Energy Distance Based Similarity Analysis of Hyperparameter Optimization Results for Random Forests

Thuy Thi Tran [1,2], Nghia Quoc Phan [1], and Hiep Xuan Huynh [3,4,*]

[1] Assessment Office, Tra Vinh University, Tra Vinh, Vietnam
[2] Network Management Center, University of Cuu Long, Vinh Long, Vietnam
[3] College of Information and Communication Technology, Can Tho University, Can Tho, Vietnam
[4] CTU Leading Research Team on Automation, Artificial Intelligence, inforMation tEchnology and Digital Transformation, Can Tho University, Can Tho, Vietnam
Email: tranthithuy.dhcl@gmail.com (T.T.T.); nghiavnt@tvu.edu.vn (N.Q.P.); hxhiep@ctu.edu.vn (H.X.H.)
*Corresponding author

***Abstract*—In this study, we propose a novel approach for analyzing the results of Hyperparameter Optimization (HPO) for Random Forest (RF) models by applying Energy Distance (ED), a metric based on pairwise Euclidean distances. This method provides a quantitative measure of the similarities and differences between the configurations of hyperparameters and their corresponding performance metrics. We use a dataset from a hyperparameter optimization experiment for RF, where we explore the relationship between hyperparameter settings and model accuracy. The results indicate that Energy Distance can offer useful insights into the proximity of different hyperparameter configurations and help identify clusters of similar configurations, which can be useful for model selection and optimization.**

***Keywords*—energy distance, hyperparameter optimization, random forest, Euclidean distance, machine learning**

I. Introduction

Hyperparameter optimization is crucial for developing machine learning models, as it significantly affects model accuracy, robustness, and generalization [1–4]. Hyperparameters, such as the number of trees in a Random Forest or the learning rate in gradient boosting, are set before training and govern the model's structure and training process [5–7]. Unlike model parameters, which are learned during training, hyperparameters play a vital role in the model's effectiveness. Incorrect choices can lead to overfitting or underfitting, resulting in poor performance [8–10].

For ensemble methods like Random Forests, the optimal selection of hyperparameters can dramatically influence model accuracy. Key hyperparameters, such as tree depth and the minimum number of samples required to split a node, control model complexity and generalization. Incorrect choices can lead to overfitting or underfitting, both of which degrade performance [2]. Thus, fine-tuning hyperparameters is essential for achieving the right balance between bias and variance, which enhances accuracy and generalization.

Traditional hyperparameter optimization methods, like grid search and random search, are widely used but have limitations. Grid search exhaustively tests all combinations, which is computationally expensive, especially with high-dimensional spaces. Random search is more efficient but may miss optimal configurations due to its lack of strategic exploration [11, 12]. Both methods fail to capture complex interactions between hyperparameters, which are often crucial in finding the best model configurations [1, 2].

The importance of hyperparameters is particularly critical in AI applications like predictive analytics, healthcare, finance, and autonomous systems, where optimal configuration can lead to reliable and efficient decision-making [13–17]. Recent methods like Bayesian optimization and Hyperband aim to improve efficiency by using past performance data to guide exploration. However, these methods still struggle to fully capture the complex relationships between hyperparameters [18–21].

This study introduces Energy Distance as a novel tool for analyzing hyperparameter optimization results in Random Forests. Energy Distance (ED), based on the Euclidean distance between data points, helps uncover similarities or dissimilarities between hyperparameter configurations and their performance distributions [22–24]. Unlike traditional optimization techniques, ED reveals insights into the structure of the hyperparameter space and identifies clusters of configurations with similar performance outcomes. This provides a deeper understanding of hyperparameter interactions, allowing for more informed optimization decisions.

Manuscript received January 24, 2025; revised February 27, 2025; accepted May 9, 2025; published September 19, 2025.

Research Questions are as follows:

How can Energy Distance be applied to analyze hyperparameter optimization results in ensemble methods like Random Forests?

What insights can ED provide about the relationships between hyperparameters and model performance?

We hypothesize that applying Energy Distance to hyperparameter optimization will uncover patterns in the search space, leading to better identification of optimal configurations and insights into hyperparameter interactions that traditional methods may overlook. ED is particularly suited for this task, as it does not assume a specific data distribution, making it ideal for the non-normal distributions common in hyperparameter optimization [24]. By calculating ED between pairs of configurations, we aim to understand performance dynamics and identify configurations likely to produce optimal results.

The paper is organized as follows: Section II reviews related work on hyperparameter optimization and distance measures, focusing on Euclidean Distance. Section III covers hyperparameter optimization and the role of ED. Section IV outlines the methodology, including the hyperparameter search space, Random Forest model, and ED application. Section V presents experimental results, comparing ED with other techniques. Section VI concludes with key findings, advantages, limitations, and future research directions.

II. Related Works

The task of hyperparameter optimization and similarity analysis in machine learning models has been widely studied in recent years. Various methods, including distance measures, clustering techniques, and machine learning approaches, aim to enhance model selection and optimize performance.

Hyperparameter optimization is key to improving machine learning models. Traditional methods like grid search and random search are commonly used [1]. More advanced techniques, such as Bayesian optimization [19], genetic algorithms, Hyperband [20], and population-based training, aim to reduce model evaluation costs and find optimal configurations more effectively [25]. However, these methods often fail to capture the complex relationships between hyperparameters and their impact on model performance.

