Adaptive Hyperparameter Tuning within Neural Network-based Efficient Global Optimization

Taeho Jeong[1], Pavankumar Koratikere[1], Leifur Leifsson[1 [0000−0001−5134−870X]], Slawomir Koziel[2,3 [0000−0002−9063−2647]], and Anna Pietrenko-Dabrowska[3 [0000−0003−2319−6782]]

[1] School of Aeronautics and Astronautics, Purdue University, West Lafayette, Indiana 47907, USA
{jeong183@purdue.edu, pkoratik@purdue.edu, leifur@purdue.edu}
[2] Engineering Optimization & Modeling Center, Department of Engineering, Reykjavík University, Menntavegur 1, 102 Reykjavík, Iceland
koziel@ru.is
[3] Faculty of Electronics Telecommunications and Informatics, Gdansk University of Technology, Narutowicza 11/12, 80-233 Gdansk, Poland
anna.dabrowska@pg.edu.pl

Abstract. In this paper, adaptive hyperparameter optimization (HPO) strategies within the efficient global optimization (EGO) with neural network (NN)-based prediction and uncertainty (EGONN) algorithm are proposed. These strategies utilize Bayesian optimization and multi-armed bandit optimization to tune HPs during the sequential sampling process either every iteration (HPO-1itr) or every five iterations (HPO-5itr). Through experiments using the three-dimensional Hartmann function and evaluating both full and partial sets of HPs, adaptive HPOs are compared to traditional static HPO (HPO-static) that keep HPs constant. The results reveal that adaptive HPO strategies outperform HPO-static, and the frequency of tuning and number of tuning HPs impact both the optimization accuracy and computational efficiency. Specifically, adaptive HPOs demonstrate rapid convergence rates (HPO-1itr at 28 iterations, HPO-5itr at 26 for full HPs; HPO-1itr at 13, HPO-5itr at 28 iterations for selected HPs), while HPO-static fails to approximate the minimum within the allocated 45 iterations for both scenarios. Mainly, HPO-5itr is the most balanced approach, found to require 21% of the time taken by HPO-1itr for tuning full HPs and 29% for tuning a subset of HPs. This work demonstrates the importance of adaptive HPO and sets the stage for future research.

Keywords: Neural networks, surrogate-based optimization, hyperparameter optimization, sequential sampling.

1 Introduction

The engineering design optimization process merges computational simulations with optimization methods to determine optimal design under specific constraints. This process is typically iterative, which increases the computational

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6

cost. Surrogate-based optimization (SBO) presents an effective solution by employing data-driven surrogate models instead of repetitive simulations, and can save computational cost while yielding optimum results [14].

The efficient global optimization (EGO) algorithm [8] is a widely used SBO technique that typically employs kriging [3] as its surrogate model. This algorithm adopts a sequential sampling method, starting with an initial kriging model. Each iteration involves selecting a new sample point that maximizes the expected improvement (EI) criterion, balancing exploration and exploitation. The new samples are then integrated into the kriging model for subsequent iterations, progressively enhancing the performance [16]. However, the computational cost associated with kriging increases quickly with more samples [12].

Neural networks (NNs) are effective surrogate models for managing large, nonlinear data sets. They learn complex input-output relationships, enabling accurate predictions of new data outcomes [14]. Their performance largely depends on the data's quality and quantity, as NNs do not typically employ sequential sampling, a process often referred to as one-shot optimization or sampling [7]. This is primarily due to their inability to estimate uncertainty, a requirement for exploration. Sequential sampling could lead to convergence at local minima without exploration and limiting further refinement of the surrogate model. Therefore, optimizing an initial data set size is important when using NNs [17].

Koratikere et al. [10] recently introduced the efficient global optimization using neural networks (EGONN) algorithm, which combines sequential sampling with NN-based predictions. This method uniquely employs a secondary NN to model uncertainty, allowing EGO to combine with NNs' predictive abilities for sequential sampling. A significant advantage of this approach is simplifying the preliminary setup and enhancing the efficiency of NNs in global modeling and optimization tasks [9].

The structure of NNs is defined using hyperparameters (HPs), including structural parameters and learning algorithms, which need to be set before training. Consequently, HP optimization (HPO) is important for constructing an optimally performing NN [17,15]. In one-shot optimization, the optimal HP configuration is determined for the initial data set and kept static throughout the algorithm run [7]. However, HPs require occasional tuning as the optimal NN structure and algorithms may vary with the number of training samples for sequential sampling.

This work proposes an adaptive HPO for sequential sampling within the EGONN algorithm. This adaptive HPO addresses the limitations associated with static HP, offering a more dynamic and responsive optimization strategy. It combines Bayesian optimization (BO) [15] and multi-armed bandit (MAB) optimization [5] to optimize continuous and discrete HPs, respectively. These strategies balance exploration and exploitation, enabling efficient identification of the HP configuration. This integrated approach facilitates finding the best HP configuration with fewer design iterations and reduced time [1,4].

Additionally, this paper conducts a comparative analysis of different HPO strategies within the EGONN framework, examining static HP tuning, tuning

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6

HPs every five iterations, and every iteration. These strategies are evaluated based on metrics such as the number of infill points required for convergence and the time cost, using the EGO as a benchmark. The analysis utilizes the three-dimensional Hartmann function by tuning a complete set and a subset of HPs for numerical experiments.

