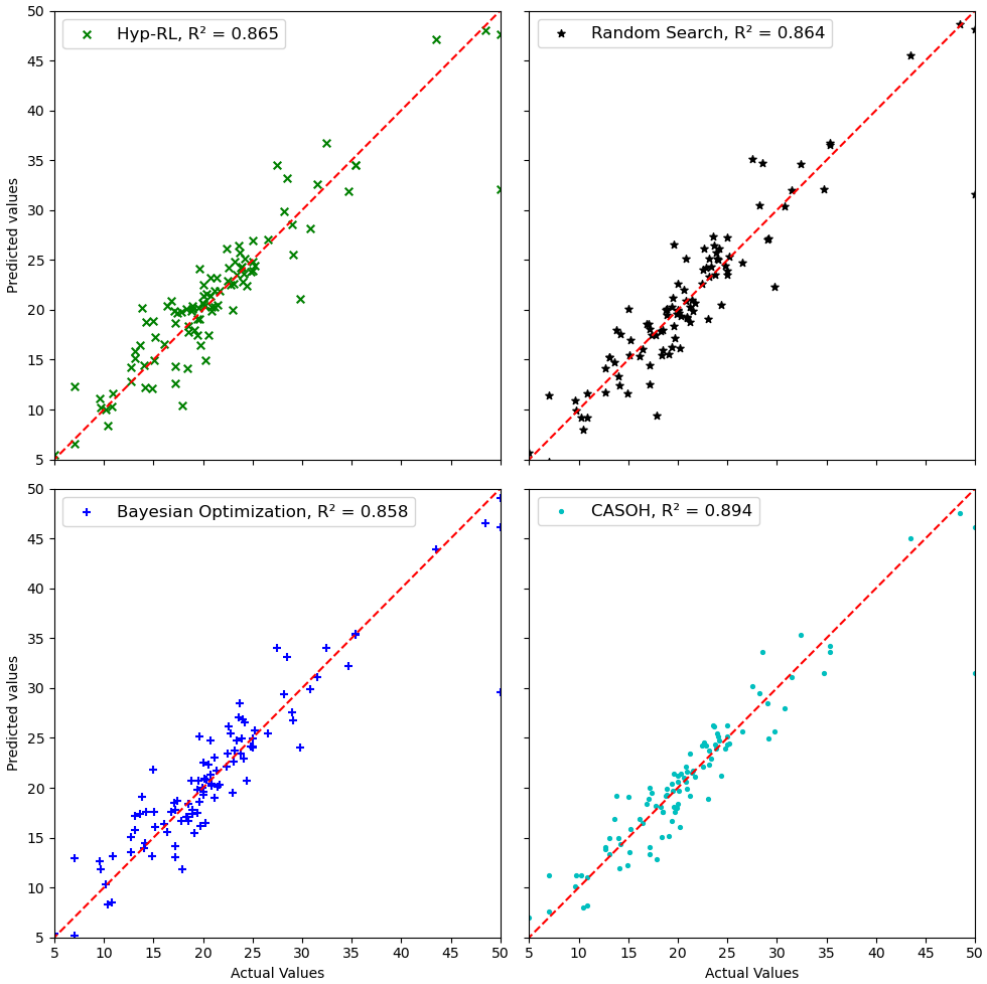

Figure 11. The comparison results for the Boston Housing dataset. This test dataset was extracted from the training dataset with a ratio of 0.2.

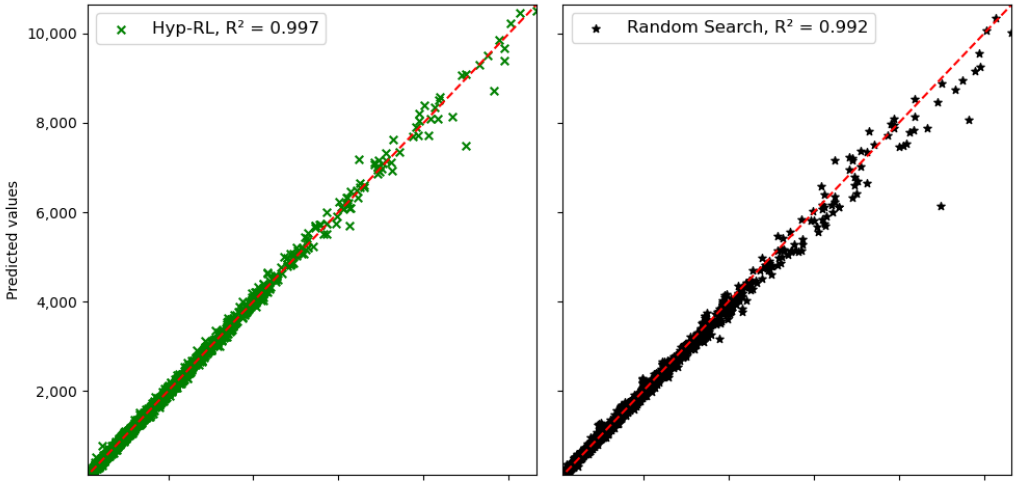

Figure 12. *Cont.*

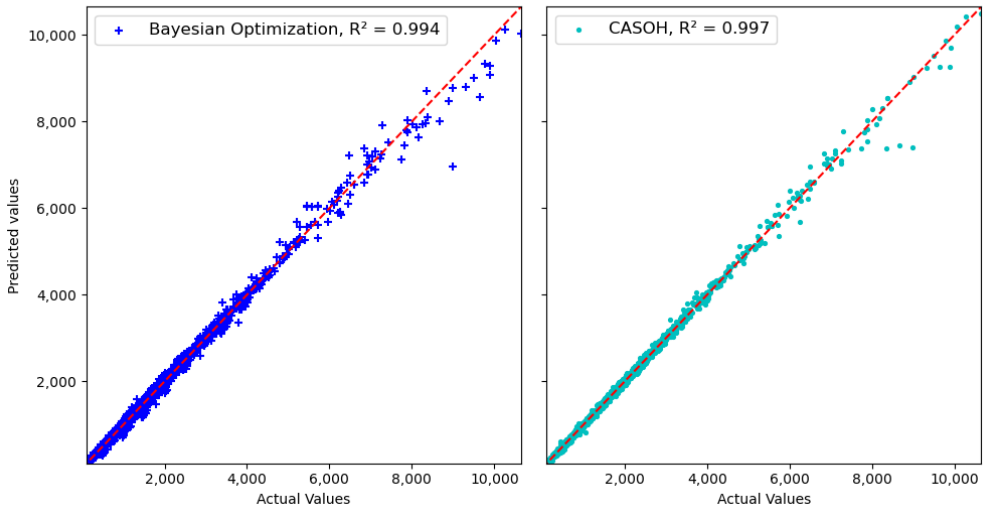

Figure 12. CHF dataset comparison. This test dataset was extracted from the training dataset with a ratio of 0.2.