Pioneering work by Bergstra and Bengio [12] combined grid search with Bayesian optimization, reducing computational costs and enhancing model performance, though it did not address the deeper relationships between hyperparameters. Recent advances, such as asynchronous optimization algorithms [2, 3] and Meta-Learning [26], seek to improve efficiency by adapting based on previous optimization runs. Yet, these techniques still rely on traditional search algorithms and may miss underlying patterns in the hyperparameter space.

Assessing similarity between hyperparameter configurations is crucial. Common methods like Euclidean distance and cosine similarity help compare configurations based on performance. A more powerful tool, Energy Distance, introduced by Szekely *et al.* [22], is effective for comparing data distributions and assessing the similarity between hyperparameter configurations. ED has been shown to reduce computational costs by grouping similar models, making optimization more efficient.

Further work by Rizzo and Szekely [23] demonstrated ED's advantages in optimization and classification tasks. Gortz and Wouters [27] applied ED to optimize hyperparameters by grouping similar models, thus speeding up the search for optimal configurations. Szekely and Rizzo [28] expanded ED's application to non-normal data distributions, particularly useful for high-dimensional spaces in models like Random Forests and deep learning. Jung *et al.* [29] enhanced ED by combining it with other metrics, such as mutual information and correlation distance, for faster convergence during optimization tasks.

While Kim *et al.* [30] demonstrated ED's usefulness in deep learning models, this study focuses on applying ED to Random Forests, which have different hyperparameter dynamics.

Random Forests are a popular machine learning algorithm, and optimizing their hyperparameters is crucial for improving performance [31]. Key hyperparameters include the number of trees (n_estimators), maximum depth (max_depth), and bootstrap sampling (bootstrap) [9]. Traditional optimization methods often fail to capture the relationships between these hyperparameters, leaving room for improvement [8].

Zhou *et al.* [32] combined Bayesian optimization with Random Forests to optimize hyperparameters, showing better efficiency than grid search. However, their approach did not address the interdependencies between hyperparameters. Zhao *et al.* [33] applied deep reinforcement learning for Random Forest optimization, achieving improvements in accuracy and efficiency. This study complements their work by using ED to identify patterns in hyperparameter configurations, offering deeper insights into optimization.

This study builds on previous work by applying Energy Distance to uncover hidden relationships between hyperparameter configurations in Random Forests. While methods like Bayesian optimization and Hyperband focus on improving the search for optimal configurations, they often overlook the nuanced relationships between hyperparameters. By leveraging ED, this study offers a deeper understanding of how hyperparameters interact and affect model performance, contributing to more informed optimization strategies.

III. Hyperparameter Optimization

Hyperparameter optimization is the process of identifying the most effective set of hyperparameters for a machine learning model to enhance its performance [5]. Hyperparameters are external configurations set before training that influence the model's learning process and overall effectiveness. Let $\theta = (\theta_1, \theta_2, \ldots, \theta_n)$ represent the hyperparameters of a machine learning model. The objective is to determine the optimal hyperparameters θ^* that minimize a loss function $L(\theta)$:

$$\theta^* = \arg\min_{\theta} L(\theta) \tag{1}$$

where, $L(\theta)$ is typically evaluated using a validation set or through cross-validation techniques. The challenge in hyperparameter optimization lies in efficiently searching the hyperparameter space to find the best values of θ that minimize the loss.

Grid Search is a straightforward and exhaustive approach for hyperparameter optimization. It involves systematically searching through a predefined hyperparameter space by considering all possible combinations of values [12]. For a hyperparameters set $\theta = (\theta_1, \theta_2, \dots, \theta_n)$, where each θ_i belongs to a discrete discrete set Θ_i, Grid Search creates a grid of all possible combinations:

$$\Theta_{grid} = \prod_{i=1}^{n} \Theta_i \tag{2}$$

For each combination $\theta^* \in \Theta_{grid}$, the model is trained and evaluated using cross-validation:

$$L(\theta^*) = \frac{1}{k}\sum_{i=1}^{k} L_i(\theta^*) \tag{3}$$

where k is the number of folds in cross-validation and $L_i(\theta^*)$ is the loss on the $i-th$ fold. Selection the optimal hyperparameters by identifying the combination θ_{opt} that minimizes the average loss:

$$\theta_{opt} = arg \min_{\theta^* \in \Theta_{grid}} L(\theta^*) \tag{4}$$

The computational burden of Grid Search is influenced by the dimensionality of the hyperparameter space and the number of discrete values per hyperparameter. The total number of evaluations required is:

$$N_{eval} = \prod_{i=1}^{n} |\Theta_i| \times k \tag{5}$$

where $|\Theta_i|$ denotes the cardinality of the set Θ_i. This exponential growth in evaluations with respect to n and $|\Theta_i|$ can render Grid Search computationally prohibitive for models with a large number of hyperparameters or extensive hyperparameter spaces.

Random Search is a simpler method where hyperparameters are selected randomly from predefined distributions over the search space [12]. Instead of exhaustively checking all combinations like Grid Search, Random Search samples combinations of hyperparameters and evaluates the model performance. This can be more computationally efficient for high-dimensional spaces. Given a hyperparameter set $\theta = (\theta_1, \theta_2, \dots, \theta_n)$, where each θ_i is chosen randomly from a distribution, the objective is to minimize the loss function (Eq. (1)).

