European Journal of Engineering and Applied Sciences

ISSN: 2651-3412 (Print) & 2667-8454 (Online)
Journal homepage: http://dergipark.gov.tr/EJEAS
Published by Çorlu Faculty of Engineering, Tekirdağ Namık Kemal University

European J. Eng. App. Sci. 7(2), 77-83, 2024

Research Article

Comprehensive Analysis of Grid and Randomized Search on Dataset Performance

Nadir Subaşı[1,*]
[1]**Department of Computer Programming, Vocational School of Technical Sciences, Kırklareli University, Kırklareli, Türkiye, 39100**
[1]**nadir.subasi@klu.edu.tr**

Geliş: 08.11.2024 Kabul: 28.11.2024 DOI: 10.55581/ejeas.1581494

Abstract: This paper presents a comprehensive comparison of grid search and randomized search, the two main hyperparameter search methods used in machine learning. The paper analyses the performance of these two methods in terms of efficiency, scalability and applicability on different machine learning models and datasets. In the paper, it is emphasized that grid search provides a comprehensive search since it searches all hyperparameter combinations on a regular grid, but it creates high computational cost. On the other hand, while random search provides faster results by selecting random samples from the hyperparameter space, it has the disadvantage of not providing complete coverage. Practical suggestions and decision-making processes are also presented for which search method should be preferred in real-world applications. In conclusion, the paper summarizes the situations where grid search and random search can be advantageous according to factors such as the complexity of the model, the size of the hyperparameter space and the available computational resources and aims to provide a comprehensive guide for practitioners.

Keywords: *Dataset, Grid Search, Hyperparameter Optimization, Machine Learning, Model Performance, Random Search.*

Veri Kümesi Performansı Üzerinde Izgara ve Rastgele Aramanın Kapsamlı Analizi

Öz. Bu makale, makine öğreniminde kullanılan iki ana hiperparametre arama yöntemi olan ızgara arama ve rastgele arama yöntemlerinin kapsamlı bir karşılaştırmasını sunmaktadır. Makale, bu iki yöntemin performansını verimlilik, ölçeklenebilirlik ve farklı makine öğrenimi modelleri ve veri kümeleri üzerinde uygulanabilirlik açısından analiz etmektedir. Makalede, ızgara aramanın düzenli bir ızgara üzerinde tüm hiperparametre kombinasyonlarını aradığı için kapsamlı bir arama sağladığı, ancak yüksek hesaplama maliyeti yarattığı vurgulanmaktadır. Öte yandan, rastgele arama hiperparametre uzayından rastgele örnekler seçerek daha hızlı sonuçlar sağlarken, tam kapsam sağlamama dezavantajına sahiptir. Gerçek dünya uygulamalarında hangi arama yönteminin tercih edilmesi gerektiğine dair pratik öneriler ve karar verme süreçleri de sunulmuştur. Sonuç olarak makale, modelin karmaşıklığı, hiperparametre uzayının büyüklüğü ve mevcut hesaplama kaynakları gibi faktörlere göre grid arama ve rastgele aramanın avantajlı olabileceği durumları özetlemekte ve uygulayıcılar için kapsamlı bir rehber sunmayı amaçlamaktadır.

Anahtar kelimeler: *Veri Kümesi, Izgara Arama, Hiperparametre Optimizasyonu, Makine Öğrenmesi, Model Performansı, Rastgele Arama,.*

1. Introduction

Hyperparameter optimization plays a critical role in the performance of machine learning models by improving the accuracy and generalization capability of algorithms. Correct tuning of hyperparameters such as learning rate and regularization power allows the model to perform optimally on unseen data. The most common methods for this optimization

∗Corresponding author
E-mail address: nadir.subasi@klu.edu.tr (N. Subaşı)

are grid search and random search, which follow different strategies [1].

Grid search guarantees the best result by evaluating all possible combinations of hyperparameters; however, the computational cost increases rapidly as the number of

hyperparameters increases. This method has been a fundamental tool for many years, especially in models such as support vector machines and decision trees with limited hyperparameters but has become less effective in complex models such as deep neural networks [2], [3].

