BootBOGS: Hands-on optimizing Grid Search in hyperparameter tuning of MLP

Genane Youness, Nu Uyen Thi Phan, Benjamin Cohen Boulakia

BootBOGS: Hands-on optimizing Grid Search in hyperparameter tuning of MLP

Genane Youness [1,2], Nu Uyen Thi Phan [1,3] and Benjamin Cohen Boulakia [1,4]
[1]Laboratoire LINEACT CESI, IDFC, 92000 Nanterre, France; https://orcid.org/0009-0003-9038-3489
[2]Laboratoire Cedric-MSDMA, 75003 Paris, France
[3]Université Paris Cité, 75006 Paris, France; nu-uyen-thi.phan@etu.u-paris.fr
[4]EFREILab, EFREI, IDFC, 94800, Villejuif, France

***Abstract*—Neural networks are widely used in the literature in a variety of fields and for a large number of applications. A major challenge in their use is the need to identify and process hyperparametric values. Grid Search is a widely used technique for meeting this task. It systematically searches for values in a predefined range of hyperparameters. However, selecting the appropriate range of hyperparameters can be difficult, as the search space can be vast, resulting in an extensive number of combinations to be tested. It is more suited to short, fast searches for hyperparameter values, within ranges that are known to be generally efficient. In this paper, we present an improvement to Grid Search using BootBOGS, a bootstrap-based approach to hyperparameter optimization. BootBOGS is a hybrid approach that combines bootstrap and Bayesian Search with the Grid Search technique to perform an efficient search in hyperparameter space. Bayesian Search is used to initialize hyperparameter ranges. Bootstrap is used to explore the distribution of model performance for each hyperparameter combination and to reduce its variance, enabling us to better understand the margins of these hyperparameters and to reduce these ranges. Grid Search is then used to refine the selection of hyperparameters. To evaluate the effectiveness of the proposed approach, a set of computational experiments are carried out on four different datasets from classification problems, for which we compared BootBOGS to several other strategies: Grid Search, Random Search, and Bayesian Optimization. The results show that our method is able to find better hyperparameter configurations in terms of predictive quality with a reasonable runtime and lead to more robust and reliable hyperparameter tuning processes.**

***Index Terms*—Grid Search, Bayesian Search, hyperparameter tuning, MLP, Bootstrap.**

I. Introduction

In many areas of machine learning, neural networks and more specifically multi-layer perceptron neural networks are proving effective in solving the problem of classification. MLP classifiers require the use of hyperparameters that must be defined beforehand. The setting of these hyperparameters is a crucial phase in their development. It significantly influences a model's performance and its ability to generalize to new data. Among these techniques, Grid Search has always played a predominant role (Figure 1) and is widely used. However, it does have some limitations. As models become more complex and datasets increase in size, the limits of traditional Grid Search have become obvious. Despite its conceptual simplicity, Grid Search struggles with the challenges posed by intricate models operating in high-dimensional parameter spaces. The manual specification of hyperparameters and the lack of adaptability hinder its efficiency in finding optimal configurations.

To address this issue, the main contribution of this work is to propose a novel hybrid method called BootBOGS. BootBOGS combines bootstrapping on hyperparameters with Bayesian optimization and Grid search. On the basis of empirical performance models, BootBOGS is evaluated alongside established and widely used strategies, including Grid search, Random search, and Bayesian optimization, across four datasets. This evaluation encompasses various metrics, such as accuracy, AUC, G-mean, and F2. This comprehensive assessment reveals both the strengths and limitations of BootBOGS, offering valuable insights to inform future advancements and explorations in the field. Furthermore, we have made the code for this experiment open-source, with the aim of contributing to future developments.

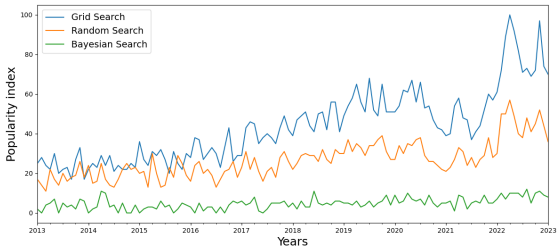


Fig. 1: Popularity index over years [1]

II. Context and Related Work

Hyperparameter tuning for neural networks is a key area of research aimed at optimizing the performance and improving the efficiency of neural network models. Research continues to develop in this area, presenting innovative methods and tools to meet this challenge. (1) presents Random Search as a simple and more efficient approach in hyperparameter optimization, compared to the Grid Search experiments of (2). (3) have developed the Hyperband algorithm, which uses a combination of random sampling and successive halving to efficiently allocate resources to different hyperparameter configurations.