The number of evaluations in Random Search is defined by the number of iterations $N_{\text{iterations}}$, so:

$$N_{\text{eval}} = N_{\text{iterations}} \tag{6}$$

Random Search is more efficient than Grid Search in high-dimensional spaces but may still miss the optimal configuration due to its random nature.

On the other hand, Bayesian Optimization is an advanced approach that builds a probabilistic model, often using a Gaussian Process (GP), to model the objective function $L(\theta)$ [19]. The goal is to find the optimal hyperparameters θ^* that minimize the loss function, by iteratively refining a surrogate model based on past evaluations. A Gaussian Process is used to model the loss function as:

$$f(\theta) \sim GP\big(m(\theta), k(\theta, \theta')\big) \tag{7}$$

where $m(\theta)$ is the mean function, and $k(\theta, \theta')$ is the kernel function, which defines the covariance between points in the hyperparameter space.

Next, an acquisition function $\alpha(\theta)$ is used to decide the next set of hyperparameters to evaluate. This function strikes a balance between exploring regions of the space that have not been tested and exploiting areas that have already shown good performance. The next hyperparameter configuration θ^* is selected by maximizing the acquisition function:

$$\theta^* = \arg\max_{\theta} \alpha(\theta) \tag{8}$$

After evaluating θ^* and updating the model, the process repeats. The acquisition function helps guide the search more efficiently than Random Search, reducing the number of evaluations required.

Bayesian Optimization is particularly useful for high-dimensional and expensive evaluation functions. It is computationally more intensive than Random Search because it involves training surrogate models (like Gaussian Processes), but it typically converges to the optimal solution in fewer iterations, making it more efficient in many cases.

A. Energy Distance

The Energy Distance is a statistical measure used to quantify the discrepancy between two distributions [23, 24], and it can be applied in hyperparameter optimization to assess the similarity between different configurations of hyperparameters [29, 30]. The approach leverages Euclidean distance to measure dissimilarity between data points in the feature space and provides a method for comparing hyperparameter configurations based on their performance.

The first step in calculating Energy Distance is to compute the pairwise Euclidean distance between two data points x_i and x_j in the feature space [24]. This distance measures how far apart the data points are in the n-dimensional space, where n is the number of features. The Euclidean distance between two points is defined as:

$$d_{ij} = \sqrt{\sum_{k=1}^{n}\big(x_{ik} - x_{jk}\big)^2} \tag{9}$$

where x_{ik} and x_{jk} are the values of the $k-th$ feature of the $i-th$ and $j-th$ data points, and n is the total number of features.

Once the pairwise Euclidean distances are calculated, the Energy Distance between two sets of points (or distributions) can be derived. This is particularly useful when comparing distributions of hyperparameter configurations. The Energy Distance is defined as:

$$ED = \sqrt{2 \times \mu\Delta^2 - \left(\frac{1}{n}\sum_j \mu_{(\Delta^2)j}\right)} \qquad (10)$$

where, μ_{Δ^2} is the mean of the squared differences between data points in the dataset, $\mu_{(\Delta^2)j}$ represents the mean squared difference for each data point jjj within the dataset, $\frac{1}{n}\sum_j \mu_{(\Delta^2)j}$ is the average of the mean squared differences for each of the data points (calculated across all points in the dataset).

The Energy Distance measures the "spread" or "discrepancy" between two sets of data points. In hyperparameter optimization, a lower Energy Distance indicates that the hyperparameter configurations, represented as data points in the feature space, are closer to each other, suggesting similar performance or behavior [30]. On the other hand, a higher Energy Distance implies that the configurations are more dissimilar, with their model performance being significantly different. Therefore, Energy Distance offers a means to assess the similarity between different hyperparameter configurations based on their Euclidean distance in the feature space. Configurations with a smaller Energy Distance are more likely to produce similar outcomes and can be grouped or clustered together for further exploration.

B. *Energy Distance in Hyperparameter Optimization*

In hyperparameter optimization, Energy Distance is a valuable metric for assessing the similarity between different sets of hyperparameters based on their corresponding model performances [29, 30]. The Energy Distance between two hyperparameter configurations, θ_1 and θ_2, can be calculated using the Euclidean distance between their performance metrics in the feature space. When the configurations θ_1 and θ_2 result in similar Energy Distances, it indicates that these two configurations produce comparable performance metrics, making them strong candidates for further fine-tuning or clustering. Mathematically, the Energy Distance between two configurations can be expressed as:

$$ED(\theta_1, \theta_2) = \sqrt{\sum_{k=1}^{n}(L_k(\theta_1) - L_k(\theta_2))^2} \qquad (11)$$

where $L_k(\theta_1)$ and $L_k(\theta_2)$ are the performance metrics (such as loss or accuracy) of configurations θ_1 and θ_2 on the $k-th$ evaluation fold. A smaller Energy Distance suggests similar performance, while a larger one indicates significant differences in model behavior.

IV. Energy Distance-Based Hyperparameter Optimization

A. *Model*

The model for computing the Energy Distance for hyperparameter optimization involves several steps, including data preprocessing, feature extraction, pairwise distance calculation, and Energy Distance computation (Fig. 1). This approach is used to measure the similarity between hyperparameter configurations based on their performance (e.g., accuracy), allowing for optimization and selection of the best-performing configurations.