For the CCS dataset, the R^2 values for CASOH, Bayesian Search, and Random Search are 0.908, 0.886, and 0.898, respectively, indicating generally consistent performance across these methods (see Figure 13). In the case of the CCPP dataset, CASOH achieves an R^2 value of 0.941, slightly surpassing Bayesian Search (0.932) and Random Search (0.918) (see Figure 14). Lastly, for the NOX dataset, the R^2 values are closely aligned, with CASOH at 0.914, Bayesian Search at 0.909, and Random Search at 0.913 (see Figure 15). Notably, CASOH and Hyp-RL exhibit similar performance on this dataset as well.

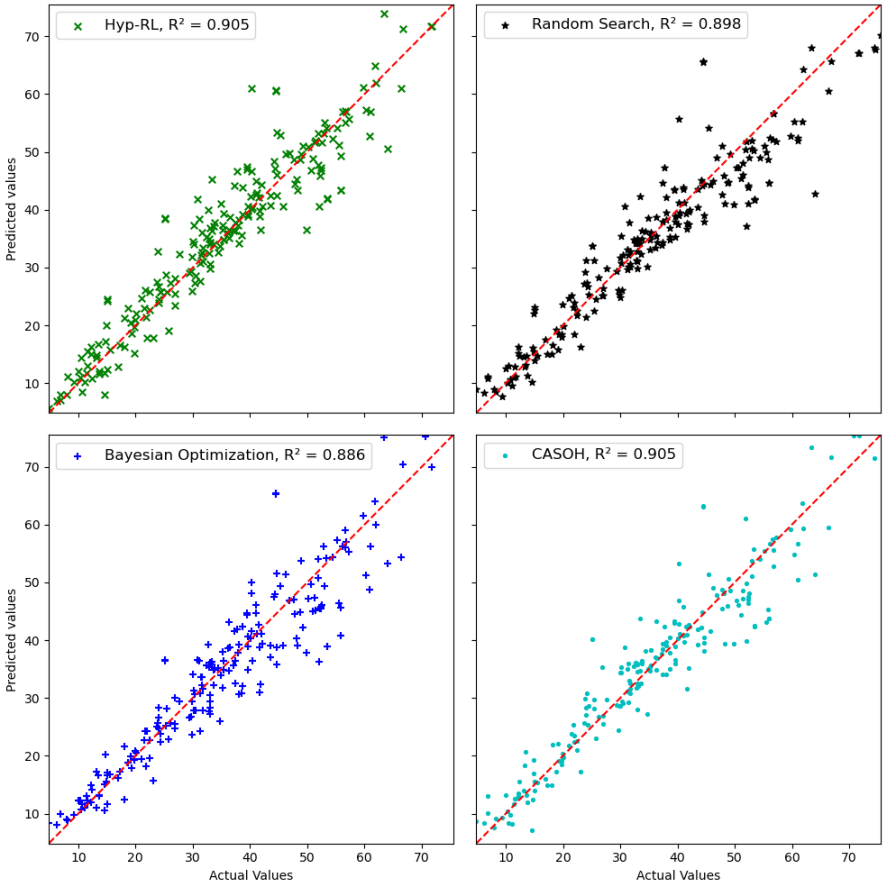

Figure 13. CCS dataset comparison. This test dataset was extracted from the training dataset with a ratio of 0.2.

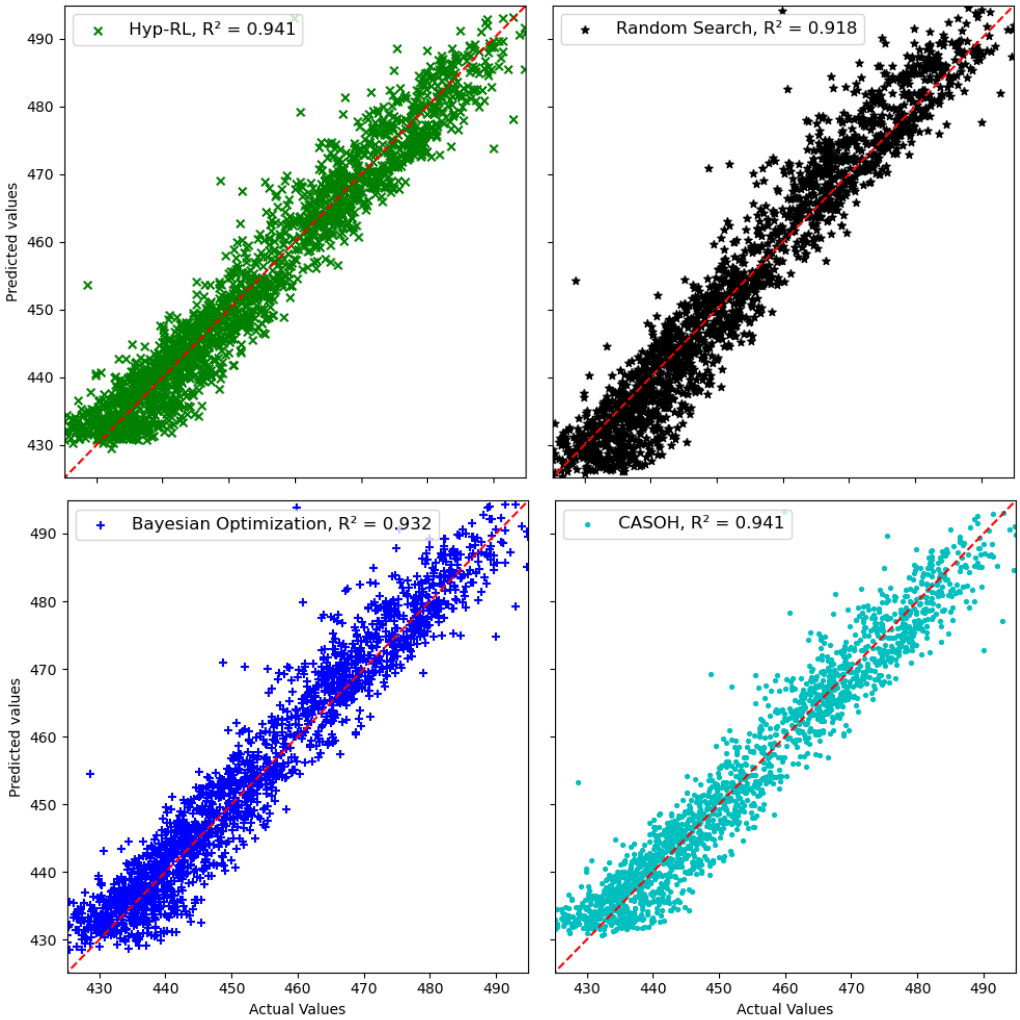

Figure 14. CCPP dataset comparison. This test dataset was extracted from the training dataset with a ratio of 0.2.