[1]Source: Google Trends for Three Hyperparameter Tuning Methods in Computer Science Worldwide from 2013 to 2023 [accessed 4 Sep, 2023]

(4) combines Hyperband with Bayesian Optimization in a multi-fidelity optimization method BHBO. Predictive hyperparameter optimization is used by (5) to predict the final performance of a machine learning system based on the early termination of poor performance. (6) presents an approach based on Bayesian Optimization with Gaussian Processes for selecting MLP model hyperparameters for state estimation purposes.

A. Problem of search space

The parameters are the weights of the connections in the neural network that are learned during the training stage. Hyperparameters are external configurations that cannot be estimated from data, and which are set by trainers and refined by tuning ((7)). They must be carefully optimized to obtain the best possible performance. The choice can be made on the basis of previously published recommendations in the literature for the same data set, expertise or trial-and-error that can guide the selection of hyperparameters, particularly with limited training data.

Recent advancements in hyperparameter tuning have spurred the development of advanced strategies that enhance Grid Search techniques. To perform a search in the hyperparameters space, Grid Search builds a model for each combination of hyperparameter values and evaluates each model. However, it makes no assumptions about the search space; the user must specify a fixed range of values to search in. This selection plays a key role in the quality of the resulting model, as well as in the efficiency and convergence speed of the development process. The problem arises when dealing with hyperparameters which domains are continuous, leading to exponential growth in the combinations of possible values. The vast multidimensional space resulting from combinations of many hyperparameters presents an even greater challenge. The challenge is to select the right ranges for each hyperparameter. The most used optimization methods, namely Grid Search, Random Search, and Bayesian Search are described below.

B. Grid Search

Grid Search (GS) is a commonly used hyperparameter optimization technique that involves testing different combinations of hyperparameters in a cartesian way to determine the best one for a given problem. All combinations have the same probability of impact on the process. The range for each hyperparameter is generally determined by the developer, along with a step for moving from one hyperparameter value to another. The number of hyperparameter combinations grows exponentially with the search space ((1)), resulting in heavy computational costs. In addition, Grid Search may exploit many insignificant hyperparameter areas or fail to find the best hyperparameters if it is not included in the selected search space.

C. Random Search

Random search (RS) (8) is a hyperparameter tuning technique that involves randomly sampling hyperparameters from a defined search space and evaluating the algorithm's performance. It is a simple and efficient method of hyperparameter tuning (9) allowing a large hyperparameter search space to be explored with fewer iterations than Grid Search. Compared to Grid Search, Random search is particularly useful when the search space is large and the relationship between hyperparameters and model performance is complex (1).

D. Bayesian Search

Bayesian Optimization (BO) is an algorithm that effectively optimizes objective functions with high evaluation costs based on Bayes' Theorem ((10), (11)). BO significantly reduces the number of trials (12) and errors when tuning hyperparameters compared to Grid Search or Random Search (13). The key idea behind Bayesian optimization is to build a probabilistic model of the objective function (14), which is typically the validation accuracy or loss of the model. This probabilistic model is updated as new evaluations of the objective function are made, allowing the algorithm to intelligently choose the next set of hyperparameters on the basis of prior samples ((15)). (16) introduce a model-based sequential Bayesian optimization approach, called Tree-structured Parzen Estimator (TPE), It uses a tree-structured search process to iteratively split the space of hyperparameters into regions with a high and low probability of containing good hyperparameters and then evaluates the objective function at the best hyperparameters within the high-probability regions.

E. Overview

Table I provides an overview of the advantages and disadvantages of these different optimization methods.Each approach has merits that meet specific characteristics of the problem.

TABLE I: Advantages and disadvantages of optimization method.

Method	Advantages	Disadvantages
Grid Search	Guaranteed to find the best combination if it exists within the selected search space. Simple and transparent.	Computationally expensive, especially with large search spaces. Inefficient when hyperparameters are not equally important. Limited to predefined hyperparameter values.
Random Search	Efficient in terms of computation time. Better exploration of the search space compared to Grid Search.	No guarantee of finding the best combination. Randomness may lead to suboptimal solutions in some cases
Bayesian Search	Efficient and intelligent exploration of the search space. Typically requires fewer iterations to find good hyperparameters.	More complex to implement than Grid and Random Search. May require tuning additional parameters for the probabilistic model.

In short, Random Search presents itself as a quick and direct approach to explore hyperparametric spaces, but with randomness in optimization. Grid Search offers full coverage but can extend calculation time. Bayesian Optimization establishes a trade-off by using probabilistic models to strategically

maneuver the research space and adjust its approach based on historical assessments. The choice of the most appropriate technique depends on factors such as the complexity of the problem, the available computing resources, and the delicate balance between exploration and exploitation.