In contrast, random search takes a more probabilistic approach by randomly sampling hyperparameter values from predefined ranges. Instead of evaluating every possible combination, randomized search randomly selects a subset of hyperparameters to be evaluated. Although it may seem counterintuitive, Bergstra and Bengio showed that random search often outperforms grid search in high-dimensional spaces, especially when only a few hyperparameters are important [4]. This is because randomized search avoids wasting computational effort exploring irrelevant parts of the hyperparameter space.

Random search, developed by Bergstra and Bengio in 2012, aims to explore a large space at a lower cost by taking random samples from the hyperparameter space. This method has emerged as a more advantageous option when working with large hyperparameter spaces or limited resources and has become widely used in machine learning libraries such as Scikit-learn.

This paper aims to present a comprehensive comparison of grid search and randomized search in terms of efficiency, scalability and applicability in machine learning models. We will examine the advantages and disadvantages of the methods, evaluate their performance on benchmark datasets and discuss the reasons for their preference over real-world applications. At the end of the paper, clear guidelines on which method is more appropriate depending on model complexity, size of the hyperparameter space and computational resources will be presented.

1.1. Hyperparameter Optimization

Hyperparameter optimization is a critical process to improve model performance and avoid under- or over-fitting problems. Grid search, a widely used method, has high computational cost while systematically evaluating all possible combinations of hyperparameters. In contrast, random search offers a faster and more efficient alternative by randomly sampling hyperparameters in the search space, but at the risk of missing optimal regions. Understanding the advantages and disadvantages of these methods is a fundamental requirement for choosing the optimization technique that suits the requirements of the model.

1.1.1. Grid Search

Grid search is a traditional method in hyperparameter optimization as it systematically evaluates all possible combinations. For example, for three hyperparameters with three potential values (learning rate, chunk size and regularization power), a total of 27 combinations are tested and the best performing combination is guaranteed to be found.

This method is effective in low-dimensional spaces, but as the number of hyperparameters increases, the combinations increase exponentially, and the computational cost rises rapidly. This is known as the curse of dimensionality and makes grid search inefficient, especially for complex models. Furthermore, in high-dimensional spaces where only a few hyperparameters contribute significantly to performance, grid search often wastes computational resources by working on unnecessary combinations.

1.1.2. Random Search

Bergstra and Bengio proposed randomized search to overcome the limitations of grid search. Random search works faster and more efficiently in high-dimensional spaces by sampling hyperparameters from specific distributions. Unlike grid search, instead of giving the same importance to every hyperparameter, it enables faster selection of important parameters. This is particularly advantageous when performance depends on several hyperparameters.

Mathematically, the random search complexity is limited to a certain number of iterations (N), which can be adjusted according to resources or time. This flexibility can provide good results even with a low number of iterations. However, randomized search may not discover specific areas, and since the results depend on the sampling distributions used, important areas may be missed if the distributions are inadequate.

2. Summary of Real-World Applications and Case Studies

The theoretical comparisons between grid search and random search can be better understood through the performance of these hyperparameter optimization techniques on real-world problems. Case studies in areas such as deep learning and natural language processing (NLP) demonstrate the effectiveness of these methods [1].

In deep learning, especially in complex models such as Convolutional Neural Networks (CNN), random search offers a significant advantage. In a study on the CIFAR-10 dataset, grid search evaluated 729 combinations, while random search achieved similar accuracy with only 100 combinations. Random search also halved the optimization time [5]. In an optimization study for transducer models in NLP, random search achieved 92.5% accuracy with 100 combinations, while grid search achieved 92.7% accuracy with 500 combinations. Randomized search saves time by providing similar results even though fewer configurations are evaluated [6]. Automated Machine Learning (AutoML) systems favor random search to deal with large hyperparameter spaces. In the H2O.ai AutoML framework, optimization with random search resulted in faster training times and better accuracy results. [7]. In the financial sector, training times were reduced by 40% using random search for fraud detection. This emphasises the ability of randomized search for rapid tuning and large-scale deployment in high-risk environments [8]. Studies show that random search is more efficient in high-dimensional spaces and should be preferred especially in areas such as deep learning and NLP. Grid search can be useful in small-scale

tasks, but random search is generally a better choice for more complex and large data sets.

2.1. Challenges and Limitations

Although grid search and random search are effective methods for hyperparameter optimization, they both have various challenges and limitations. Understanding these limitations is critical to choosing the right optimization technique.