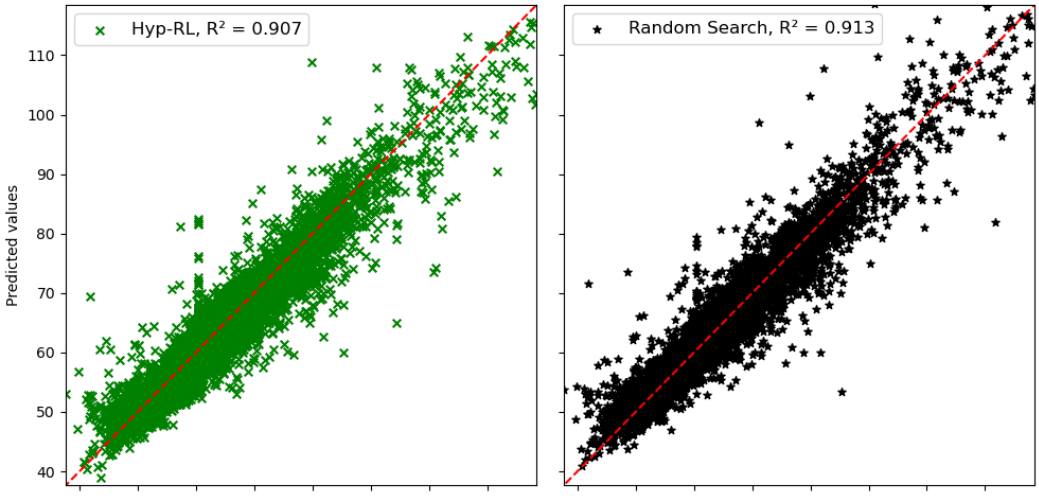

Figure 15. *Cont*.

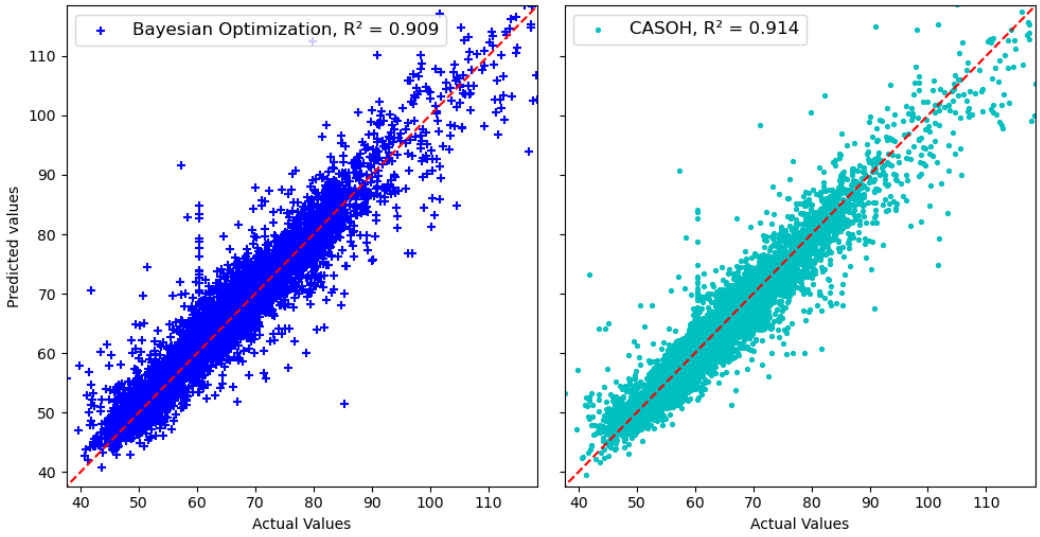

Figure 15. NOX dataset comparison. This test dataset was extracted from the training dataset with a ratio of 0.2.

Overall, CASOH demonstrates improvements of up to approximately 12% across the datasets, with more modest gains observed in the CHF dataset, where the Hyp-RL method achieves a similar level of improvement. These results suggest that CASOH offers incremental yet meaningful performance enhancements across multiple datasets and can be effectively compared with the state-of-the-art Hyp-RL method, as highlighted in the statistical analysis. However, further investigation is required to establish its consistent effectiveness, as the performance of CASOH and Hyp-RL remains comparable in certain cases.

4.4. Lattice-Physics Dataset Analysis Results

For this dataset, we conducted a thorough examination of the valid hyperparameter range and adjusted categorical variables, including the optimizer and activation functions. As the results of the sensitivity analysis (see Supplemetary File S2), the valid ranges/values of non-categorical hyperparameters and categorical hyperparameters are illustrated in Table 7. For the categorical hyperparameters, we selected ReLU [51], ELU [52], LeakyReLU, Softplus, and linear activators. In particular, the LeakyReLU is a type of activation function based on a ReLU, but it has a small slope for negative values instead of a flat slope; Softplus is a smooth approximation to the ReLU function and can be used to constrain the output of a model to always be positive. In addition, the selected optimizers were Adam [53] and the optimizers in Keras [54], such as SGD, RMSprop, and Adagrad (see Table 7). It is important to note that the number of hidden layers is an adjustable hyperparameter in the CASOH framework. However, to ensure a consistent basis for comparisons in this investigation, we employed a fixed two-hidden layer configuration of DNN architecture.

In this investigation, we compared the results of CASOH with nominal hyperparameters configuration, Random search [18], Optuna Bayesian optimization method [21], and Hyp-RL method [37]. It can be illustrated that the nominal candidate refers to a random sample, as indicated in Equation (5). All these methods employ the same valid hyperparameter ranges in Table 7, as well as the DNN architecture, differing only in their sampling approaches.