Let the dataset D consist of n hyperparameter configurations X_i with corresponding performance values y_i, where $i = 1,2,\dots,n$. Each hyperparameter configuration X_i is represented as a vector of hyperparameters $X_i = \left(\theta_1^{(i)}, \theta_2^{(i)}, \dots, \theta_k^{(i)}\right)$, where $\theta_j^{(i)}$ denotes the value of the j − th hyperparameter in the $i-$th configuration. The performance value y_i represents the performance metric (such as accuracy) associated with the hyperparameter configuration X_i.

$$X = \begin{bmatrix} \theta_1^{(1)} & \theta_2^{(1)} & \cdots & \theta_k^{(1)} \\ \theta_1^{(2)} & \theta_2^{(2)} & \cdots & \theta_k^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_1^{(n)} & \theta_2^{(n)} & \cdots & \theta_k^{(n)} \end{bmatrix}$$

The dataset D is loaded, where the hyperparameter configurations X_i and their corresponding performance values y_i, are assigned to their respective columns. In this step, categorical hyperparameters are converted into numerical values. Specifically, each categorical hyperparameter θ_j is mapped to a numerical representation, where $\theta_j \in R^+$for continuous variables or $\theta_j \in Z^+$ for integer-valued parameters. Furthermore, missing values in the dataset are handled by replacing them with -1. If a particular value $\theta_j^{(i)}$ is missing, it is substituted with $\theta_j^{(i)} = -1$, ensuring that the dataset is complete and ready for further analysis.

$$X_D = \begin{bmatrix} \theta_1^{(1)} & \theta_2^{(1)} & \cdots & \theta_k^{(1)} & d_{Energy}^{(1)} \\ \theta_1^{(2)} & \theta_2^{(2)} & \cdots & \theta_k^{(2)} & d_{Energy}^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \theta_1^{(n)} & \theta_2^{(n)} & \cdots & \theta_k^{(n)} & d_{Energy}^{(n)} \end{bmatrix}$$

The matrix representing hyperparameter adjustments and Energy Distance (X_D) is structured as follows: each row corresponds to an iteration i, with the first k columns representing the values of the k hyperparameters at that iteration, $\theta_j^{(i)}$, and the last column containing the Energy Distance $d_{Energy}^{(i)}$, which measures the difference between the predicted and actual distributions after the $i-$th adjustment. This matrix provides a comprehensive view of how hyperparameters evolve over time and how the model's performance improves, as indicated by the decrease in Energy Distance after each adjustment.

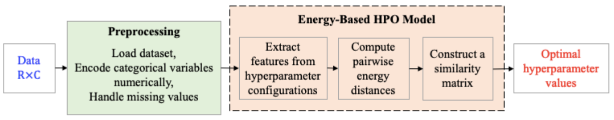
Data
R×C
Preprocessing
Load dataset,
Encode categorical variables numerically,
Handle missing values
Energy-Based HPO Model
Extract features from hyperparameter configurations
Compute pairwise energy distances
Construct a similarity matrix
Optimal hyperparameter values

Fig. 1. Energy-based HPO model.

B. Algorithm

```
Algorithm: Computation of Energy Distance for Hyperparameter Optimization
Input: Dataset D with hyperparameter configurations $X_i$ and
       corresponding accuracy $y_i$.
Output: Energy Distance matrix $E_D$ quantifying similarity
        between hyperparameter configurations.
Begin
  Step 1. Data Preprocessing
    1.1. Load the dataset and assign column names for
         hyperparameters and accuracy.
    1.2. Convert categorical variables into numerical values.
    1.3. Replace NaN (missing) values with −1.
  Step 2. Feature Extraction
    2.1. Extract the hyperparameter values as a feature matrix
         $X = X_1, X_2, \dots, X_k$, where $X_i$ represents the values of
         the $i-th$ hyperparameter configuration.
  Step 3. Compute pairwise Euclidean distances
    3.1. Compute the Euclidean distances between all pairs of
         hyperparameter configurations $(X_i, X_j)$ (Eq. (9))
  Step 4. Compute Energy Distance
    4.1. Compute the squared differences $\Delta_{ij}^2 = d_{ij}^2$ between each
pair of configurations.
    4.2. Calculate the mean squared distance $\mu_{\Delta^2}$.
    4.3. Compute Energy Distance (Eq. (11))
  Step 5. The algorithm returns the Energy Distance matrix $E_D$.
End
```

V. Experiment

A. Data Used

The study uses the MNIST dataset, a benchmark for image classification, consisting of 60,000 training and 10,000 test images of handwritten digits (0–9) with 28×28 pixel resolution. The images are flattened into 1D vectors and processed using a Random Forest Classifier. Preprocessing techniques like PCA were applied to reduce dimensionality and enhance performance. The model's accuracy was optimized through hyperparameter tuning, including parameters such as tree depth, number of estimators, and max features, using grid search with cross-validation to achieve the best configuration for classification (Table I).