The remainder of the paper is structured as follows. Section 2 provides an overview of the employed SBO methods and HPO. Section 3 describes numerical experiments for evaluating the proposed methods. Finally, Section 4 summarizes the essential findings and identifies potential future research directions.

2 Methods

This section introduces the basics of SBO and specific sequential sampling algorithms for optimization. It then describes the HPO methods and adaptive HPO strategies employed in this work.

2.1 Surrogate-based optimization

SBO is effective for design optimization problems that minimize an objective function $f(\mathbf{x}_d)$, subject to constraints $g_i(\mathbf{x}_d) \leq 0$ and $h_j(\mathbf{x}_d) = 0$, which can be written as:

$$\begin{aligned} \min_{\mathbf{x}_d} \quad & f(\mathbf{x}_d) \\ \text{subject to} \quad & g_i(\mathbf{x}_d) \leq 0,\ i = 1, \ldots, q, \\ & h_j(\mathbf{x}_d) = 0,\ j = 1, \ldots, r, \end{aligned} \tag{1}$$

where $\mathbf{x}_d = (x_{d_1}, x_{d_2}, ..., x_{d_n})$ is the vector of design variables, q is the number of the inequality constraints, and r is the number of the equality constraints. These problems often require extensive evaluation using simulations, making SBO a valuable tool for efficiently finding solutions using a data-driven surrogate model instead of directly evaluating the objective function or constraints [3]. The effectiveness of SBO, however, is contingent upon the accuracy of the surrogate model. Inaccurate approximations can result in sub-optimal optimization outcomes. Furthermore, constructing a reliable surrogate model requires substantial initial data. SBO iteratively refines the surrogate model by adding a sample point, resulting in progressively more accurate approximations of the objective function. [16].

Figure 1 presents a flowchart outlining the SBO process with sequential sampling utilized in this research. The process is initiated by generating an initial set of samples through a deterministic Halton sequence sampling[2]. These samples are used to acquire observations for training the surrogate models, such as NN or kriging. Then, infill criteria, such as the expected improvement, are maximized to explore and exploit the design space using differential evolution [11]. The new infill point is added to the training data for the next iteration, continuing until the allocated budget is exhausted or a convergence criterion is satisfied.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6

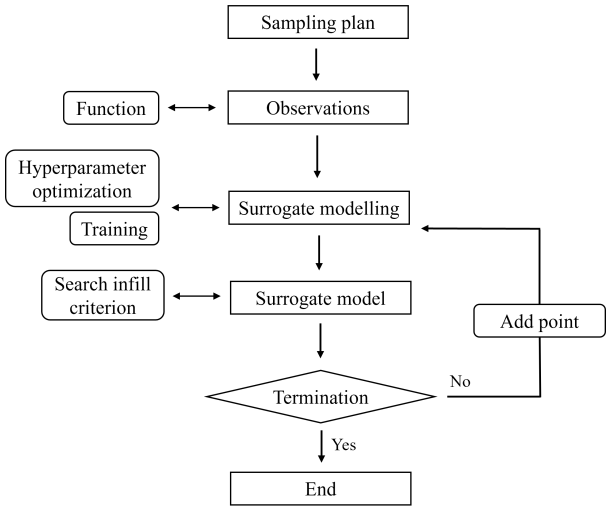

Fig. 1: A flowchart illustrating SBO with sequential sampling and adaptive HPO.

2.2 Efficient global optimization

EGO is a widely used SBO method to optimize a black box function that uses a kriging model for prediction and uncertainty estimation [8]. As described in Algorithm 1, the process begins by generating an initial sample plan $\mathbf{X}$ and observing the corresponding outputs $\mathbf{Y}$. In each iteration, a kriging model is created using the available data set. Then, EI [3] is maximized to find the next infill point $\mathbf{p}$, which balances the exploration of areas characterized by high uncertainty against the exploitation of areas with low estimated function values. The EI function is formulated as

$$EI(\mathbf{x}) = \left[f(\mathbf{x}^*) - \hat{f}(\mathbf{x})\right]\Phi(Z) + \hat{s}(\mathbf{x})\phi\left(\frac{f(\mathbf{x}^*) - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right), \tag{2}$$

Algorithm 1 Efficient global optimization [8].

1: Create initial sampling plan $\mathbf{X}$
2: Compute objective function value $\mathbf{Y}$ at initial sampling $\mathbf{X}$
3: **while** infill budget lasts **do**
4: Create kriging model using $(\mathbf{X}, \mathbf{Y})$
5: Find the best sample y^* in $\mathbf{Y}$
6: Maximize $EI(\mathbf{x})$ to find next infill point $\mathbf{p}$
7: Compute objective function value $\mathbf{y}$ at $\mathbf{p}$
8: Append $(\mathbf{X}, \mathbf{Y})$ with $(\mathbf{p}, y)$
9: **end while**
10: Return best sample $(\mathbf{x}^*, y^*)$ in appended data set

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6

where y^* is the best objective function value in the data set, $\hat{y}(\mathbf{x})$ and $\hat{s}(\mathbf{x})$ represent the surrogate model prediction and the corresponding uncertainty in the prediction. Φ and ϕ are the cumulative distribution function and probability density function of the standard normal distribution, respectively, and the standard normal variable (Z) is evaluated as:

$$Z = \frac{y^* - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}. \tag{3}$$

The new observation $\mathbf{y}$ is evaluated at infill point $\mathbf{p}$ and is appended to the data set $(\mathbf{X}, \mathbf{Y})$. The kriging model is retrained with an updated data set, and this iterative cycle is continued until the budget for infill is exhausted. Finally, the best-observed sample y^* and corresponding design variable vector $\mathbf{x}^*$ is returned as the optimum design.