In the CASOH framework, the step size δ_L and δ_N are 0.0001 and 1 for the learning rate and the number of nodes, respectively. The detailed best candidates identified by each optimization method are summarized in Table 8, with all methods evaluated over 300 iterations. As illustrated in Figure 16, CASOH consistently identifies superior candidates with significantly fewer iterations, highlighting the efficiency of its combined sampling algorithm. While the results of Hyp-RL are comparable to those of Random

Search, Bayesian optimization achieves a lower MSE compared to Random Search. Furthermore, the comparison of the k-inf and PPPF predictions using the different methods is presented in Figures 17 and 18.

Table 7. The valid ranges/values of non-categorical hyperparameters and the valid choices for categorical hyperparameters.

No.	Non-Categorical Hyperparameters	Original Test Range	Valid Range/Value
1	Learning rate	0.0001 to 0.1	0.0005 to 0.01
2	Number of nodes per layer	0 to 200	30 to 200
3	Validation split	0.05 to 0.85	0.1
4	Epochs	1 to 200	200
	Categorical Hyperparameters	Valid Choice	
5	Activator	$\{S_a\} = \{ReLu,\ ELu,\ LeakyReLu,\ softplus,\ linear\}$	
6	Optimizer	$\{S_o\} = \{Adam,\ SGD,\ RMSprop,\ Adagrad\}$	

Table 8. The best candidates for the optimization process for the lattice-physics dataset.

Quantity	Nominal	Random	Bayesian	Hyp-LR	CASOH
Total number of trials	-	300	300	300	300
Learning rate	0.02	9.58×10^{-3}	5.64×10^{-3}	4.59×10^{-3}	7.03×10^{-4}
Number of nodes for 1st layer	40	64	55	137	46
Number of nodes for 2nd layer	50	187	131	141	177
Activator for 1st layer	ReLU	ReLU	ReLU	ReLU	ReLU
Activator for 2nd layer	LeakyReLU	LeakyReLU	LeakyReLU	ReLU	LeakyReLU
Optimizer	RMSprop	Adam	Adam	Adam	Adam
R^2 mean	0.633	0.896	0.934	0.946	0.987

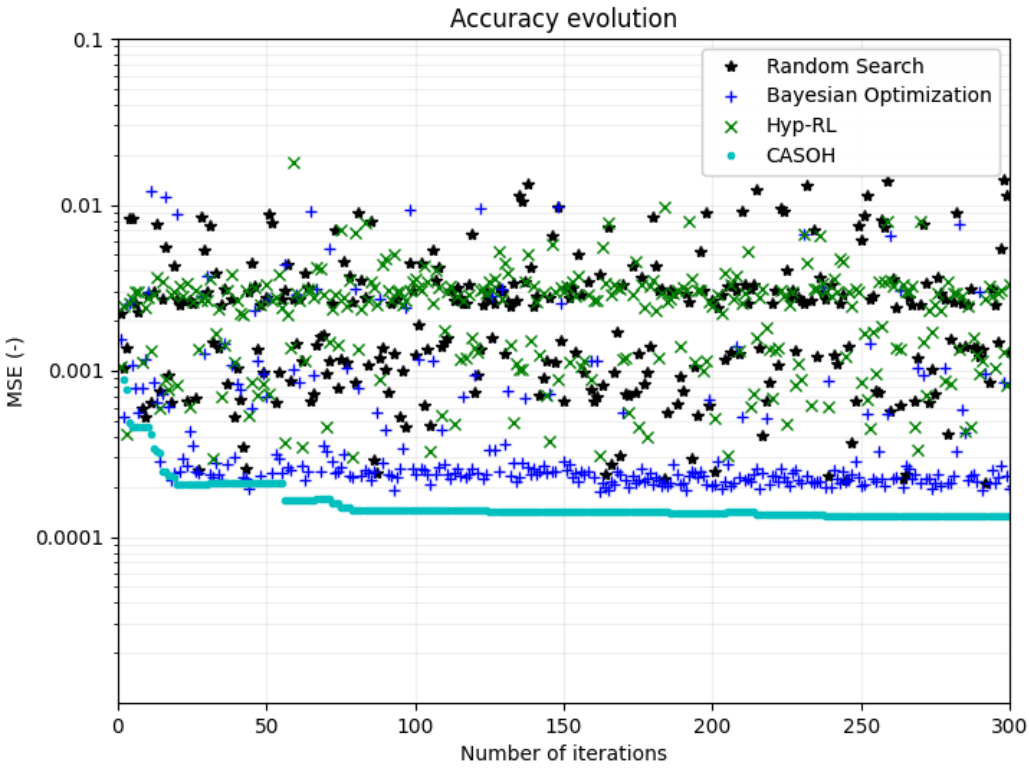

Figure 16. Accuracy evolution comparison.