Challenges of Grid Search

- **Computational Cost:** As the hyperparameter space expands, grid search becomes computationally inefficient. For example, in an optimization with three hyperparameters, the number of possible combinations increases exponentially and becomes difficult to manage in large models.
- **The Curse of Dimensionality:** As the number of hyperparameters increases, the number of combinations also increases exponentially, which reduces the chance of reaching the optimal combination. Grid search often wastes time and resources by evaluating sub-optimal points.
- **Lack of Flexibility:** Since a fixed hyperparameter values work on a grid search, there is a risk of missing the optimum values. This is especially problematic for hyperparameters that take continuous values.

Challenges of Randomized Search

- **Stochastic Structure:** Randomized search shows variability as the results depend on the random seed. Without enough iterations, sub-optimal results can be obtained.
- **Risk of Inadequate Exploration:** Under-exploration may occur in lower dimensional spaces. If very few iterations are performed, better results can be obtained due to the exhaustive nature of grid search.
- **Dependence on Hyperparameter Distributions:** Randomized search is dependent on predefined distributions. Poorly chosen intervals can lead to missing important hyperparameter values.

Limitations of Both Methods

- **Lack of Adaptability:** Both grid and random search are static methods. New sets of hyperparameters are determined independently of previous results, which can lead to wasted computations.
- **Scalability Issues in Large Models:** In large models such as deep neural networks with millions of parameters, both methods struggle to find the optimal hyperparameters.

As a result, grid search is computationally expensive and ineffective in high-dimensional spaces, while randomized search offers a more efficient alternative but carries the risk of under-exploration. Both methods lack adaptability and may have difficulty in dealing with large, complex models.

3. Material and Method

This section describes the methodology used to compare grid search and random search for hyperparameter optimization. The aim is to evaluate the efficiency, scalability and accuracy of these methods on different machine learning models and datasets.

1. **Experimental Setup:** The objective of the experiment was to evaluate the efficacy of grid search and random search techniques using the most commonly employed datasets (MNIST (5000x784), Iris (150x4)) and machine learning models, comprising support vector machines (SVM), neural networks (NN), and random forest (RF). Among the datasets used in the study, the MNIST dataset contains 5000 samples, each with 784 feature vectors of 28x28 pixels. On the other hand, the IRIS dataset consists of 150 samples and has 4 features for each sample [9][10].
2. **Hyperparameter Space:** The impact of hyperparameters on model performance should be assessed in terms of accuracy, generalizability and computational cost. For example, while higher parameter numbers generally lead to better performance, they can also increase computational cost and risk over-learning. For each model, an attempt is made to balance the number of parameters. Each model has its own hyperparameters and search space. For SVM, parameters such as gamma, regularization parameter C, kernel type and kernel coefficient were determined. For NN, the hidden layer size, activation function, L2 regularization term (Alpha), learning rate parameters are selected. For Random Forest (RF), the learning coefficient, number of trees and depth parameters are selected.
3. **Implementation:** Both methods are implemented with the GridSearchCV and RandomisedSearchCV functions of the Scikit-learn library. Performance is evaluated by k- fold cross-validation and compared on metrics such as accuracy and computational cost.
4. **Evaluation Criteria:** Performance is evaluated based on training time as a and calculation cost and verification accuracy.
5. **Scalability Test:** The scalability of all methods was tested with hyperparameter spaces of different dimensions.
6. **Stopping Criteria:** For random search, stopping criteria were determined using a certain number of iterations and performance threshold.

The performance and practical applications of this methodology between grid search and random search are extensively analyzed.

4. Main Results and Performance Indicators

4.1. Performance Comparison

The performance of grid search and random search in hyperparameter optimization is evaluated based on key factors such as efficiency, scalability and effectiveness in finding optimal hyperparameters. In this section, we present the results of both methods using different machine learning models and datasets. We will focus on training time, validation accuracy (or model performance) and computational cost as metrics. It is shown in Table 1.