2.3 EGO using neural networks

The recently developed EGONN algorithm enhances the EGO process by incorporating two separate NNs for modeling the objective function and uncertainty [10]. The first NN (NN_y) models the objective function and the second NN (NN_u) models uncertainty. A third NN (NN_g) is added if there are constraints in the problem [9]. As described in Algorithm 2, the EGONN process is similar to EGO but there are two major differences. The process starts with generating two different sampling plans, $\mathbf{X}$ and $\mathbf{X}_u$, with corresponding function values Y and Y_u. Within the iterative cycle, a NN model for $\hat{y}$ is created using ($\mathbf{X}$,$\mathbf{Y}$). Then, $\hat{y}$ model is used for predicting the values at $\mathbf{X}$ and $\mathbf{X}_u$. The prediction error can be computed since the true values are available. In the next step, the $\hat{s}$ model is created using $\mathbf{X}$ and $\mathbf{X}_u$, and the corresponding prediction error. In this way, EGONN creates two NN models, one for prediction and the other for uncertainty estimation.

The remaining part of the cycle is the same as EGO. The next step is to maximize EI($\mathbf{x}$) using the prediction and uncertainty models to find the next

Algorithm 2 Efficient global optimization using neural networks [10].

1: Create initial sampling plan $\mathbf{X}$ and $\mathbf{X}_u$
2: Compute objective function value $\mathbf{Y}$ and $\mathbf{Y}_u$ for initial samples
3: **while** infill budget lasts **do**
4: Create NN model for $\hat{y}$ using $(\mathbf{X}, \mathbf{Y})$
5: Create NN model for $\hat{s}$ using prediction error of $\hat{y}$ for $\mathbf{X}$ and $\mathbf{X}_u$
6: Find the best sample y^* in $\mathbf{Y}$
7: Maximize $EI(\mathbf{x})$ to find next infill point $\mathbf{p}$
8: Compute objective function value $\mathbf{y}$ at $\mathbf{p}$
9: Append $(\mathbf{X}, \mathbf{Y})$ with ($\mathbf{p}$,y)
10: **end while**
11: Return best sample ($\mathbf{x}^*$,y^*) in appended data set

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6

Table 1: A table of the HPs with their respective types and ranges or options.

Name	Type	Bounds / options
Learning rate (x_1)	$\mathbb{R}^+$	$10^{-4} \leq x_1 \leq 10^{-1}$
Epoch number (x_2)	$\mathbb{Z}^+$	$10^3 \leq x_2 \leq 5 \cdot 10^3$
Activation function (x_3)	[-]	(relu, elu, tanh, sigmoid)
Optimizer (x_4)	[-]	(SGD, Adam, RMSprop, Adagrad)
Hidden layer number (x_5)	$\mathbb{Z}^+$	$1 \leq x_5 \leq 4$
Initial neuron number (x_6)	$\mathbb{Z}^+$	$4 \leq x_6 \leq 16$

infill point. The function value at the infill point is evaluated, and the ($\mathbf{X}$,$\mathbf{Y}$) data set is updated. This cycle is repeated until the infill budget is exhausted. Recently, a constrained version of EGONN was proposed, which uses a third NN model for representing the constraints [9].

2.4 Hyperparameter optimization

A critical step while creating an NN model involves configuring its structure and the training algorithm, which are problem-dependent and must be tuned for good performance [7]. Table 1 illustrates the HPs used in this study with their respective ranges and details. The number of hidden layers and neurons defines the NN's depth and pattern recognition capacity of input data features. The activation function introduces non-linearity in the NN, which is essential to recognize complex patterns. The learning rate controls the optimization pace to ensure efficient learning while the optimizer guides the learning through weight updates. Finally, the epoch size indicates the number of iterations of the optimization algorithm [18].

The HPO problem involves minimizing the root mean squared error (RMSE) of NN prediction on the validation set with respect to $\mathbf{x} = (x_1, x_2, \ldots, x_6)$, where each x_i denotes a distinct tunable HP within its respective bounds. The HPO process is executed during steps 4 and 5 of Algorithm 2 for all the NNs, which is written as

$$\mathbf{x}^* = \min_{\mathbf{x} \in \chi} \mathrm{RMSE}(\mathbf{x}), \tag{4}$$

where $\mathbf{x}^*$ is the optimal HP configuration, χ is the HP domain, and RMSE is the discrepancy between a model's predictions $\hat{f}(\mathbf{x}, \mathbf{x}_d)$ and actual observation $f(\mathbf{x}_d)$ for the number of validation data points n_v, which is formulated as

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n_v} [f(\mathbf{x}_d^{(i)}) - \hat{f}(\mathbf{x}, \mathbf{x}_d^{(i)})]^2}{n_v}}. \tag{5}$$

The optimization process concludes upon meeting a predefined convergence criterion, typically defined as either reaching a maximum number of HP sets or attaining a target loss function value.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6

This paper evaluates the impact of different HPO frequencies on NN performance and computational efficiency. It contrasts static HPO with tuning the HPs every five iterations and every iteration, focusing on the optimal balance between their time efficiency and effectiveness in optimizing the function.