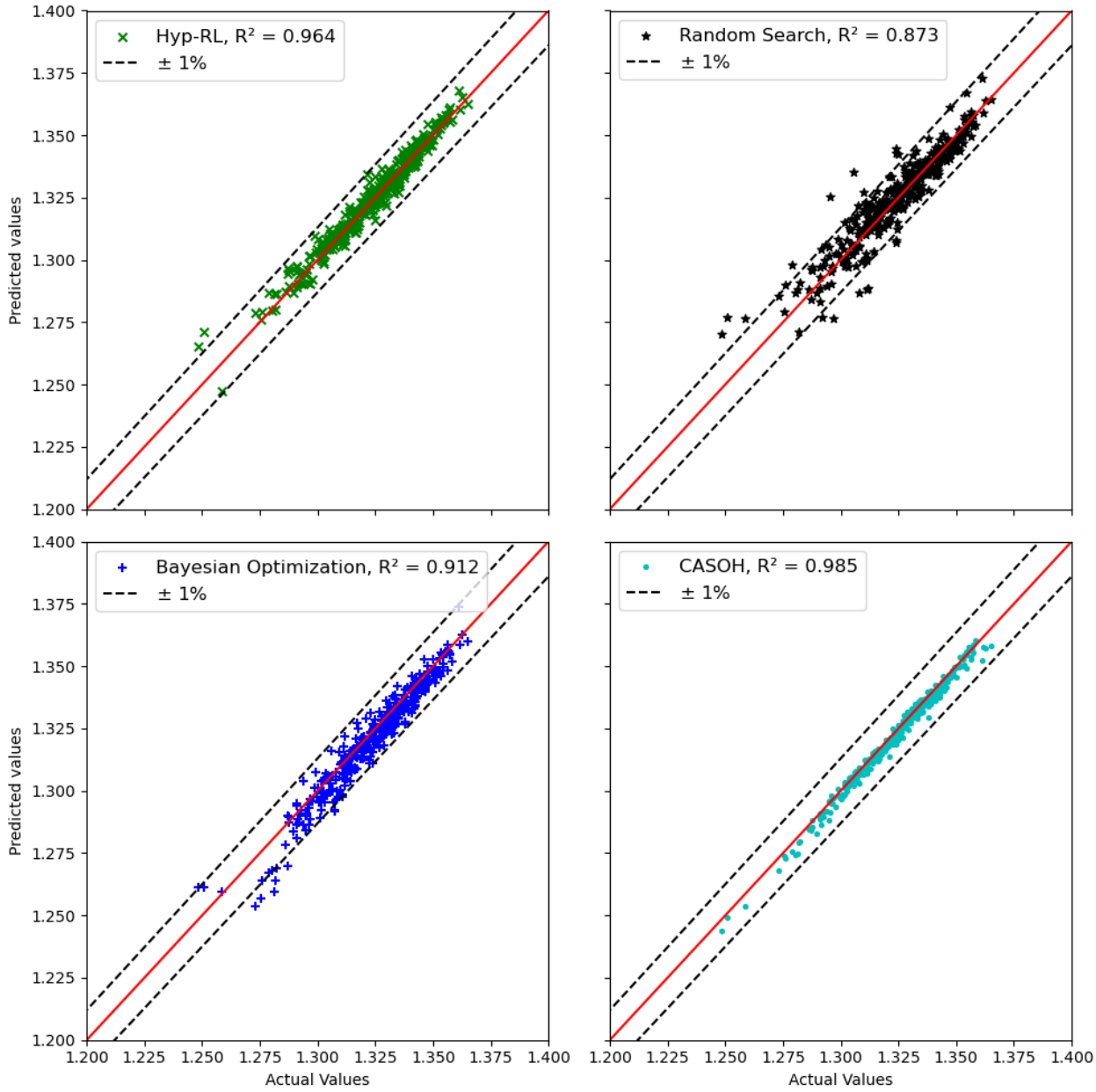

Figure 17. Comparison of the best predictions for k-inf.

The improvements achieved by the CASOH optimization method on the lattice-physics dataset are notably more pronounced compared to those observed with the public datasets. This disparity may be attributed to factors such as the increased number of iterations and the complexity of handling categorical variables. Both CASOH and Hyp-RL demonstrated superior performance compared to Bayesian optimization and the random search method. For example, in the prediction of k-inf, CASOH improved the R^2 value by 56.6% (from 0.629 to 0.985), compared to improvements of 53.2% by Hyp-RL, 44.9% by Bayesian optimization, and 38.8% by random search relative to the nominal prediction. Similarly, in the prediction of PPPF, CASOH achieved a 55.4% increase in R^2 (from 0.637 to 0.990), surpassing the improvements of Hyp-RL (45.5%), Bayesian optimization (50.1%), and random search (44.1%). These findings highlight the potential effectiveness of CASOH in hyperparameter optimization, particularly for complex problems and high-dimensional parameter spaces such as those encountered in lattice-physics datasets. Nevertheless, the observed improvements may also be influenced by dataset-specific characteristics and parameter configurations, underscoring the need for further studies to validate and generalize these results.