B. Tool Used

In this study, several Python libraries and tools were utilized to facilitate data processing, model building, and evaluation. The PyTorch library was employed for deep learning-related tasks, leveraging its powerful tensor computation capabilities. For data manipulation and analysis, Pandas was used to handle structured datasets, while Numpy provided support for numerical computations. Scikit-learn played a central role, with the RandomForestClassifier used for classification tasks and GridSearchCV for hyperparameter tuning to enhance model performance. Additionally, cdist from Scipy was used for computing pairwise distances between data points, assisting in various distance-based algorithms [34].

TABLE I. RandomForest GridSearch Hyperparameter Optimization Results for Digit Classification

No	bootstrap	criterion	max_depth	max_features	min_samples_leaf	min_samples_split	n_estimators	Accuracy
1	TRUE	gini	15	sqrt	1	2	10	0.907
2	TRUE	gini	30	sqrt	1	5	30	0.9265
3	TRUE	entropy	15	sqrt	1	2	10	0.8965
…	…	…	…	…	…	…	…	…
2591	FALSE	log_loss	50	0.5	8	2	10	0.8703

C. Energy Distance Analysis of Hyperparameter Similarity in Random Forest Optimization

The computed Energy Distance matrix, which quantifies the similarity between different hyperparameter configurations, reveals important insights into the performance behavior of the models. The Energy Distance metric, based on the distributional differences between two sets, is used to determine how similar or distinct two hyperparameter configurations are in terms of their predicted outputs.

Configurations that exhibit low Energy Distance values are considered similar, suggesting that they yield comparable performance or predicted results. This can indicate that, despite differences in hyperparameter choices, certain configurations produce outputs with closely aligned distributions. Conversely, configurations with high Energy Distance values are considered distinct, highlighting significant differences in model performance or behavior. These configurations may have diverging predictions, which could help identify combinations of hyperparameters that lead to markedly different model behaviors.

The dataset contains hyperparameter configurations and corresponding performance metrics for a machine learning model, focusing on accuracy and the "Distance to Max Accuracy". Key elements of the analysis include the bootstrap method, which indicates whether bootstrap sampling is used to create data subsets (1 for True, 2 for False); the criterion, which measures the quality of a split (1 for Gini impurity and 2 for entropy); and max depth, which controls the maximum depth of the tree, influencing the model's complexity and potential overfitting. Other factors include max features, specifying the number of features considered for splits; min samples leaf, ensuring a minimum number of samples at a leaf node to prevent overfitting; min samples split, setting the minimum number of samples needed to split an internal node; and n_estimators, representing the number of trees in the forest. Finally, accuracy measures the model's performance on the test set, while the distance to max accuracy calculates the difference between the current configuration's accuracy and the highest accuracy observed.

From the dataset, it’s clear that the accuracy for most configurations hovers around 93.6% to 94%, indicating that the model performs consistently across a wide range of hyperparameter combinations. The highest accuracy of 93.99% appears multiple times, primarily with configurations having a maximum depth of 15, 20, or 30, and various values for the number of features, minimum samples per leaf, and number of estimators.

The column labeled “Distance to Max Accuracy” shows how close each configuration’s accuracy is to the maximum observed accuracy (93.99%). The results with the smallest distances (0) represent configurations that achieved this maximum accuracy, indicating that there is minimal impact from other hyperparameters on the model’s performance at these settings. For example, configurations with: bootstrap = 1, criterion = 1, max_depth = 15, max_features = 1, min_samples_leaf = 1, min_samples_split = 2, n_estimators = 30 all have an accuracy of 93.99% (Table II).

TABLE II. ENERGY DISTANCE MATRIX FOR HYPERPARAMETER SIMILARITY

bootstrap	criterion	max_depth	max_features	min_samples_leaf	min_samples_split	n_estimators	Accuracy (%)	Distance to Max Accuracy
1	1	15	1	1	2	30	93.99	
1	1	15	2	1	2	30	93.99	1
1	1	20	1	1	2	30	93.88	5
1	1	20	2	1	2	30	93.88	5.10
1	3	15	3	8	10	30	88.26	11
2	2	15	3	8	2	20	88.31	12.45
1	3	20	3	8	5	10	87.03	22.16
…	…	…	…	…	…	…	…	…
1	1	50	3	8	10	10	86.25	41.74

The impact of hyperparameters on model accuracy reveals several important insights. The *tree depth (max_depth)* significantly influences accuracy, but its effect tends to stabilize once it reaches a certain value. For instance, tree depths of 15, 20, 30, and even 50 yield similar accuracy values (around 93.6%–93.99%), with only minor variations in the distance to maximum accuracy (Fig. 2). This suggests that the model is relatively resilient to moderate changes in tree depth. Similarly, the *number of features (max_features)* has a relatively minor impact on accuracy, with configurations where max_features equals 1 or 2 performing similarly. However, in some cases, increasing the number of features slightly increases the distance to maximum accuracy, such as with max_features set to 50. The *number of estimators (n_estimators)* also shows diminishing returns beyond 30, with accuracy remaining constant at around 0.936 to 0.9399 even when increased to 50 (Fig. 2). The distance to maximum accuracy only experiences slight increases, indicating that adding more estimators does not provide substantial improvements. Moreover, the model demonstrates robustness to changes in the *sample-related parameters (min_samples_leaf and min_samples_split)*, where variations in values like 1 and 2 have minimal impact on accuracy, with the distances to maximum accuracy remaining within a narrow range.