BO [15] offers a solution for continuous HPs with infinite choices. It adopts an adaptive strategy, using observations from previous evaluations to decide subsequent sets of the HPs to evaluate. BO begins by constructing a smooth surrogate model using a Gaussian process (GP), characterized by a mean function, which offers the best prediction based on existing observations, and a kernel function, which measures prediction uncertainty [17,12]. The radial basis function (RBF) kernel [15], also known as the Gaussian kernel, is used in GP regressions, quantifying similarities between data points. The RBF kernel includes HPs such as the length scale and the variance, which are internally tuned in BoTorch [1].

The predictions and uncertainty estimates are combined to form an acquisition function. In this study, the EI in (2) is taken to be the acquisition function, which is maximized to select the most promising HP configuration for the next observation. After each observation, the surrogate model is updated, and this iterative process continues until it converges with the optimal HP configuration.

MAB optimization [5], in contrast to BO, is particularly suited for discrete HPs with a finite set of choices. In this work, MAB optimization utilizes the Thompson sampling (TS) algorithm [4] for sequentially selecting discrete HP sets. TS bases its choice on the probability of each HP set being the most effective, balancing between exploring lesser-known options and exploiting those that have yielded promising results.

TS begins by assigning a prior distribution to each HP configuration's reward probability. For every iteration, it samples a reward value from the distribution of each HP configuration, selecting the one with the highest sampled value. The observed reward from the chosen configuration updates its distribution. This iterative process continues until the HP configuration with the highest reward probability distribution is identified, which is the optimal HP configuration [4].

3 Numerical Experiments

This section presents the results from employing HPO strategies for SBO with sequential sampling, applied to a three-dimensional Hartmann function. The investigation explores the impact of varying the number of HPs optimized in each scenario.

3.1 Problem formulation of the Hartmann function

The three-dimensional Hartmann function minimization problem [13], which has four local minima and one global minimum, is defined as:

$$\min_{\mathbf{x}_d} f(\mathbf{x}_d) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}(x_{dj} - P_{ij})^2\right), \tag{6}$$

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6

where $\alpha = [1.0, 1.2, 3.0, 3.2]^T$,

$$\mathbf{A} = \begin{bmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix},$$

and

$$\mathbf{P} = 10^{-4} \begin{bmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}.$$

The lower and upper bounds are $\mathbf{x}_{d_L} = (0, 0, 0)$ and $\mathbf{x}_{d_U} = (1, 1, 1)$, respectively, with the global minimum $f(\mathbf{x}_\mathbf{d}^*)$ of -3.86 located at $\mathbf{x}_d^* = (0.11, 0.56, 0.85)$.

3.2 Strategies and algorithm setup

HPO strategies within the EGONN framework include static HPO (HPO-static), tuning HPs every five iterations (HPO-5itr), and tuning every iteration (HPO-1itr). The HPs are tuned using the open-source tool Ax [1] built on PyTorch [6], which integrates BO and MAB optimization. Ax constructs a combined search space for the continuous and discrete HPs where each type of HP is optimized by its respective algorithm simultaneously.

The minimization problem (6) is tackled using the SBO methodology depicted in Fig. 1 and EGONN algorithm (cf. Alg. 2). The initial sample size is 10 with 45 infill points. Deterministic Halton sequence sampling [2] ensures consistency in the starting points across all methods. This study evaluates the efficacy of various HPO strategies by comparing their convergence performance and time efficiency against the traditional EGO method (cf. Alg. 1) as a benchmark. Initially, a full set of the HPs, as detailed in Table 1, is optimized, followed by the optimization of a subset of the HPs based on preliminary results.

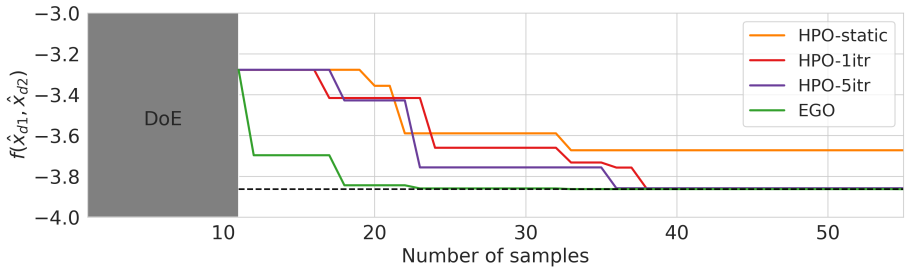

Fig. 2: Optimization of the three-dimensional Hartmann function utilizing EGO and EGONN with various HPO strategies that tune a full set of HPs.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6

3.3 Optimizing the full set of HPs

The minimization of the three-dimensional Hartmann function is explored through EGONN, employing NNs optimized with a set of six tunable HPs, and EGO. Figure 2 illustrates varied convergence rates towards the minimum with increasing sample numbers. While all methods gradually approach the minimum, they require different numbers of samples for accurate approximation. The results indicate that while EGO outperforms EGONN, the two adaptive HPO strategies show comparable performance. EGO achieves convergence in 12 infills, whereas EGONN with HPO-5itr and HPO-1itr take 26 and 28 infills, respectively. HPO-static shows no significant improvement after 22 infills.