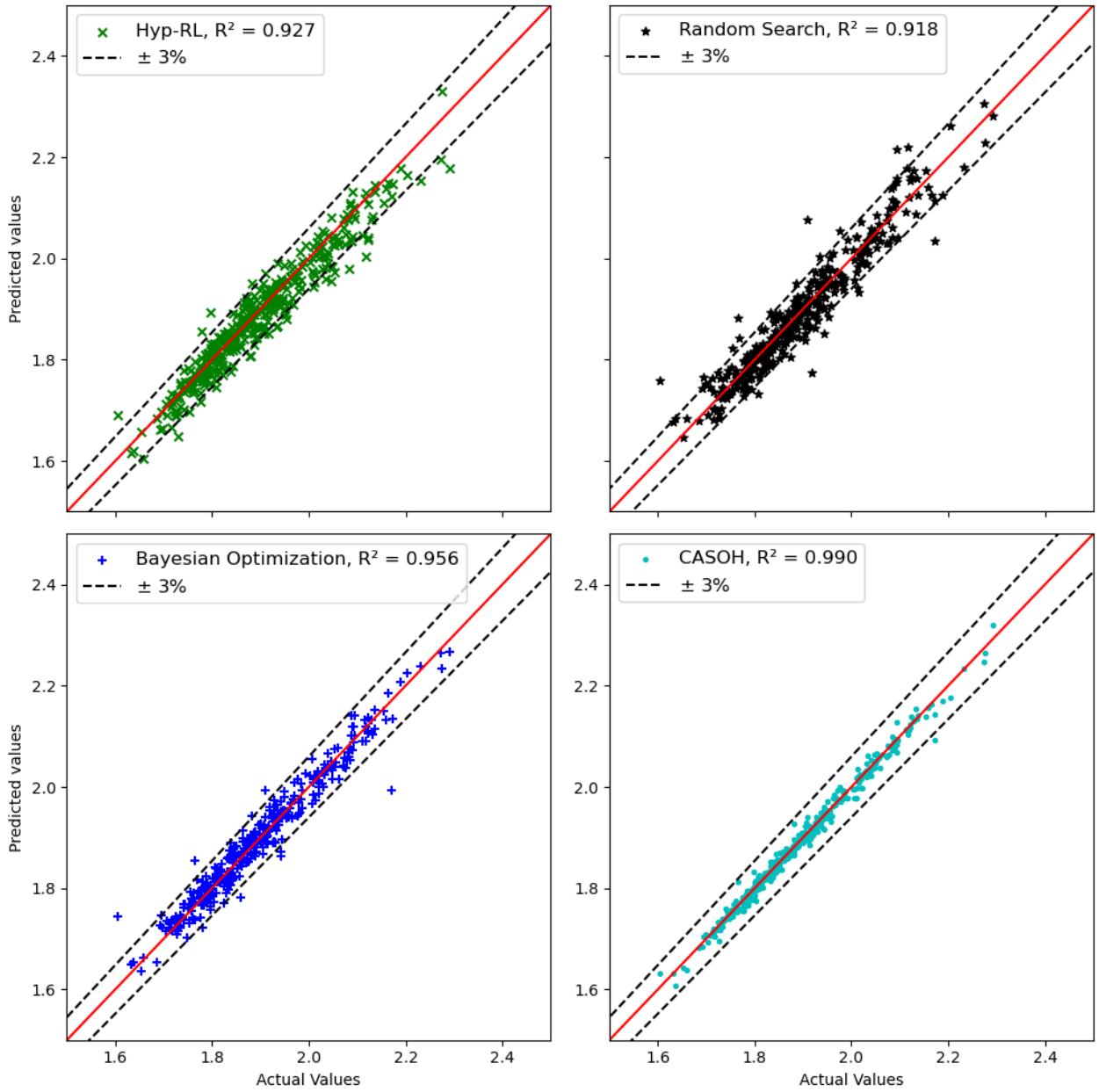

Figure 18. Comparison of the best predictions for PPPF.

5. Conclusions

We developed a novel combined-sampling technique that integrates the Metropolis-Hastings algorithm with uniform random sampling to create a hyperparameter optimization framework named CASOH. This framework was tested on six datasets: Boston Housing, CHF in water-cooled tubes, CCS, CCPP, NOX, and an in-house dataset for predicting nuclear fuel assembly parameters. The results indicate that CASOH demonstrates relatively rapid convergence and significant performance improvements, particularly when applied to high-complexity datasets. The most significant enhancement was observed in the lattice-physics dataset, achieving a 56.6% improvement in prediction accuracy, compared to improvements of 53.2% by Hyp-RL, 44.9% by Bayesian optimization, and 38.8% by random search relative to the nominal prediction.

Despite its advantages, CASOH incurs higher computational costs than traditional methods, which can be mitigated through parallel computations. The adaptability of CASOH to various parameter types and its robust combined sampling algorithm suggest its potential as a promising approach for optimizing hyperparameters in high-dimensional,

non-linear, and multi-objective systems. Nevertheless, several challenges persist, including scalability to deeper neural networks, sensitivity to noisy and missing data, methods for determining the interdependency of parameters, and the need for more comprehensive comparisons with state-of-the-art optimization techniques. Addressing these challenges and exploring these advanced methodologies will be considered in future work.

Supplementary Materials: The following supporting information can be downloaded at https://www.mdpi.com/article/10.3390/math12243892/s1, File S1: The CASOH sampling explanations; File S2: Sensitivity analysis.

Author Contributions: Conceptualization, N.H.T., K.-D.K. and H.-Y.J.; methodology, N.H.T., N.N.A. and H.-Y.J.; software, N.H.T., K.-D.K. and H.-Y.J.; validation, N.H.T., N.N.A. and V.-K.H.; formal analysis, N.H.T. and N.N.A.; investigation, N.H.T. and N.N.A.; resources, N.H.T., N.X.M. and M.T.V.; data curation, N.H.T., N.X.M., N.N.A. and V.-K.H.; writing—original draft preparation, N.H.T.; writing—review and editing, N.H.T., K.-D.K., N.X.M., H.-Y.J., N.N.A., V.-K.H., H.-N.T., N.-N.D. and M.T.V.; visualization, N.H.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The "Boston Housing" dataset [32] is publicly available in the Kaggle system (https://www.kaggle.com/code/prasadperera/the-boston-housing-dataset, accessed on 30 October 2024). The "CHF" dataset [33] is publicly available in the USNRC database and on the Kaggle system (https://www.kaggle.com/datasets/kaustubhdhote/critical-heat-flux-dataset, accessed on 30 October 2024). The "Concrete compressive strength" dataset [34] is publicly available in the UCI machine learning repository (https://archive.ics.uci.edu/dataset/165/concrete+compressive+strength, accessed on 30 October 2024). The "Combined Cycle Power Plant" dataset [35] is publicly available in the UCI machine learning repository (https://archive.ics.uci.edu/dataset/294/combined+cycle+power+plant, accessed on 30 October 2024). The "Gas Turbine CO and NOx Emission" dataset [36] is publicly available in the UCI machine learning repository (https://archive.ics.uci.edu/dataset/551/gas+turbine+co+and+nox+emission+data+set, accessed on 30 October 2024). The "in-house" dataset used to support the findings of this study is available from the corresponding author upon a reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

Nomenclature

Acronyms:

DNN	Deep Neural Network
CASOH	Combined-sampling Algorithm to Search the Optimized Hyperparameters
PPPF	Pin power peaking factor
MCNP6	Monte Carlo N-Particle Neutron Transport Code version 6
SRAC2006	A comprehensive neutronic calculation code system developed by Japan Atomic Energy Institute
SCALE	Standard Computational Analysis for Licensing Evaluation
BOC	Beginning of Cycle
BPN	Back propagation network
NFAC-01	NuScale fuel assembly type C-01
MSE	Mean Square Error
Relu	Rectified Linear Unit
Elu	Exponential Linear Unit
LeakyRelu	Leaky Rectified Linear Unit
Softplus	Softplus Activation Function
Linear	Linear Activation Function
Adam	Adaptive Moment Estimation
SGD	Stochastic Gradient Descent Optimizer
RMSprop	Root Mean Square Propagation Optimizer
Adagrad	Adaptive Gradient Algorithm Optimizer
AI	Artificial Intelligence