Table 1 Comparison of performance and computational cost of models

Dataset	Model	Search Algorithms	Accuracy (%)	Duration (sec)
MNIST	SVM	GRID SEARCH	92.440%	3349.189
		RANDOM SEARCH	90.500%	2581.997
	NN	GRID SEARCH	92.400%	1395.008
		RANDOM SEARCH	93.080%	776.624
	RF	GRID SEARCH	93.660%	2149.913
		RANDOM SEARCH	**93.460%**	**516.123**
IRIS	**SVM**	GRID SEARCH	97.333%	5.810
		RANDOM SEARCH	**97.333%**	**0.648**
	NN	GRID SEARCH	97.333%	54.176
		RANDOM SEARCH	96.667%	17.266
	RF	GRID SEARCH	96.667%	164.022
		RANDOM SEARCH	96.667%	42.972

Evaluations on the MNIST dataset show that the RF algorithm is ahead in terms of accuracy. However, the RandomSearch method stands out by achieving a 4.16-fold reduction in computational cost for only a 0.2% reduction in accuracy. In the IRIS dataset, SVM achieves a high accuracy rate due to the low number of features, whereas the RandomSearch method achieves the same accuracy rate but reduces the computational cost by 9.5 times in the hyperparameter selection process.

Table 1 emphasizes that various algorithms should be preferred for different datasets. However, when the hyperparameter selection process is considered, the RandomSearch method stands out as a viable option that offers a significant computational cost advantage with an acceptable loss of accuracy.

4.2. Efficiency and Computational Cost in Hyperparameter Space

While grid search evaluates each hyperparameter combination in a systematic way, random search explores the hyperparameter space more efficiently by randomly sampling these values. For example, in our experiment using Support Vector Machine (SVM), the SVM hyperparameter pool offers 32 different options. With the cross-validation value defined as 5, the grid search evaluated 160 models, while the random search worked with only 100 combinations.

Although both methods achieved similar validation accuracy, the random search completed the optimization process in less time than the grid search. The other model parameter pools are detailed in Table 2.

Table 2 Model Parameters Pool

	SVM	NN	RF
PARAMETERS	32	48	108
CROSS-VALIDATION	5	5	5
GRID SEARCH CANDIDATES	160	240	540
RANDOM SEARCH CANDIDATES	100	100	100
CANDIDATE DROPOUT RATE %	62,5	41,6	18,5

Grid search leads to high computational costs due to its exhaustive nature, as it evaluates all possible combinations. For example, a random forest model with three hyperparameters may need to evaluate hundreds or thousands of configurations. Randomized search can work with fewer combinations and perform fewer evaluations, making it a more time and resource efficient option. In the experiments on the MNIST dataset, random search is limited to 100 combinations and evaluated, while grid search works with 540 combinations by trying all possibilities. This is presented in Table 2.

As can be seen from Table 1, in terms of computational cost (time), random search is only inferior to the random forest model, and the success rate is only 0.2% lower against this requirement, as mentioned in the previous table.

4.3. Practical Considerations

Practitioners should consider the balance of computational cost and efficiency when choosing between grid search and randomized search. Grid search is generally more suitable for small hyperparameter spaces, but random search is a better choice in high dimensional spaces. Especially for scenarios where time and resources are limited, random search offers a more practical approach for complex models such as deep learning.

As a result, random search consistently outperforms grid search in terms of computational efficiency and scalability, while providing similar or near-optimal results in most cases. While the exhaustive approach of grid search may not be suitable for high-dimensional hyperparameter spaces, random search offers a flexible and effective alternative for many machine learning tasks.

For the evaluation, the MNIST (5000x784) dataset, which is frequently used in machine learning studies, was used. At this stage, the dataset was hyperparameter optimized with SVM, NN and RF algorithms using grid search and random search, respectively, and the graphs of success rates and computational costs (time) are presented in Figure 1, Figure 2 and Figure 3.

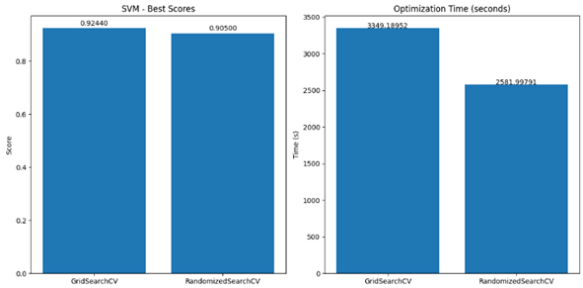

Figure 1. *SVM performance for MNIST Dataset*

Figure 1 shows the accuracy and computational cost of the SVM algorithm for the MNIST dataset. GridSearch calculates it in 3349 seconds with 92.44% accuracy. The same dataset and algorithm with RandomSearch can compute in 2581 seconds with 90.5% accuracy. Thus, the 1.94% accuracy decrease was realized 23% more rapidly.