Figure 3 presents the history of six HPs across various HPO strategies. While the HPO-static approach uses a fixed HP configuration, including the ReLU ac-

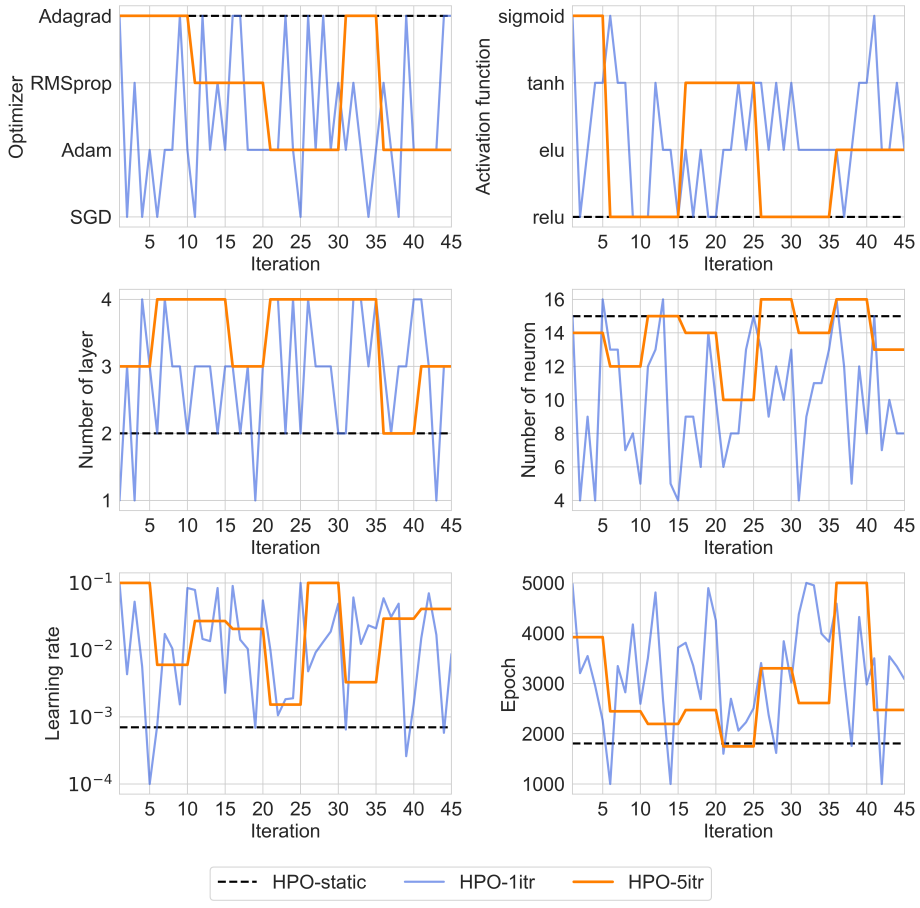

Fig. 3: Convergence analysis of 6 HPs across iterations of different HPO strategies within the EGONN algorithm for optimizing a three-dimensional Hartmann function.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6

Table 2: Percentage of each HP calculated for each method using a full set of HPs.

Hyperparameter		Methods		
		HPO-1itr	HPO-5itr	HPO-static
Activation	relu	20.0%	**37.5%**	**100%**
	elu	**44.4%**	25.0%	0.0%
	tanh	28.9%	25.0%	0.0%
	sigmoid	6.7%	12.5%	0.0%
Optimizer	SGD	15.6%	0.0%	0.0%
	Adam	**48.9%**	**44.4%**	0.0%
	RMSprop	11.1%	22.2%	0.0%
	Adagrad	24.4%	33.3%	**100%**
Layer	1	8.9%	0.0%	0.0%
	2	20.0%	11.1%	**100%**
	3	**46.7%**	33.3%	0.0%
	4	24.4%	**55.6%**	0.0%
Neuron	$4 \sim 8$	**37.8%**	0.0%	0.0%
	$8 \sim 12$	31.1%	11.1%	0.0%
	$12 \sim 16$	31.1%	**88.9%**	**100%**
Epoch	$1 \sim 2$ (10^3)	13.3%	11.1%	**100%**
	$2 \sim 3$ (10^3)	26.7%	**55.6%**	0.0%
	$3 \sim 4$ (10^3)	**37.8%**	22.2%	0.0%
	$4 \sim 5$ (10^3)	22.2%	11.1%	0.0%
LR	$10^{-3} \sim 10^{-4}$	11.1%	0.0%	**100%**
	$10^{-2} \sim 10^{-3}$	26.7%	33.3%	0.0%
	$10^{-1} \sim 10^{-2}$	**60.0%**	**66.7%**	0.0%

tivation function, Adagrad optimizer, two hidden layers, 15 neurons, a learning rate of 0.0007, and 1805 epochs for NN_y, adaptive HPO strategies show enhanced flexibility and effectiveness by adjusting HPs to fit better the evolving data set size. The trends, however, indicate that the HP convergence is not directly correlated with increased sample size. This is attributed to the nonlinear and multimodal relationship between HPs and optimization performance, indicating multiple HP configurations can yield a similar optimized NN performance. This suggests continuous HP tuning with each added infill point may be unnecessary, yet adaptive HPO is critical for identifying the most effective HPs for NNs across different data set sizes.