ML	Machine learning
SVM	Support vector machine
CNN	Convolutional neural network
RF	Random Forest
DEEP-BO	Deep neural network Bayesian Optimization
TPE	Tree-structured Parzen Estimator
GEM-ITH	Generalized Weighted Ensemble with Internally Tuned Hyperparameters
OpenMC	open-source Monte Carlo particle transport simulation code
CHF	Critical heat flux
NO	Nominal–NO optimization
RS	Random search
BO	Bayesian optimization
BH	Boston Housing
CCS	Concrete compressive strength
CCPP	Combined Cycle Power Plant
NOX	Gas Turbine CO and NOx Emission
CI	Confidence interval
Hyp-RL	Hyperparameter reinforcement learning
Symbols:	
$f(X)$	The continuous function depends on X under set A
F	The optimized value of function $f(X)$
x	Uniform sample
y	Continuous sample
z	Categorical sample
a	Minimum value of the parameter's range
b	Maximum value of the parameter's range
δ	Step size of the continuous sample
$\{S\}$	Set of predefined character combinations
$L_1,\ L_2$	Minimum and maximum value of learning rate
$N_1,\ N_2$	Minimum and maximum value of the number of nodes per layer
$F_a(X_a)$	Cost function of the accepted sample
$F_i(X_i)$	Cost function of the uniform and continuous samples
$\alpha_i = \frac{F_i}{F_a}$	The ratio of cost functions
$F_{min}(X_{min})$	The minimum cost function of the sample
α_{min}	The minimum value of $\alpha_i (\alpha_{min} = \min\{1, \alpha_i\})$
β	User-defined value to control the convergence
u	Random number from β to 1
$C_1, C_2, \ldots, C_n$	Fuel pin enrichments
n	Number of fuel rods
$f_1, f_2, \ldots, f_n$	Pin-by-pin power factors
R^2	Accuracy of the prediction
y_i	Actual value
$\hat{y}_i$	Predicted value
$\overline{y}$	Mean value
$\pi(x)$	Target distribution
$q(x^t \mid x)$	Proposal distribution with variables x and x^t
μ	Concentration control variable
$X_{optimal}$	Optimized value
$f_0(x)$	Objective function
$f_i(x)$	Constraint function

References

1. Ayodeji, A.; Amidu, M.A.; Olatubosun, S.A.; Addad, Y.; Ahmed, H. Deep learning for safety assessment of nuclear power reactors: Reliability, explainability, and research opportunities. *Prog. Nucl. Energy* **2022**, *151*, 104339. [CrossRef]
2. Sun, L.; Gao, H.; Pan, S.; Wang, J.X. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Comput. Methods Appl. Mech. Eng.* **2020**, *361*, 112732. [CrossRef]

3. Racheal, S.; Liu, Y.; Ayodeji, A. Evaluation of optimized machine learning models for nuclear reactor accident prediction. *Prog. Nucl. Energy* **2022**, *149*, 104263. [CrossRef]
4. Gurgen, A.; Dinh, N.T. Development and assessment of a reactor system prognosis model with physics-guided machine learning. *Nucl. Eng. Des.* **2022**, *398*, 111976. [CrossRef]
5. Jin, Y.; Bajorek, S.M.; Cheung, F.B. Validation and Uncertainty Quantification of Transient Reflood Models Using COBRA-TF and Machine Learning Techniques Based on the NRC/PSU RBHT Benchmark. *Nucl. Sci. Eng.* **2023**, *197*, 967–986. [CrossRef]
6. Tiep, N.H.; Kim, K.D.; Jeong, H.Y.; Xuan-Mung, N.; Hoang, V.K.; Ngoc Anh, N.; Vu, M.T. Machine learning applications and uncertainty quantification analysis for reflood tests. *Appl. Sci.* **2024**, *14*, 324. [CrossRef]
7. Jo, Y.; Shin, H.C. Design optimization of cylindrical burnable absorber inserted into annular fuel pellets for soluble-boron-free SMR. *Nucl. Eng. Technol.* **2022**, *54*, 1464–1470. [CrossRef]
8. Gu, X.; Radaideh, M.I.; Liang, J. OpenNeoMC: A framework for design optimization in particle transport simulations based on OpenMC and NEORL. *Ann. Nucl. Energy* **2023**, *180*, 109450. [CrossRef]
9. Tiep, N.H.; Kim, K.-D.; Jeong, H.-Y.; Anh, N.N.; Mung, N.X. Application of data assimilation in searching better lattice-physics parameters of fuel assembly. *Nucl. Eng. Des.* **2023**, *411*, 112415. [CrossRef]
10. Akbari, R.; Ochbelagh, D.R.; Gharib, A.; Maiorino, J.R.; D'Auria, F. Small modular reactor full scope core optimization using Cuckoo Optimization Algorithm. *Prog. Nucl. Energy* **2020**, *122*, 103271. [CrossRef]
11. Goorley, T.; James, M.; Booth, T.; Brown, F.; Bull, J.; Cox, L.J.; Durkee, J.; Elson, J.; Fensin, M.; Forster, R.A.; et al. Initial MCNP6 release overview. *Nucl. Technol.* **2012**, *180*, 298–315. [CrossRef]
12. Keisuke, O.; Kugo, T.; Kaneko, K.; Tsuchihashi, K. SRAC2006: A Comprehensive Neutronics Calculation Code System. 2007. Available online: https://www.osti.gov/etdeweb/biblio/20952076 (accessed on 30 October 2024).
13. Rearden Bradley, T.; Jessee, M.A. *SCALE Code System*; No. ORNL/TM-2005/39 Version 6.2. 3.; Oak Ridge National Lab. (ORNL): Oak Ridge, TN, USA, 2018.
14. Romano Paul, K.; Forget, B. The OpenMC monte carlo particle transport code. *Ann. Nucl. Energy* **2013**, *51*, 274–281. [CrossRef]
15. Gon, K.H.; Chang, S.H.; Lee, B.H. Pressurized water reactor core parameter prediction using an artificial neural network. *Nucl. Sci. Eng.* **1993**, *113*, 70–76.
16. Souza Rose Mary, G.P.; Moreira, J.M.L. Power peak factor for protection systems-experimental data for developing a correlation. *Ann. Nucl. Energy* **2006**, *33*, 609–621. [CrossRef]
17. Shriver, F.; Gentry, C.; Watson, J. Prediction of Neutronics Parameters Within a Two-Dimensional Reflective PWR Assembly Using Deep Learning. *Nucl. Sci. Eng.* **2021**, *195*, 626–647. [CrossRef]
18. James, B.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
19. Jasper, S.; Larochelle, H.; Adams, R.P. Practical bayesian optimization of machine learning algorithms. *Adv. Neural Inf. Process. Syst.* 2012, 25. Available online: https://arxiv.org/abs/1206.2944 (accessed on 30 October 2024).
20. Jasper, S.; Rippel, O.; Swersky, K.; Kiros, R.; Satish, N.; Sundaram, N.; Patwary, M.; Prabhat, M.; Adams, R. Scalable bayesian optimization using deep neural networks. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015.
21. Takuya, A.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019.
22. Seunghyup, S.; Lee, Y.; Kim, M.; Park, J.; Lee, S.; Min, K. Deep neural network model with Bayesian hyperparameter optimization for prediction of NOx at transient conditions in a diesel engine. *Eng. Appl. Artif. Intell.* **2020**, *94*, 103761.
23. Patrick, K.; Golovidov, O.; Gardner, S.; Wujek, B.; Griffin, J.; Xu, Y. Autotune: A derivative-free optimization framework for hyperparameter tuning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018.
24. Golovin, D.; Solnik, B.; Moitra, S.; Kochanski, G.; Karro, J.; Sculley, D. Google vizier: A service for black-box optimization. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017.
25. Liaw, R.; Liang, E.; Nishihara, R.; Moritz, P.; Gonzalez, J.E.; Stoica, I. Tune: A research platform for distributed model selection and training. *arXiv* **2018**, arXiv:1807.05118.
26. Probst, P.; Wright, M.N.; Boulesteix, A.L. Hyperparameters and tuning strategies for random forest. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2019**, *9*, e1301. [CrossRef]
27. Weerts, H.J.; Mueller, A.C.; Vanschoren, J. Importance of tuning hyperparameters of machine learning algorithms. *arXiv* **2020**, arXiv:2007.07588.
28. Li, H.; Chaudhari, P.; Yang, H.; Lam, M.; Ravichandran, A.; Bhotika, R.; Soatto, S. Rethinking the hyperparameters for fine-tuning. *arXiv* **2020**, arXiv:2002.11770.
29. Bardenet, R.; Brendel, M.; Kégl, B.; Sebag, M. Collaborative hyperparameter tuning. In Proceedings of the International Conference on Machine Learning, Atlanta, GA, USA, 17–19 June 2013; pp. 199–207.
30. Duan, K.; Keerthi, S.S.; Poo, A.N. Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing* **2003**, *51*, 41–59. [CrossRef]