In terms of *bootstrap* and *criterion*, neither appears to have a significant effect on accuracy. The use of bootstrap sampling (bootstrap = 1) does not notably affect accuracy when compared to configurations without bootstrap sampling (bootstrap = 2), as the overall accuracy remains similar across both settings. Likewise, the choice of *criterion*—whether Gini impurity (criterion = 1) or entropy (criterion = 2)—shows minimal variation in accuracy, suggesting that this parameter does not play a major role in determining the model’s performance under the current configurations.

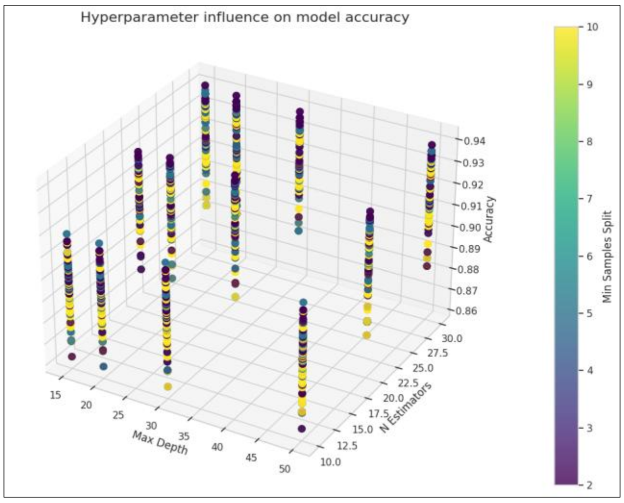

Fig. 2. Hyperparameter influence on model accuracy.

The exploration of optimal configurations demonstrates that the highest accuracy of 93.99% is consistently achieved with the following settings: *bootstrap* = 1 (bootstrap sampling enabled), *criterion* = 1 (using Gini impurity), *max_depth* = 15, *min_samples_split* = 2, and *n_estimators* = 30. For other configurations, the accuracy remains close to this optimal value, with the “Distance to Max Accuracy” typically being low. This suggests that the model performs well across various combinations of hyperparameters. However, when parameters such as *max_depth* or *max_features* deviate from the optimal values, there is a slight decrease in performance, indicating the model’s sensitivity to changes in these settings.

D. Scenario 1: Optimal Hyperparameter Values for Random Search and Bayesian Optimization on MNIST Dataset (HPO1)

In Scenario 1, the optimal hyperparameter values for the Random Forest model on the MNIST dataset were determined through experimental Random Search and Bayesian Optimization. The best configuration included the following values: n_estimators = 30, max_depth = 15 or 20, min_samples_split = 2, min_samples_leaf = 1, criterion = "gini", bootstrap = False, and max_features = "sqrt" or "auto" (Table III). These hyperparameters were selected based on their performance in the experiment, striking a balance between model complexity and generalization.

For Grid Search (GS), the default configuration achieved an accuracy of 0.9083, which was reasonable but not optimal. However, when optimized with the HPO1 hyperparameters, Grid Search significantly improved its performance, reaching an accuracy of 0.9399. This illustrates that, despite being computationally expensive, Grid Search can provide a substantial performance boost when hyperparameters are effectively tuned.

TABLE III. OPTIMAL VALUES OF HYPERPARAMETERS

bootstrap	criterion	max_depth	max_features	min_samples_leaf	min_samples_split	n_estimators
FALSE	gini	15, 20	sqrt, auto	1	2	30

In the case of Random Search (RS), the default configuration yielded an accuracy of 0.8998, slightly lower than that of Grid Search. Nevertheless, after applying the HPO1 hyperparameters, Random Search achieved the highest accuracy of 0.9406, outperforming both the default configuration and Grid Search with optimized hyperparameters. This highlights the advantage of Random Search in exploring the hyperparameter space, especially when combined with a targeted optimization approach.

For Bayesian Optimization (BO), the default configuration achieved an accuracy of 0.9007, which was similar to that of Random Search. With the HPO1 optimize hyperparameters, Bayesian Optimization reached an accuracy of 0.9383, slightly lower than both Random Search (HPO1) and Grid Search (HPO1), but still a significant improvement over the default configuration. This slight difference in performance may result from the way Bayesian Optimization balances exploration and exploitation, which might not always identify the absolute optimal solution in every scenario.

All three optimization methods showed considerable improvements with the HPO1 hyperparameters compared to their default configurations. Random Search (HPO1) achieved the highest accuracy, closely followed by Grid Search (HPO1), while Bayesian Optimization (HPO1) provided a slightly lower accuracy but was still effective (Table IV). This demonstrates the critical role of hyperparameter tuning in improving model performance and efficiency, with Random Search and Bayesian Optimization proving to be more computationally efficient than Grid Search.

TABLE IV. ACCURACY COMPARISON BY OPTIMIZATION METHODS

Methods	GS	RS	BO
Default	0.9083	0.8998	0.9007
HPO1	0.9399	0.9406	0.9383
HPO2	0.9461	0.9471	0.9457

E. Scenario 2: Optimal Hyperparameter for Random Search and Bayesian Optimization on MNIST Dataset (HPO2)

In Scenario 2, the three most influential hyperparameters for the Random Forest model on the MNIST dataset—n_estimators, max_depth, and max_features—were selected for optimization. These hyperparameters were chosen because they significantly impact the model's performance. To further explore their effects, the values of n_estimators and max_depth were increased. Specifically, n_estimators were set to [10, 50, 200], max_depth to [10, 50, 200], and max_features to ['sqrt', 'auto', 'log2']. These configurations were based on prior evaluations indicating that these parameters would likely have the greatest influence on accuracy.