Table 2 outlines the percentage distribution of HP selections, with bolded most frequently chosen HPs. The data reveals that the choice of activation functions is nearly uniformly distributed for both the adaptive HPO strategies, indicating no clear preference. However, a noticeable trend is the selection of a higher number of layers, suggesting a move towards more complex NN architectures as sample sizes increase. Additionally, the Adam optimizer, alongside higher learning rates and epoch sizes, is consistently preferred in the adaptive HPO strategies, indicating the critical HPs for achieving convergence. The pref-

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6

Table 3: Time cost for iterations of HPO of a full set of HPs and surrogate training in the three-dimensional Hartmann function optimization.

Time cost (hr)		Methods			
		HPO-1itr	HPO-5itr	HPO-static	EGO
HPO	Mean	0.21	0.19	-	-
	SD	0.09	0.08	-	-
	Total time	9.40	1.75	0.09	-
Training	Mean	$8.9{\cdot}10^{-3}$	$9.1{\cdot}10^{-3}$	$6.9{\cdot}10^{-3}$	$1.6{\cdot}10^{-4}$
	SD	$3.1{\cdot}10^{-3}$	$3.6{\cdot}10^{-3}$	$1.7{\cdot}10^{-3}$	$1.1{\cdot}10^{-5}$
	Total time	0.36	0.35	0.32	$7.5{\cdot}10^{-3}$
Cumulative time		9.76	2.10	0.41	$7.5{\cdot}10^{-3}$

erence for HPs in the adaptive HPO strategies explains the limitations of the static HPO approach. As sample sizes increase, the necessity for complex HP configurations grows, emphasizing the effectiveness of adaptive HPO strategies.

A comprehensive analysis of the time costs throughout all 45 iterations is presented in Table 3. It specifies each method's mean and standard deviation (SD) of HPO and surrogate training per iteration along with the total time cost, demonstrating significant variations in time cost. This variability is mainly attributed to the growing training data sets as iterations progress. Despite these variations in HPO time costs, the time required to train NNs shows minimal differences across strategies. The finding reveals that the time cost of HPO-1itr is 465% of HPO-5itr's time cost, primarily due to the frequency of HPO. Consequently, HPO-5itr stands out as the most efficient EGONN approach, balancing the convergence speed and the overall computational time.

3.4 Optimizing a subset of the HPs

Analyzing the optimization convergence history in Fig. 2 and the HP distribution in Table 2 suggests that a strategic HP selection could enhance the efficiency

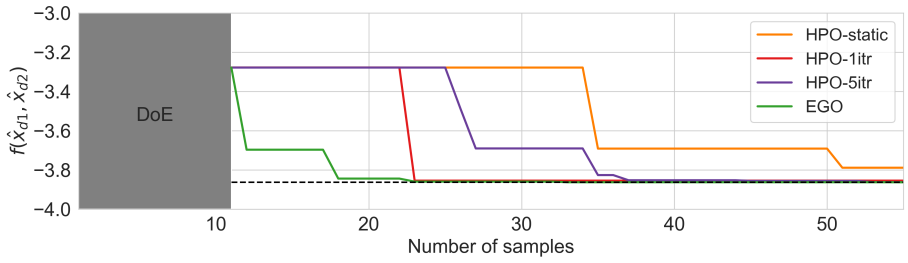

Fig. 4: Optimization of the three-dimensional Hartmann function utilizing EGO and EGONN with HPO-static, HPO-5itr, and HPO-1itr strategies for a subset of HPs.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6

of HPO. The activation function, number of epochs, and learning rate are identified as critical HPs for tuning. Although the optimizer choice also influences performance, the consistent selection of the Adam optimizer from the previous evaluation reflects its effectiveness for this test case. The preference for a high learning rate necessitates further optimization because it is a continuous parameter offering a broad spectrum of possibilities. Based on the superior performance of HPO-5itr in the previous evaluation, the mean values for the number of neurons and layers are adopted, which are 13 and 3, respectively. The options for the activation function and the ranges of the epoch size and the learning rate are maintained as detailed in Table 1.

Figure 4 displays the convergence performance of each HPO strategy on the subset of the HPs. Despite a similar trend to the previous test, noticeable improvements are observed in the HPO-static and HPO-1itr strategies. HPO-1itr achieves early convergence at 13 iterations, while HPO-static approaches closer to the minimum from -3.67 to -3.79. In contrast, HPO-5itr shows limited improvement, converging at 25 iterations.

The convergence history and HP distribution, as shown in Fig. 5 and Table 4, indicate the differences in HP selection across the strategies. The improved performance of the HPO-static is attributed to its initial choice of a high learning rate. However, the HPO-static strategy chose a relatively low epoch size, contrasting with the higher epoch sizes preferred by adaptive HPO strategies.