III. Methodology

The motivation behind developing a new hyperparameter optimization method is rooted in the limitations of existing methods. This novel approach aims to combine their strengths, and thus optimize classical Grid Search. The method we propose, BootBOGS, uses bootstrap with Bayesian Search to select efficient hyperparameter ranges for Grid Search, in order to perform an efficient search in hyperparameter space. In the following, we provide information and details about the BootBOGS methodology used on the MLP-based model.

A. Selection of MLP's hyperparameters

MLP's hyperparameters are selected from three distinct types: those pertaining to the network's structural configuration, those governing the learning and optimization process, and those responsible for introducing regularization effects. In our study, we deliberately singled out one key hyperparameter from each of these categories. Within the first category, we elected to manipulate the number of nodes in each layer, recognizing that this choice is intrinsically intertwined with the nature of the dataset, including factors like the number of hidden layers and the activation functions applied. In the second category, our focus was on hyperparameters such as the learning rate, loss function, batch size, optimizer, and loss function. Among these, we gave precedence to the learning rate, as the others primarily relate to the dataset's feature correlations. Lastly, for the third category, we considered the dropout rate, primarily due to its linkage with feature correlations within the dataset, denoted as Lambda. These meticulous selections aim to optimize our neural network's performance by fine-tuning critical aspects of its configuration and learning process.

B. Bootstrap Resampling

Bootstrap (17) is a statistical inference technique that has become increasingly popular in recent years, due to its versatility and ability to provide reliable estimates of uncertainty in statistical analyses. It is widely used to estimate the sampling distribution of a quantity or to build confidence intervals by generating new data sets through resampling with the replacement of the original data set. Percentile bootstrap confidence intervals are a valuable statistical tool that can be used to estimate the uncertainty or variability of sample statistics such as mean. Achieved through resampling from the original dataset, these intervals come into their own when the true distribution of data remains unknown or when dealing with small sample sizes. One of the remarkable attributes of these intervals is their applicability across a spectrum of scenarios, regardless of the underlying distribution's complexity. In the realm of hyperparameter optimization, these confidence intervals offer an insightful approach to identifying the optimal parameter values.

C. Proposed Method

The algorithm 1 gives us an overview of the step-by-step process of the proposed method.

Algorithm 1: *BootBOGS*

Split dataset, utilize only the train dataset for the bootstrap step
Choose the range of search space for hyperparameters
repeat *2* **times**
  1) **repeat** *10* **times**
    - *Bootstrap sampling train set with Stratified K-Folds cross-validator*
    - *Apply HyperOpt TPE and store the combination of hyperparameters*
    - *Get the AUC score performance for the test set*
  end
  2) *Calculate 68% confidence interval from the list of scores (AUC)*
  3) *Remove hyperparameter values combinations that have score out of this confidence interval*
  4) *Create a new search space of HyperOpt TPE based on this new list of hyperparameters*
end
Use the new reduced search space for Grid Search

First, after splitting the dataset into trainning and testing sets, Bootstrap resampling is applied, iterating this process n times on the training set. It appears as a dynamic tool, allowing a complete exploration of performance variations through various combinations of hyperparameters. This iterative technique not only provides a holistic understanding of the ranges in which these hyperparameters work but also facilitates the identification of optimal values that give better model performance. Here, We choose n equal 10. At each iteration, Bayesian optimization, and more specifically the tree structure of the Hyperband Parzen (TPE) estimator, is used with HyperOpt, a hyperparametric optimization library, to identify optimal parameter combinations in the resampled training set. HyperOpt TPE efficiently navigates through the hyperparameter space, rapidly converging toward promising parameter regions. Subsequently, this optimal combination is used within the test set to compute the AUC score. From these values, we derive a confidence interval that provides a measure of the statistical significance of our results. At this step, striking the delicate balance between hyperparameters and model performance requires a discerning strategy. A bootstrap confidence interval of the 68% percentile of AUC values is used within a region of ± 1 standard deviation. This confidence interval provides a region within ±1 standard error of the mean. This narrower interval provides a reliable range in which optimal hyperparameters are likely to reside. Expanding the perspective and opting for a broader CI 95% could also be considered, but should not be taken lightly, since it has the potential to inadvertently dilute the precision of the analysis. We proceed with a second round of bootstrap sampling, employing the new search space to

iteratively fine-tune the hyperparameters. This dual-iteration process substantially narrows down the scope of exploration, leading to a more focused and efficient search for optimal hyperparameter values. The Grid Search is then used on the new reduced search space to select the best hyperparameters from the search space.

IV. Experimental Results and Evaluation

In this section, we detail the experiments used to validate the proposed BootBOGS method on four available real datasets from various application domains. We compare BootBOGS' results to three traditional approaches, classical Grid Search, Random Search, and Bayesian Optimization. The proposed method is also compared with the most recent studies carried out on the same data sets.