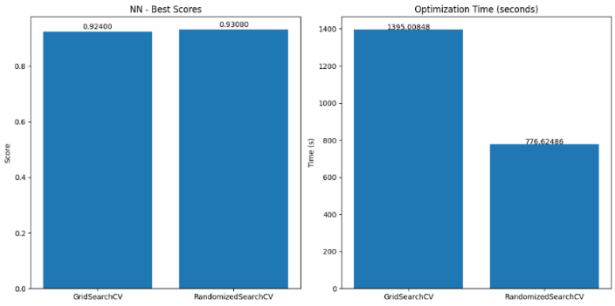

Figure 2. *NN performance for MNIST Dataset*

Figure 2 shows the accuracy and computational cost of the NN algorithm for the MNIST dataset. GridSearch calculates it in 1395 seconds with 92,4% accuracy. The same dataset and algorithm with RandomSearch can compute in 776 seconds with 93% accuracy. In this calculation, Random Search achieved both better accuracy (0.6%) and 44% quicker.

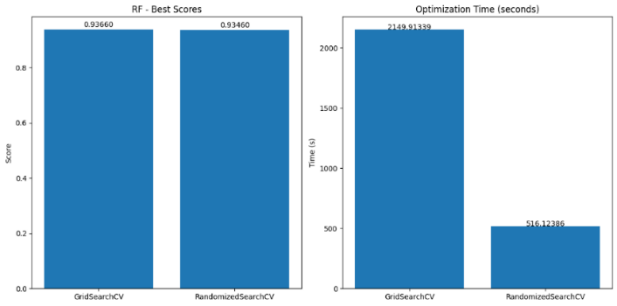

Figure 3. *RF performance for MNIST Dataset*

Figure 3 shows the accuracy and computational cost of the RF algorithm for the MNIST dataset. GridSearch calculates it in 20149 seconds with 93,6% accuracy. The same dataset and algorithm with RandomSearch can compute in 516 seconds with 93,4% accuracy. Thus, the 0,2% accuracy decrease was realized 76% more rapidly.

Figure 4 shows a box plot of the cross-validation values for the performance comparison of these algorithms. This box plot compares the cross-validation scores of SVM, NN and RF models. SVM models exhibit a wider distribution of performance compared to the other models. In some cases, very low scores (outliers around 0.2) were obtained, indicating that SVM may be inadequate in certain scenarios. The NN models are quite consistent in terms of performance, with scores generally concentrated between 0.8 and 0.9. This shows that NN has a stable performance. RF models, on the other hand, are as consistent and high performing as NN, but with slightly less variance. Overall, NN and RF models show similarly high performance, but RF may be a step ahead in terms of stability. On the other hand, SVM performs poorly compared to the other models with its low median value and high variance.

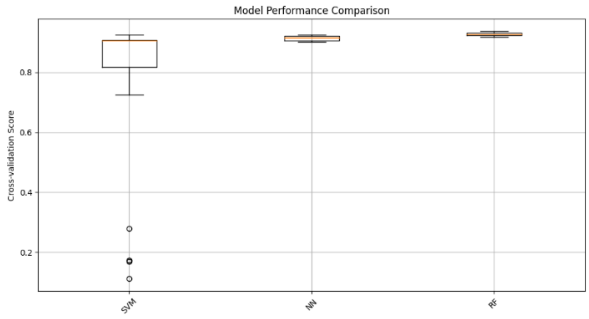

Figure 3. *Performance comparison of Algorithms for MNIST Dataset*

While continuing the evaluation, another dataset frequently used in machine learning studies, IRIS (150x4) dataset was also tested.

At this stage, the dataset was hyperparameter optimized with SVM, NN and RF algorithms using grid search and random search respectively and the graphs of success rates and computational costs (time) are presented in Figure 5, Figure 6 and Figure 7.