The results clearly showed significant improvements in model performance when the optimized hyperparameters (HPO2) were applied, compared to the default configurations. Specifically, Grid Search (Default) achieved an accuracy of 0.9083, Random Search (Default) reached 0.8998, and Bayesian Optimization (Default) obtained 0.9007. However, after applying HPO2, the performance of all methods improved substantially. Grid Search (HPO2) increased its accuracy to 0.9461, which represents a 3.78% improvement over the default. Similarly, Random Search (HPO2) achieved an accuracy of 0.9471, showing a 4.73% increase, while Bayesian Optimization (HPO2) reached an accuracy of 0.9457, improving by 4.50% over the default configuration (Fig. 3).

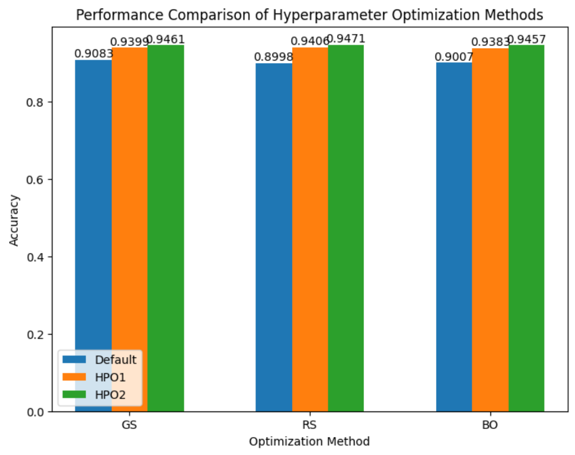

Fig. 3. Performance comparison of hyperparameter optimization method.

In this case, Random Search (HPO2) provided the highest accuracy, outperforming both Grid Search (HPO2) and Bayesian Optimization (HPO2). However, Bayesian Optimization (HPO2) demonstrated its efficiency by achieving almost identical results to

Random Search (HPO2) with fewer computational resources. These results underline the importance of hyperparameter optimization in improving model performance. While Random Search delivered the best accuracy, Bayesian Optimization proved to be a more resource-efficient option while still providing strong performance.

VI. Conclusion

The use of Energy Distance in this study introduces a novel method for analyzing hyperparameter optimization in machine learning models. Similarity between hyperparameter configurations is quantified based on their performance, enabling the identification of clusters with similar configurations. Insights into the most promising hyperparameter sets are provided through this method. Unlike traditional clustering methods like k-means, Energy Distance does not require predefined clusters and can detect subtle patterns in the data, making it more flexible and capable of capturing a broader range of relationships.

Outliers, or hyperparameter configurations that deviate from the expected performance trend, are also identified using Energy Distance. These outliers may indicate overfitting or underfitting, offering opportunities for further model refinement. Compared to traditional methods such as Grid Search or Random Search, Energy Distance provides deeper insights by exploring the relationships between configurations, rather than only focusing on point-wise accuracy. Additionally, Energy Distance considers the distribution of performance metrics, offering a more comprehensive view of model performance.

Despite the advantages, some limitations are acknowledged. The study focused on Random Forests and digit classification, which may limit the generalizability of the findings across other machine learning models or domains. Furthermore, the computational cost of grid search and Energy Distance calculation may hinder scalability. Alternative optimization methods could be explored in future research, and Energy Distance can be tested on additional models and datasets.

Overall, Energy Distance is shown to offer a promising approach for hyperparameter optimization, revealing patterns and relationships often overlooked by traditional methods. Further exploration of this method across various machine learning tasks is encouraged, with future research aiming to refine the technique and expand its application in domains like deep learning, natural language processing, and reinforcement learning.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Thuy Thi Tran conducted the experiments, processed the data, and contributed to drafting the manuscript. Nghia Quoc Phan participated in designing the methodology, tuning the hyperparameters, and analyzing the results. Hiep Xuan Huynh supervised the research, provided critical revisions, and finalized the manuscript. All authors read and approved the final version of the manuscript.

Acknowledgment

The authors thank the supprot from CTU Leading Research Team on Automation, artificial Intelligence, inforMation tEchnology and Digital transformation.

References

[1] P. Gijsbers and M. L. P. Bueno, "AMLB: An AutoML benchmark," *Journal of Machine Learning Research*, vol. 25, no. 6, 2024

[2] M. Baratchi, C. Wang, S. Limmer, J. N. van Rijn, H. Hoos, T. Bäck, and T. Olhofer, "Automated machine learning: Past, present and future," *Artificial Intelligence Review*, vol. 57, no. 5, 2024. https://doi.org/10.1007/s10462-024-10726-1

[3] W. Sun and X. Zhang, "Efficient hyperparameter optimization for deep learning models using evolutionary algorithms," *Journal of Machine Learning Research*, vol. 23, no. 14, pp. 1–23, 2022.