A. Experimental Settings

BootBOGS is coded in Python language using the packages sci-kit-learn, TensorFlow, and Keras. The training model was run on a computer with Intel(R) Core(TM) i5 - 1.70GHz processor and 8GB RAM. The main hyperparameters of the model are shown in Table II. As shown in Table III, BootBOGS focuses on the value intervals of the following three hyperparameters: dropout, learning rate, and number of neurons in the hidden layer. The number of neurons in the experiments is chosen according to the rule of thumb (18) that considers the number of neurons as a power of two, 2^k, and the maximum number of hidden neurons as 2^{k-1}, where k is the number of input elements. This approach can help simplify the calculation of the number of neurons and reduce the number of values to consider during hyperparameter tuning. We've established a dropout range between 0.0 and 0.5 to strike a balance between retaining valuable information and preventing overfitting.

TABLE II: Main model hyperparameters.

Hyperparameter	Hyperparameter Value
Kernel	normal
Activation function	Relu
Activation function in output layer	sigmoid
Batch size	32
Epoch	100
Optimizer	Adam
Loss function	binary cross entropy

TABLE III: Hyperparameters range for experiments.

Hyperparameter	Min value	Max value
Dropout rate	0.0	0.5
Learning rate	0.001	0.1
number of neurons	4	144

The code of our experiments can be accessed with the following link: https://github.com/thiphan94/BootBOGS.

B. Datasets

We use four datasets of binary classification tasks in order to validate BootBOGS. The first dataset, known as the Diabetes PIMA dataset (19), sourced from Kaggle, comprises 768 instances, characterized by eight characteristics including glucose level, blood pressure, and body mass index (BMI), together with a target variable that denotes the presence or absence of diabetes.

The second dataset, the German Credit dataset (20), is related to the evaluation of credit risk. Comprising 1,000 instances, each housing 21 attributes, this dataset is instrumental in determining if a loan applicant is a favorable or risky credit candidate.

The third dataset, the Taiwan Bankruptcy Prediction dataset (21), encapsulates financial ratios extracted from Taiwanese companies' annual reports. With 6819 instances and 96 attributes, this dataset is commonly used to predict whether a company is likely to go bankrupt.

The fourth dataset used is the Polish Bankruptcy dataset from the third year (22). This compilation of financial indicators is designed to predict corporate bankruptcy. Sourced from the UCI Machine Learning Repository, it encompasses data from Polish companies, featuring 10,503 instances and 64 attributes.

Table IV lists the number of attributes, the number of instances, and the relative size of the majority class for each dataset.

TABLE IV: Datasets used in our experiments.

Dataset	Attrs	Size	Majority class(%)
Diabetes (19)	8	768	65.1
German Credit (20)	21	1000	70
Taiwan Bankruptcy (21)	96	6819	96.77
Polish Bankruptcy (third year) (22)	64	10503	95.3

The flowchart in Figure 2 gives us a visual representation of the step-by-step process of data preprocessing.

In the context of data preprocessing, missing value imputation is a critical process where missing data points in a dataset are filled using various techniques. In some datasets, we chose to use median imputation as a method to handle missing values. This approach involves replacing missing values with the median value of the corresponding feature. Moving on to another preprocessing step, the identification and management of outliers plays a significant role in the preparation of a data set for machine learning models. Outliers, which are data points substantially different from the majority, can distort relationships between features and the target variable, negatively affecting model performance. To address outliers in the diabetes dataset, we adopted the Interquartile Range (IQR) method. This method involves identifying observations with values greater than 1.5 times the IQR and classifying them as outliers. These outliers are then removed from the dataset to prevent undue influence on the subsequent modeling process.

The next pre-processing step is feature selection, a crucial process to improve the interpretability and efficiency of the

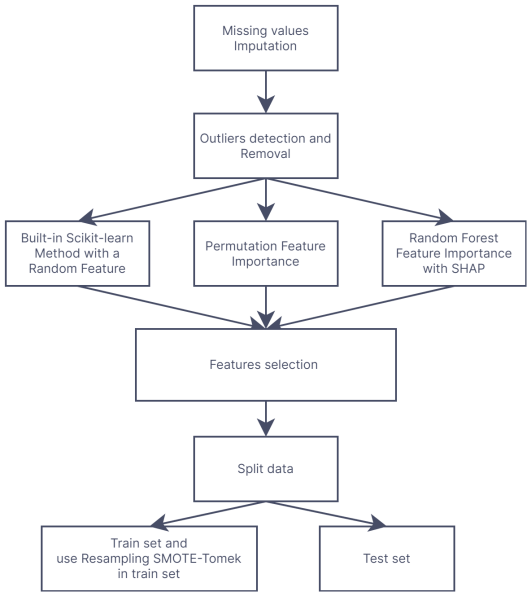


Fig. 2: Data Preprocessing.