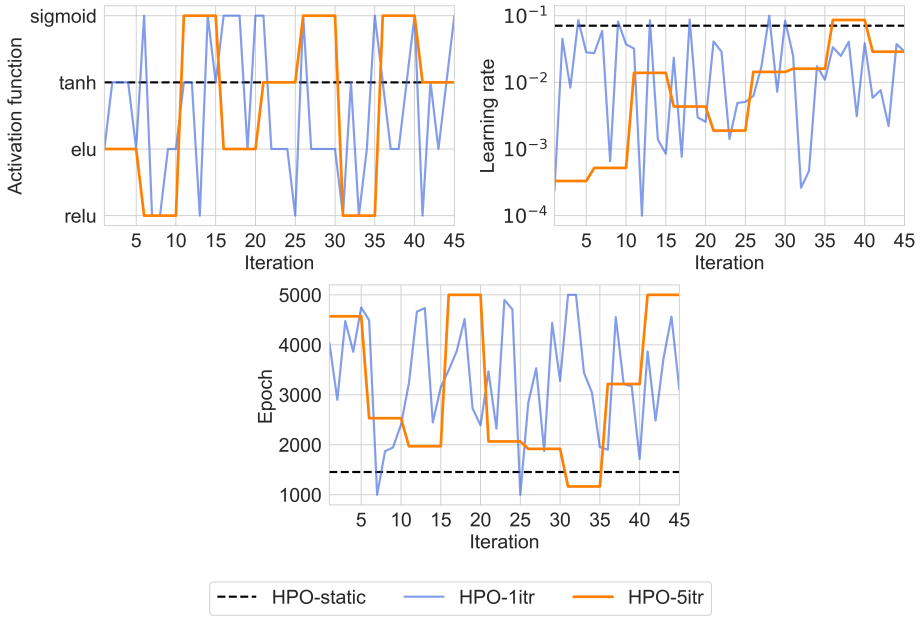

Fig. 5: Convergence analysis of 3 HPs with various HPO strategies in EGONN for the three-dimensional Hartmann function optimization.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6

Table 4: Percentage distribution of each HP for the test using a subset of the HPs.

Hyperparameter		Methods		
		HPO-1itr	HPO-5itr	HPO-static
Activation	relu	15.6%	22.2	0.0%
	elu	**35.6%**	22.2%	0.0%
	tanh	24.4%	22.2%	**100%**
	sigmoid	24.4%	**33.3%**	0.0%
Epoch	$1 \sim 2$ (10^3)	17.8%	22.2%	**100%**
	$2 \sim 3$ (10^3)	17.8%	22.22%	0.0%
	$3 \sim 4$ (10^3)	**33.3%**	11.1%	0.0%
	$4 \sim 5$ (10^3)	31.1%	**44.4%**	0.0%
LR	$10^{-3} \sim 10^{-4}$	13.3%	22.2%	0.0%
	$10^{-2} \sim 10^{-3}$	28.9%	22.2%	0.0%
	$10^{-1} \sim 10^{-2}$	**55.6%**	**55.6%**	**100%**

Despite a similar HP distribution to the previous test for HPO-1itr, retaining the optimizer and the number of neurons and layers enhanced convergence performance. Additionally, the adaptation of activation functions to sigmoid for HPO-5itr and tanh for HPO-static indicates strategic adjustments to improve optimization outcomes.

The analysis reveals a significant decrease in time costs, primarily attributed to the decreased time required for tuning fewer HPs, as detailed in Table 5. EGONN with the strategy of HPO-1itr requires 3.79 hours, HPO-5itr 1.01 hours, and HPO-static 0.40 hours to for optimization. It demonstrates substantial time savings in HPO compared to the previous case outlined in Table 3. Specifically, HPO-1itr saves 5.97 hours for 45 HPOs, HPO-5itr saves 1.04 hours for 9 HPOs, and HPO-static saves 0.04 hours for a single HPO. This comparison indicates that reducing the number of HPOs directly correlates with increased time efficiency. It notably reduces the gap between HPO-1itr and HPO-5itr from 7.66 hours to 2.75 hours, stressing the importance of the streamlined HPO processes.

Table 5: Time cost of HPO and surrogate training in optimizing the three-dimensional Hartmann function with a selected subset of the HPs.

Time cost (s)		Methods			
		HPO-1itr	HPO-5itr	HPO-static	EGO
HPO	Mean	0.08	0.08	-	-
	SD	0.02	0.01	-	-
	Total time	3.43	0.71	0.05	-
Training	Mean	$8.7{\cdot}10^{-3}$	$8.6{\cdot}10^{-3}$	$6.7{\cdot}10^{-3}$	$1.6{\cdot}10^{-4}$
	SD	$2.9{\cdot}10^{-3}$	$3.2{\cdot}10^{-3}$	$1.6{\cdot}10^{-3}$	$1.1{\cdot}10^{-5}$
	Total time	0.33	0.30	0.36	$7.5{\cdot}10^{-3}$
Cumulative time		3.79	1.01	0.41	$7.5{\cdot}10^{-3}$

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6

4 Conclusion

This paper proposes the use of adaptive hyperparameter optimization (HPO) strategies within the efficient global optimization (EGO) with neural network (NN)-based prediction and uncertainty (EGONN). Numerical experiments with the three-dimensional Hartmann function revealed that while traditional EGO is effective, integrating NNs through EGONN with HPO strategies enhances performance for optimization problems.

Moreover, this study demonstrates the importance of appropriate HPO in the success of SBO for their adaptive tuning as the complexity of the data set increases. It shows that a static HPO strategy could not adapt to the sequential sampling process, leading to sub-optimal performance. In contrast, the adaptive HPO strategies demonstrated their ability to converge with fewer iterations by effectively reaching the optimum. Tuning HPs every five iterations emerged as the most efficient approach, balancing optimization performance and computational burden. Additionally, the optimization of a subset of the HPs reveals that the most effective adaptive HPO strategies can vary as the number of tunable HPs changes.