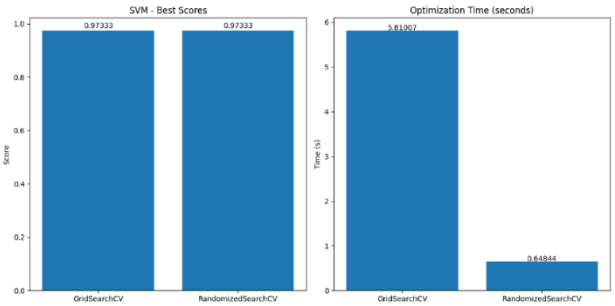

Figure 4. *SVM performance for IRIS Dataset*

Figure 5 shows the accuracy and computational cost of the SVM algorithm for the IRIS dataset. GridSearch calculates it in 5,8 seconds with 97,33% accuracy. The same dataset and algorithm with RandomSearch can compute in 0,64 seconds with the same accuracy. Even though the accuracy is the same, it is 89% faster in terms of time.

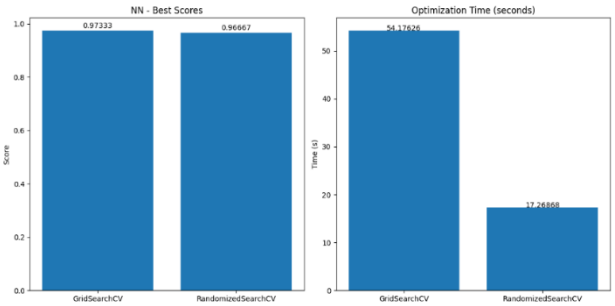

Figure 5. *NN performance for IRIS Dataset*

Figure 6 shows the accuracy and computational cost of the NN algorithm for the IRIS dataset. GridSearch calculates it in 54,17 seconds with 97,33% accuracy. The same dataset and algorithm with RandomSearch can compute in 17,26 seconds with 96,66% accuracy. In this calculation, Thus, the 0,67% accuracy decrease was realized 68% more rapidly.

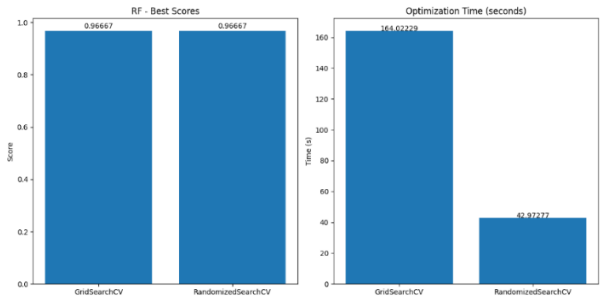

Figure 6. *RF performance for IRIS Dataset*

Figure 7 shows the accuracy and computational cost of the RF algorithm for the IRIS dataset. GridSearch calculates it in 20149 seconds with 93,6% accuracy. The same dataset and algorithm with RandomSearch can compute in 516 seconds with 93,4% accuracy. Even though the accuracy is the same, it is 74% faster in terms of time.

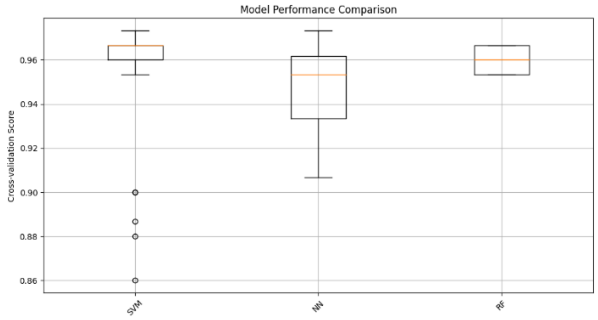

Figure 7. *Performance comparison of Algorithms for IRIS Dataset*

Figure 8 shows a box plot of the cross-validation values for the performance comparison of these algorithms. Box plot compares the cross-validation scores of SVM, NN and RF models. SVM models exhibit a wider distribution of performance compared to the other models. In some cases, very low scores (outliers around 0.2) were obtained, indicating that SVM may be inadequate in certain scenarios. The NN models are quite consistent in terms of performance, with scores generally concentrated between 0.8 and 0.9. This shows that NN has a stable performance. RF models, on the other hand, are as consistent and high performing as NN, but with slightly less variance. Overall, NN and RF models show similarly high performance, but RF may be a step ahead in terms of stability. On the other hand, SVM performs poorly compared to the other models with its low median value and high variance.