model. For this dataset, we used three distinct techniques for feature selection: random variable, permutation, and SHAP. Employing the Random Forest algorithm, a widely used technique for gauging feature importance, we adopted three strategies:

- Built-in Scikit-learn Method with a Random Feature: This approach leverages Scikit-learn's built-in feature selection method using Random Forest. It identifies pertinent features that significantly contribute to the model's predictive capability. A random feature is introduced as a reference point against which the importance of the actual features is assessed. The lower importance of a real feature compared to the random feature might indicate a chance-driven significance, necessitating further scrutiny.
- Permutation Feature Importance: This technique involves shuffling the values of a single feature and assessing its impact on the model's performance. Features demonstrating substantial influence on model performance are regarded as important for prediction.
- Random Forest Feature Importance with SHAP: SHAP is a comprehensive approach for interpreting machine learning model outputs. It evaluates the importance of features by considering all possible combinations of features. This method provides a nuanced understanding of each feature's contribution to model predictions.

Furthermore, since all datasets are imbalanced, meaning that there are significantly more instances of negative cases (class 0) in the datasets than positive ones (class 1), we applied SMOTE-Tomek (23) which combines the strengths of SMOTE and Tomek links. It first applies SMOTE to over-sample the minority class, creating synthetic instances. Then, Tomek links are used to identify and remove ambiguous samples that might cause misclassification. The result is a balanced dataset that preserves the distinctive characteristics of both classes. SMOTE-Tomek helps models generalize better by providing a more balanced training set, reducing bias, and improving their ability to capture the minority class. SMOTE-Tomek is applied to datasets with a high majority class ratio, where the relative size of the majority class exceeds the critical threshold of 70% with scaling features. Note that before applying the resampling technique, each dataset is split into 80% training set and 20% test set.

BootBOGS is evaluated using common metrics: accuracy, AUC, G-Mean, and F_β score for imbalanced datasets. Accuracy, calculated as the ratio of correctly classified instances to the total, tends to favor the majority class. As such, it can lead to suboptimal hyperparameter settings that neglect the minority class and fail to capture rare but critical events. So, for an imbalanced dataset, accuracy cannot be considered a reliable measure. Complementary metrics are used, including Area Under the Receiver Operating Characteristic Curve (AUC), Geometric Mean (G-Mean), and another measure F_β.

Geometric Mean (G-Mean) is defined as the square root of the product of true positive rate (or recall) and true negative rate (or specificity) and is formulated as follows:

$$G - Mean = \sqrt{TPR \times TNR} \tag{1}$$

Where true positive rate (TPR) and true negative rate (TNR) are :

$$TPR(Recall) = \frac{TP}{TP + FN} \tag{2}$$

$$TNR(Specificity) = \frac{TN}{TN + FP} \tag{3}$$

F_β score is a weighted harmonic mean of precision and recall. The formula of F_β is as follows (24):

$$\mathrm{F}_\beta\text{-score} = \frac{TP}{TP + \frac{1}{1+\beta^2}(\beta^2 FN + FP)} \tag{4}$$

Where TP, FP, and FN present the counts of true positives, false positives, and false negatives, respectively. In our experiments, we set the beta value beta for the F_β score to 2. This choice of F2 score allows us to emphasize the importance of recall over precision.

Through the systematic application of these evaluation metrics, we obtained a comprehensive understanding of how well BootBOGS performs hyperparameter optimization. These metrics not only allow us to quantify the quality of the resulting models but also provide insights into their adaptability across various scenarios, including situations with imbalanced class distributions. The insights gained from these metrics play a crucial role in validating the effectiveness of our approach and its potential to redefine the landscape of hyperparameter optimization for machine learning models.

V. Results and Discussion

In this section, we evaluate the performance of BootBOGS in various datasets using the metrics presented previously in order to see how the proposed approach is an improvement

compared to the classical Grid search. The results in TableV show the performances of the Grid Search method, Random Search method, Bayesian method, and BootBOGS for the datasets used in experiments.

TABLE V: Evaluation metrics of methods in various dataset.