Future research directions include establishing a predefined tolerance threshold based on numerical metrics, such as the loss function values, and identifying a subset of the HPs for iterative tuning guided by global sensitivity analysis. Such investigation could yield more reliable NN performance throughout the SBO iterations while saving computational resources. Moreover, expanding the scope of test cases to more complex and higher-dimensional problems, such as airfoil shape optimization in transonic viscous flow, will provide further insights into the efficiency of EGONN with adaptive HPO versus EGO.

Acknowledgments

This work was supported in part by the U.S. National Science Foundation (NSF) award number 2223732 and by the Icelandic Centre for Research (RANNIS) award number 239858.

References

1. Baird, S.G., Liu, M., Sparks, T.D.: High-dimensional bayesian optimization of 23 hyperparameters over 100 iterations for an attention-based network to predict materials property: A case study on crabnet using ax platform and saasbo. Computational Materials Science **211**, 111505 (2022). https://doi.org/10.1016/j.commatsci.2022.111505
2. Faure, H., Lemieux, C.: Generalized halton sequences in 2008: A comparative study. ACM Transactions on Modeling and Computer Simulation (TOMACS) **19**(4), 1–31 (2009). https://doi.org/10.1145/1596519.1596520
3. Forrester, A.I., Keane, A.J.: Recent advances in surrogate-based optimization. Progress in Aerospace Sciences **45**(1), 50–79 (2009). https://doi.org/10.1016/j.paerosci.2008.11.001

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6

4. Gupta, N., Granmo, O.C., Agrawala, A.: Thompson sampling for dynamic multi-armed bandits. In: 2011 10th International Conference on Machine Learning and Applications and Workshops. vol. 1, pp. 484–489 (Honolulu, Hawaii, December 18-21, 2011). https://doi.org/10.1109/ICMLA.2011.144
5. Henri, S., Vlachou, C., Thiran, P.: Multi-armed bandit in action: Optimizing performance in dynamic hybrid networks. IEEE/ACM Transactions on Networking **26**(4), 1879–1892 (2018). https://doi.org/10.1109/TNET.2018.2856302
6. Imambi, S., Prakash, K.B., Kanagachidambaresan, G.: Pytorch. Programming with TensorFlow: Solution for Edge Computing Applications pp. 87–104 (2021). https://doi.org/10.1007/978-3-030-57077-4$_1$0
7. Jeong, T., Koratikere, P., Leifsson, L.T.: Automated hyperparameter tuning for airfoil shape optimization with neural network models. In: AIAA SCITECH 2024 Forum. p. 2671 (Orlando, FL, USA, 8-12 Jan, 2024). https://doi.org/10.2514/6.2024-2671
8. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization **13**(4), 455–492 (1998). https://doi.org/10.1023/A:1008306431147
9. Koratikere, P., Leifsson, L., Koziel, S., Pietrenko-Dabrowska, A.: Constrained Aerodynamic Shape Optimization Using Neural Networks and Sequential Sampling, p. 425–438. Springer Nature Switzerland (Prague, Czech Republic, 3-5 July, 2023). https://doi.org/10.1007/978-3-031-36024-4$_3$3
10. Koratikere, P., Leifsson, L.T., Barnet, L., Bryden, K.: Efficient global optimization algorithm using neural network-based prediction and uncertainty. In: AIAA SCITECH 2023 Forum. AIAA (National Harbor, MD, USA, 23-27 Jan, 2023). https://doi.org/10.2514/6.2023-2683
11. Lampinen, J.: A constraint handling approach for the differential evolution algorithm. In: Proceedings of the 2002 Congress on Evolutionary Computation, volume=2, pages=1468–1473, year=2002, organization=IEEE, doi=10.1109/CEC.2002.1004459"}
12. Lim, Y.F., Ng, C.K., Vaitesswar, U., Hippalgaonkar, K.: Extrapolative bayesian optimization with gaussian process and neural network ensemble surrogate models. Advanced Intelligent Systems **3**(11), 2100101 (2021). https://doi.org/10.1002/aisy.202100101
13. Mehdad, E., Kleijnen, J.: Classic kriging versus kriging with bootstrapping or conditional simulation. Journal of the Operational Research Society **66** (03 2015). https://doi.org/10.1057/jors.2014.126
14. Queipo, N.V., Haftka, R.T., Shyy, W., Goel, T., Vaidyanathan, R., Tucker, P.K.: Surrogate-based analysis and optimization. Progress in Aerospace Sciences **41**(1), 1–28 (2005). https://doi.org/10.1016/j.paerosci.2005.02.001
15. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. Advances in Neural Information Processing Systems **25** (2012)
16. Viana, F.A., Haftka, R.T., Watson, L.T.: Efficient global optimization algorithm assisted by multiple surrogate techniques. Journal of Global Optimization **56**, 669–689 (2013). https://doi.org/10.1007/s10898-012-9892-5
17. 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. Journal of Electronic Science and Technology **17**(1), 26–40 (2019). https://doi.org/10.11989/JEST.1674-862X.80904120
18. Yang, L., Shami, A.: On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing **415**, 295–316 (2020). https://doi.org/10.1016/j.neucom.2020.07.061

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_6