[4] F. Mohr and M. Wever, "Naive automated machine learning," *Machine Learning*, vol. 112, no. 4, pp. 1131–1170, 2023. https://doi.org/10.1007/s10994-022-06200-0

[5] J. Won, H.-S. Lee, and J.-W. Lee, "A review on multi-fidelity hyperparameter optimization in machine learning," *ICT Express*, vol. 11, pp. 245–257, 2025. https://doi.org/10.1016/j.icte.2025.02.001

[6] M. Yang and J. Chen, "Hyperparameter optimization for boosting models using multi-objective optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 9, pp. 3889–3901, 2021.

[7] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges*, Springer, 2019.

[8] L. Perez and D. Wang, "Optimal hyperparameter selection for machine learning applications," in *Proc. 39th Int. Conf. Machine Learning*, 2020, pp. 2423–2431.

[9] D. Angluin and P. Laird, "Critical role of hyperparameters in artificial intelligence and machine learning models," *AI & Society*, vol. 35, no. 1, pp. 79–92, 2020.

[10] L. Liu and D. Zhang, "Enhancing machine learning algorithms with hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 23, pp. 1–26, 2017.

[11] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[12] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[13] J. van der Baan and T. van der Zwan, "The role of hyperparameters in machine learning for predictive analytics in healthcare," *Journal of Healthcare Engineering*, pp. 1–10, 2020.

[14] Z. He and Y. Zhang, "Hyperparameter tuning for financial time series prediction models," *Journal of Financial Data Science*, vol. 7, no. 4, pp. 112–125, 2021.

[15] R. Madani and A. Al-Hmouz, "Impact of hyperparameter optimization in autonomous systems for safety-critical applications," *Journal of Autonomous Systems*, vol. 15, no. 2, pp. 45–60, 2020.

[16] J. Tan and M. Chen, "Hyperparameter optimization and its effectiveness in machine learning models for predictive analytics," *Computers in Industry*, vol. 109, pp. 27–36, 2019

[17] H. Liu and F. Zhang, "Optimizing hyperparameters in machine learning models for critical decision-making in healthcare and finance," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 4, pp. 1445–1457, 2021.

[18] K. Kandasamy, J. Schneider, and S. Vasudevan, "Bayesian optimization with intractable constraints," in *Proc. 35th Int. Conf. Machine Learning*, 2018, vol. 80, pp. 4153–4162.

[19] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, vol. 25, pp. 2951–2959, 2012.

[20] L. Li, K. G. Jamieson, G. DeSalvo *et al*., "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[21] J. Wang, J. Zhang, and X. Li, "Combination of hyperband and bayesian optimization for hyperparameter optimization in deep learning," arXiv preprint, arXiv:1801.01596, 2018.

[22] G. J. Szekely, M. L. Rizzo, and N. K. Bakirov, "Measuring and testing dependence by energy distances," *The Annals of Statistics*, vol. 35, no. 6, pp. 2769–2794, 2007.

[23] M. L. Rizzo and G. J. Szekely, "Energy distance," *Statistics & Probability Letters*, vol. 83, no. 10, pp. 1815–1820, 2013.

[24] R. Jensen, "Energy distance: A new metric for similarity between distributions," *Journal of Machine Learning Research*, vol. 12, pp. 1159–1174, 2011.

[25] M. Jaderberg, W. M. Czarnecki, I. Dunning *et al*., "Population-based training of neural networks," in *Proc. 34th Int. Conf. Machine Learning*, 2017, vol. 70, pp. 1958–1967.

[26] P. Brazdil and C. Giraud-Carrier, "Metalearning and algorithm selection: Progress, state of the art and introduction to the 2018 special issue," *Mach. Learn.*, vol. 107, pp. 1–14, 2018.

[27] S. Gortz and D. Wouters, "Energy distance and its applications to machine learning," *Computational Statistics & Data Analysis*, vol. 72, pp. 189–197, 2014.

[28] G. J. Szekely and M. L. Rizzo, "Energy distance and statistical inference,“ *Statistical Science*, vol. 35, no. 1, pp. 51–70, 2020.

[29] K. Jung, H. Lee, and J. Kim, "Efficient hyperparameter optimization using energy distance combined with mutual information," in *Proc. 41st Int. Conf. Machine Learning*, 2023, pp. 372–380.

[30] Y. Kim, J. Lee, and T. Park, "Energy distance in deep learning: An exploration of hyperparameter optimization," *Journal of Machine Learning*, vol. 15, no. 4, pp. 233–245, 2023.

[31] M. Liao, H. Wen, L. Yang, G. Wang, X. Xiang, and X. Liang, "Improving the model robustness of flood hazard mapping based on hyperparameter optimization of random forest," *Expert Systems with Applications*, vol. 241, 122682, 2024. https://doi.org/10.1016/j.eswa.2023.122682

[32] X. Zhou, L. Zhang, and L. Xie, "Bayesian optimization for hyperparameter tuning of Random Forest models," *Journal of Computational and Graphical Statistics*, vol. 29, no. 4, pp. 931–944, 2020.

[33] B. Zhao, S. Huang, and X. Wang, "Hyperparameter optimization with deep reinforcement learning," in *Proc. 38th Int. Conf. Machine Learning*, 2021, vol. 128, pp. 4183–4192.

[34] F. Pedregosa *et al*., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.