Dataset	Method	AUC	Accuracy	G-Mean	F2
Diabetes	GS	0.877	0.889	0.877	0.837
	RS	0.901	0.913	0.90	0.867
	BO	0.877	0.889	0.877	0.837
	BootBOGS	**0.914**	**0.921**	**0.914**	**0.888**
German Credit	GS	0.71	0.77	0.704	0.591
	RS	0.691	0.755	0.673	0.54
	BO	0.705	0.755	0.695	0.585
	BootBOGS	**0.735**	**0.79**	**0.723**	**0.612**
Taiwan Bankruptcy	GS	0.64	0.942	0.554	0.301
	RS	**0.75**	0.932	**0.732**	**0.468**
	BO	0.74	0.937	0.72	0.465
	BootBOGS	0.67	**0.944**	0.61	0.361
Polish Bankruptcy	GS	0.716	0.950	0.667	0.458
	RS	0.762	0.94	0.736	0.483
	BO	**0.77**	**0.955**	**0.742**	0.514
	BootBOGS	0.757	0.953	0.726	**0.556**

Note: GS: Grid Search, RS: Random Search, BO: Bayesian search.

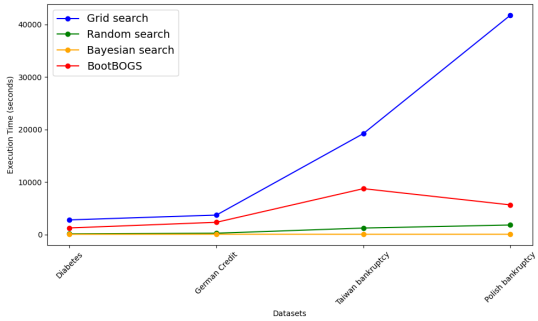
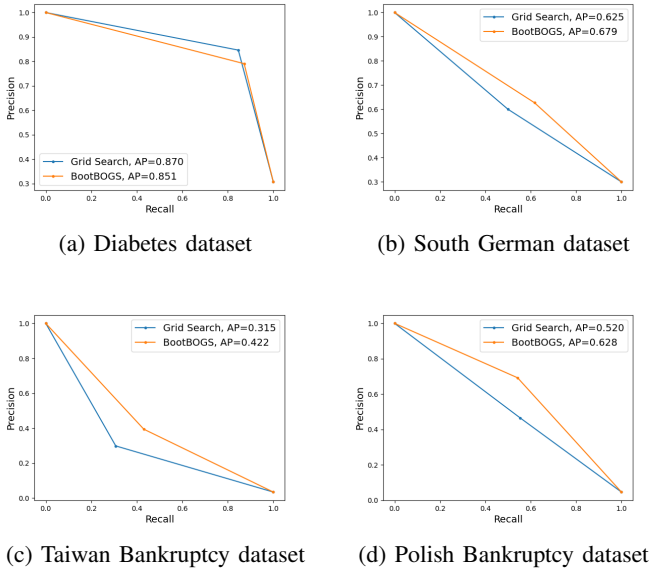


Fig. 3: Execution time (seconds) of four method through various datasets

From the results shown in Table V, we can see that BootBOGS always identifies hyperparameter values that yield better results than traditional Grid Search across the four datasets, evaluated based on AUC, Accuracy, and G-mean metrics and F2 score. In the case of the Taiwan Bankruptcy and Polish Bankruptcy datasets, BootBOGS' performance is not the best when compared to Random Search and Bayesian. However, the achieved performance remains very close to theirs. The fact that those datasets are significantly imbalanced class distribution (96.77% for Taiwan Bankruptcy, 95.3% for Polish Bankruptcy, as opposite to Diabetes with 65.1%, and Diabetes with 65.1%) suggests that in scenarios where the majority class dominates to such a degree, alternative optimization methods may be more effective. Conversely, across all four optimization methods, our approach consistently excels when class imbalances are less pronounced.

A common performance statistic for assessing the effectiveness of binary classification models, particularly when the data is unbalanced, is the AUC-PR score (Area Under the Precision-Recall Curve) (25). It evaluates a model's ability to distinguish between the positive and negative classes by taking into account precision and recall. The AP score, which we now know also represents the AUC-PR score, is shown in Figure 4.



(a) Diabetes dataset

(b) South German dataset

(c) Taiwan Bankruptcy dataset

(d) Polish Bankruptcy dataset

Fig. 4: Precision Recall curves of Grid Search and BootBOGS through four datasets

In terms of speed of execution, figure 3 shows that BootBOGS always reduces execution compared to Grid Search, although it still takes more time than Random Search and Bayesian. Time gain over Grid Search becomes quite significant for the Diabetes dataset (54.7%) and for the German Credit dataset (37.3%).

A comparison of BootBOGS with previous research is presented in Table VI. It shows the efficiency of our method compared to other studies with the same datasets (information such as hyperparameters values or performance metrics for those works are provided when made available) by authors. BootBOGS improves most of the performance metrics, even though it uses the simple model of a neural network with only one hidden layer and a small search space for hyperparameter tuning.