5. Future Directions and Improvements

As the complexity of machine learning models increases and datasets grow, more sophisticated techniques for hyperparameter optimization are needed. In this chapter, advanced approaches such as Bayesian optimization, adaptive search techniques, automated machine learning (AutoML) systems and hybrid methods will be reviewed and their potential on hyperparameter tuning efficiency will be discussed.

Bayesian optimization is a method that guides hyperparameter search using past performance information. Builds a surrogate model (usually a Gaussian process) and establishes a balance between exploration and exploitation [11].

- **Efficiency:** It provides a more efficient search by focusing on the regions of the hyperparameter space containing the best solution.
- **Adaptability:** It becomes adaptive with further evaluation.
- **Scalability:** Works well for models with less than 20 hyperparameters, but complexity increases as the number increases.

Adaptive randomized search techniques become more effective by dynamically adjusting the search process based on previous results. Methods such as Hyperband terminate low-performing configurations early, directing resources to more promising configurations. This is particularly efficient for large hyperparameter spaces.

Evolutionary algorithms, such as genetic algorithms, optimize hyperparameter configurations by simulating the process of natural selection. In each generation the best configurations are selected and improved over time. This method offers the ability to effectively explore large and complex spaces [12].

AutoML systems automate processes such as model selection and hyperparameter optimization. By integrating advanced search techniques, it becomes useful for non-experts with user-friendly interfaces. It also provides more efficient optimization by automatically adapting to the characteristics of data sets.

Hybrid methods combine the strengths of more than one optimization technique. For example, they can start with a broad random search and continue with Bayesian optimization around promising regions. These methods strike a balance between exploration and exploitation, resulting in more robust results.

6. Conclusion

The development of advanced hyperparameter optimization techniques such as Bayesian optimization, adaptive search methods and evolutionary algorithms reflects the growing need for more efficient and scalable solutions in machine learning. While grid search and randomized search continue to be widely used, these new methods offer significant improvements in computational efficiency, adaptability and scalability. As machine learning continues to evolve, hybrid approaches and AutoML systems will likely play an increasingly important role in simplifying and automating the hyperparameter optimization process. In this paper, we show that when random search is compared to grid search, random search offers a high gain in computational cost over grid search, with a modest performance degradation when using the same hyperparameter spaces, but a high computational cost gain.

Author Contribution

Formal analysis – Nadir Subaşı (NS); Investigation – NS; Experimental Performance – NS; Collection – NS; Processing – NS; Literature review – NS; Writing – NS; Review and editing – NS.

Declaration of Competing Interest

The authors declared no conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

[1] Mekonnen, T. (2019). Random vs. Directed Search for Scarce Resources. *International Journal of Advanced Computer Science and Applications,*269-278.

[2] Lawrence, J. P., & Steiglitz, K. (1972). Randomized Pattern Search. *IEEE Transactions on Computers*, 21(4), 382–385.

[3] Vincent, P., & Rubin, I. (2004). Cooperative search versus random search using UAV swarms. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 37(8), 944–949.

[4] Bergstra, J., Ca, J. B., & Ca, Y. B. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13, 281–305.

[5] Aszemi, N. M., & Dominic, P. D. D. (2019). Hyperparameter Optimization in Convolutional Neural Network using Genetic Algorithms. *International Journal of Advanced Computer Science and Applications,* 10(6).

[6] Sudhakaran, P., & Baitalik, S. (2022). XGBoost Optimized by Adaptive Tree Parzen Estimators for Credit Risk Analysis. *2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon),* (pp. 1-6). *IEEE.*

[7] Japa, L., Serqueira, M., Mendonca, I., Aritsugi, M., Bezerra, E., & Gonzalez, P. H. (2023). A Population-Based Hybrid Approach for Hyperparameter Optimization of Neural Networks. *IEEE Access*, 11, 50752–50768.

[8] Zhao, Z., & Bai, T. (2022). Financial Fraud Detection and Prediction in Listed Companies Using SMOTE and Machine Learning Algorithms. *Entropy*, 24(8), 1157.

[9] Y Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

[10] Unwin, A., & Kleinman, K. (2021). The Iris Data Set: In Search of the Source of Virginica. *Significance*, 18(6), 26–29.

[11] Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, *25*.

[12] Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, *10*(2), 99-127