31. Arnold, C.; Biedebach, L.; Küpfer, A.; Neunhoeffer, M. The role of hyperparameters in machine learning models and how to tune them. *Political Sci. Res. Methods* **2024**, *12*, 841–848. [CrossRef]
32. Available online: https://www.kaggle.com/code/prasadperera/the-boston-housing-dataset (accessed on 30 October 2024).
33. Groeneveld, D.C. *Critical Heat Flux Data Used to Generate the 2006 Groeneveld Lookup Tables*; United States Nuclear Regulatory Commission: Rockville, MD, USA, 2019.
34. Yeh, I.-C. Modeling of strength of high-performance concrete using artificial neural networks. *Cem. Concr. Res.* **1998**, *28*, 1797–1808. [CrossRef]
35. Tüfekci, P. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *Int. J. Electr. Power Energy Syst.* **2014**, *60*, 126–140. [CrossRef]
36. Available online: https://archive.ics.uci.edu/dataset/551/gas+turbine+co+and+nox+emission+data+set (accessed on 30 October 2024).
37. Jomaa, H.S.; Grabocka, J.; Schmidt-Thieme, L. Hyp-rl: Hyperparameter optimization by reinforcement learning. *arXiv* **2019**, arXiv:1906.11527.
38. Tuba, E.; Bačanin, N.; Strumberger, I.; Tuba, M. Convolutional neural networks hyperparameters tuning. In *Artificial Intelligence: Theory and Applications*; Springer International Publishing: Cham, Switzerland, 2021; pp. 65–84.
39. Schratz, P.; Muenchow, J.; Iturritxa, E.; Richter, J.; Brenning, A. Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data. *Ecol. Model.* **2019**, *406*, 109–120. [CrossRef]
40. Safarik, J.; Jalowiczor, J.; Gresak, E.; Rozhon, J. Genetic algorithm for automatic tuning of neural network hyperparameters. In Proceedings of the Autonomous Systems: Sensors, Vehicles, Security, and the Internet of Everything, Orlando, FL, USA, 3 May 2018; Volume 10643, pp. 168–174.
41. Cho, H.; Kim, Y.; Lee, E.; Choi, D.; Lee, Y.; Rhee, W. Basic enhancement strategies when using Bayesian optimization for hyperparameter tuning of deep neural networks. *IEEE Access* **2020**, *8*, 52588–52608. [CrossRef]
42. Olof, S.S. A Comparative Study of Black-Box Optimization Algorithms for Tuning of Hyper-Parameters in Deep Neural Networks. Luleå University of Technology: Luleå, Sweden, 2018.
43. Wistuba, M.; Schilling, N.; Schmidt-Thieme, L. Learning hyperparameter optimization initializations. In Proceedings of the 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Paris, France, 19–21 October 2015; pp. 1–10.
44. Wu, J.; Chen, X.Y.; Zhang, H.; Xiong, L.D.; Lei, H.; Deng, S.H. Hyperparameter optimization for machine learning models based on Bayesian optimization. *J. Electron. Sci. Technol.* **2019**, *17*, 26–40.
45. Shahhosseini, M.; Hu, G.; Pham, H. Optimizing ensemble weights and hyperparameters of machine learning models for regression problems. *Mach. Learn. Appl.* **2022**, *7*, 100251. [CrossRef]
46. Brochu, E.; Cora, V.M.; De Freitas, N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv* **2010**, arXiv:1012.2599.
47. Tiep, N.H.; Kim, K.-D.; Heo, J.; Choi, C.-W.; Jeong, H.-Y. A newly proposed data assimilation framework to enhance predictions for reflood tests. *Nucl. Eng. Des.* **2022**, *390*, 111724. [CrossRef]
48. NuScale Power, LLC. *NuScale Standard Plant Design Certification Application*; Chapter Four: Reactor, Part 2, Tier 2, Revision 5; NuScale Power, LLC: Corvallis, OR, USA, 2020.
49. Wilcoxon, F. Individual comparisons by ranking methods. In *Breakthroughs in Statistics: Methodology and Distribution*; Springer: New York, NY, USA, 1992; pp. 196–202.
50. Hsu, H.; Lachenbruch, P.A. *Paired T Test*; Wiley StatsRef: Hoboken, NJ, USA, 2014.
51. Vinod, N.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010.
52. Clevert, D.-A.; Unterthiner, T.; Hochreiter, S. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv* **2015**, arXiv:1511.07289.
53. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (ICLR), Banff, AB, Canada, 14–16 April 2014.
54. Keras Documentation. 2022. Available online: https://keras.io/api/ (accessed on 30 October 2024).