VI. Conclusion

The proposed method BootBOGS presented in this paper is an improved technique compared to Grid Search. The outcomes of our study underscore the significance of intelligent hyperparameter tuning in enhancing the predictive capabilities of machine learning models. By strategically incorporating bootstrap sampling, we have not only streamlined the search process but also injected diversity into the hyperparameter space, enabling more robust exploration.

Our experimental study has several limitations. One limitation of our study is that our method was developed in the context of binary classification. Acknowledging the binary classification limitation, we fine-tuned our approach to excel

TABLE VI: Results comparison of different approaches using various datasets.

Model & optimizer	Hyperparameters	Performance
Diabetes dataset		
(26): MLP (NM)	hidden layer:2 neurons: 64,32 dropout rate: 0.25,0.5 loss function: Adadelta	Accuracy: 84.11%
(27): MLP (NM)	hidden layer:3 neurons: 100, 100, 100	Accuracy:82% AUC: 78.7%
(28): MLP (NM)	hidden layer:10 momentum: 0.9 learning rate: 0.003	Accuracy: 77.6% AUC: 84.6%
MLP (1 hidden layer) BootBOGS	neurons: 120 learning rate: 0.0283 dropout rate: 0.2	Accuracy: 92.1% AUC: 91.4%
German Credit dataset		
(29):MLP Grid Search	hidden layer: 3 neurons: 16,64,8 batch size: 64 epoch: 250	Accuracy: 77.10% AUC: 78.11%
(30):MLP Genetic algorithm	hidden layer: 1 neurons: 7 a.f: tansig, purelin t.f: traincgp	Accuracy: 87.10%
MLP (1 hidden layer) BootBOGS	neurons: 120 learning rate: 0.0325 dropout rate: 0.2 a.f: ReLU batch size: 32 epoch:100	Accuracy: 79.0% AUC: 76.48%
Taiwan Bankruptcy dataset		
(31): MLP (90% outliers)	NM	Accuracy: 64.2%
(32): MLP Model selection	hidden layer:1 neurons:5 a.f: logistic epoch: 206 optimizer: Adam	Accuracy: 86.06%
MLP (1 hidden layer) BootBOGS	neurons: 121 learning rate: 0.001 dropout rate: 0.0 a.f: ReLU epoch:100 optimizer: Adam	Accuracy: 94.4% AUC: 67.5% F2: 36.1%
Polish Bankruptcy dataset		
(33):Extreme Gradient Boosting	NM	Accuracy: 94%
(34):Adaptive whale optimization algorithm with deeplearning (AWOA-DL) Whale optimization	hidden layer: 1,2,3 learning rate: 0.1 a.f: sigmoid	Accuracy: 98.23%
MLP (1 hidden layer) BootBOGS	neurons: 105 learning rate: 0.004 dropout rate: 0.1 a.f: ReLU	Accuracy: 95.3% AUC: 75.7

Note: NM means that not mentioned, a.f: activation function, t.f: training function

in such scenarios. Another key direction for our work is the extension of BootBOGS to multi-class classification problems. This is a significant step forward in making BootBOGS capable of addressing real-world problems. Its robustness and adaptability allow us to identify the challenges associated with its use in a regression framework. This limitation offers a clear path for future research.

Moreover, the promising results obtained in this work pave the way for several avenues to test the approach on complex data such as images or text.

In our initial efforts, we fine-tuned just three key hyperparameters: the dropout rate, the learning rate, and the number of neurons in the hidden layer. As our research advances and we delve into more complex models, our plan is to broaden our optimization efforts, encompassing a wider array of hyperparameters, in particular the number of hidden layers. This expansion will further refine and enhance the performance of our machine learning models.

And finally, the study focuses on MLP hyperparameters. In terms of future work, a promising direction could be to extend BootBOGS to other machine learning methods.

REFERENCES

[1] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:15700257

[2] H. Larochelle, D. Erhan, A. C. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *International Conference on Machine Learning*, 2007. [Online]. Available: https://api.semanticscholar.org/CorpusID:14805281

[3] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018. [Online]. Available: http://jmlr.org/papers/v18/16-558.html

[4] S. Falkner, A. Klein, and F. Hutter, "Bohb: Robust and efficient hyperparameter optimization at scale," in *International conference on machine learning*. PMLR, 2018, pp. 1437–1446.

[5] D. Marinov, "Hyperparameter optimisation with early termination of poor performers 1," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:198936232

[6] J. Carmichael and Y. Liao, "Bayesian optimization of multi-layer perceptron models for power distribution system state estimation," *International Journal of Emerging Electric Power Systems*, 2023. [Online]. Available: https://doi.org/10.1515/ijeeps-2022-0328

[7] M. Feurer and F. Hutter, *Hyperparameter Optimization*, 05 2019, pp. 3–33.

[8] Z. B. Zabinsky, "Random search algorithms," 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:17261884

[9] D. Navon and A. M. Bronstein, "Random search hyperparameter tuning: Expected improvement estimation and the corresponding lower bound," 2022.

[10] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," *Towards Global Optimization*, vol. 2, no. 117-129, p. 2, 1978.

[11] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, pp. 26–40, 03 2019.

[12] T. Yamashita, N. Sato, H. Kino, T. Miyake, K. Tsuda, and T. Oguchi, "Crystal structure prediction accelerated by bayesian optimization," *Phys. Rev. Mater.*, vol. 2, p. 013803, Jan 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevMaterials.2.013803

[13] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

[14] S. Daulton, X. Wan, D. Eriksson, M. Balandat, M. A. Osborne, and E. Bakshy, "Bayesian optimization over discrete and mixed spaces via probabilistic reparameterization," 2022.

[15] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

[16] J. Bergstra, R. Bardenet, B. Kégl, and Y. Bengio, "Algorithms for hyper-parameter optimization," 12 2011.

[17] B. Efron, *Bootstrap Methods: Another Look at the Jackknife*. New York, NY: Springer New York, 1992, pp. 569–593. [Online]. Available: https://doi.org/10.1007/978-1-4612-4380-9_41

[18] P. Grabusts and A. Zorins, "The influence of hidden neurons factor on neural nework training quality assurance," *Environment. Technology. Resources. Proceedings of the International Scientific and Practical Conference*, vol. 3, p. 76, 06 2015.

[19] "Pima Indians Diabetes," Kaggle, 1988, DOI: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database.

[20] "South German Credit," UCI Machine Learning Repository, 2019, DOI: https://doi.org/10.24432/C5X89F.

[21] "Taiwanese Bankruptcy Prediction," UCI Machine Learning Repository, 2020, DOI: https://doi.org/10.24432/C5004D.

[22] S. Tomczak, "Polish companies bankruptcy data," UCI Machine Learning Repository, 2016, DOI: https://doi.org/10.24432/C5F600.

[23] G. E. A. P. A. Batista, A. L. C. Bazzan, and M. C. Monard, "Balancing training data for automated annotation of keywords: a case study," in *WOB*, 2003. [Online]. Available: https://api.semanticscholar.org/CorpusID:1579194

[24] H. He and Y. Ma, "Imbalanced learning: foundations, algorithms, and applications," 2013.

[25] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," vol. 06, 06 2006.

[26] A. Ashiquzzaman, A. K. Tushar, M. R. Islam, D. Shon, K. Im, J.-H. Park, D.-S. Lim, and J. Kim, "Reduction of Overfitting in Diabetes Prediction Using Deep Learning Neural Network," in *IT Convergence and Security 2017*, K. J. Kim, H. Kim, and N. Baek, Eds. Singapore: Springer Singapore, 2018, vol. 449, pp. 35–43, series Title: Lecture Notes in Electrical Engineering. [Online]. Available: http://link.springer.com/10.1007/978-981-10-6451-7_5

[27] G. Verma and Hemraj Verma, "A Multilayer Perceptron Neural Network Model For Predicting Diabetes," 2020, publisher: Unpublished. [Online]. Available: http://rgdoi.net/10.13140/RG.2.2.23203.89126

[28] R. Saxena, S. K. Sharma, M. Gupta, and G. C. Sampada, "A Novel Approach for Feature Selection and Classification of Diabetes Mellitus: Machine Learning Methods," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–11, Apr. 2022. [Online]. Available: https://www.hindawi.com/journals/cin/2022/3820360/

[29] L. Munkhdalai, O.-E. Namsrai, and K. Ryu, "Credit scoring with deep learning," 02 2018.

[30] H. Kazemi, K. Khalili-Damghani, and S.-N. Soheil, "Tuning structural parameters of neural networks using genetic algorithm: A credit scoring application," *Expert Systems*, vol. 38, 06 2021.

[31] C.-F. Tsai and K.-C. Cheng, "Simple instance selection for bankruptcy prediction," *Knowledge Based Systems - KBS*, vol. 27, 01 2011.

[32] R. F. Brenes, A. Johannssen, and N. Chukhrova, "An intelligent bankruptcy prediction model using a multilayer perceptron," *Intelligent Systems with Applications*, vol. 16, p. 200136, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667305322000734

[33] M. Zieba, S. Tomczak, and J. Tomczak, "Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction," *Expert Systems with Applications*, vol. 58, pp. 93–101, 2016.

[34] M. Elhoseny, N. Metawa, G. Sztano, and I. El-Hasnony, "Deep learning-based model for financial distress prediction," *Annals of Operations Research*, 05 2022.