Neural Network Based Hyperparameter Optimization for Random Forest Models

Paranawithana T.D.
Department of Computer Engineering
Faculty of Engineering
University of Sri Jayewardenepura
Nugegoda, Sri Lanka
en97601@foe.sjp.ac.lk

***Abstract* — Hyperparameter optimization is critical to enhance the performance of machine learning algorithms, such as Random Forest (RF). Classical methods like Grid Search, Random Search, and Bayesian Optimization usually need extensive computational resources and time and may not generalize well to various datasets. This study presents a neural network-based approach for hyperparameter optimization in RF models using dataset characteristics like number of samples, number of features and class imbalance ratio to predict the most optimal hyperparameter settings. In the experiment, a meta-dataset was constructed from multiple real-world datasets by evaluating different hyperparameter combinations and their corresponding accuracies along with the dataset characteristics. A neural network was trained on this meta-dataset to predict accuracy values for various hyperparameter settings, selecting the combination with the highest predicted accuracy. This approach eliminates the need for exhaustive searches and accelerates the tuning process. Findings from the experiment indicates that the proposed method is considerably more efficient and more accurate for the pre-defined hyperparameter grids than Grid Search method. And it keeps a competitive computational efficiency with Bayesian Optimization even though Bayesian optimization was not clearly outperformed by the proposed neural network-based approach. The findings demonstrate the viability of data-driven, meta-learning algorithms for hyperparameter optimization as an effective alternative to optimize RF models in numerous applications.**

Keywords — Hyperparameter Optimization, Random Forest, Neural Networks, Machine Learning.

I. Introduction

A. *Importance of Hyperparameter Optimization*

Hyperparameters of a machine learning model are the settings which are defined before the training process of a machine learning model. Machine learning models depend on various hyperparameters such as number of trees in a Random Forest, depth of decision trees, minimum samples split and the learning rate when it comes to a neural network model. Choosing appropriate hyperparameters is an important task since they control the model performance, generalization capability, and computational complexity of the model. Poorly chosen hyperparameters can lead to underfitting or overfitting, reducing the performance of the machine learning model.

B. *Existing Hyperparameter Optimization Techniques*

Hyperparameter optimization is the task of finding the best hyperparameter configuration for a model and data. The classical hyperparameter optimization techniques are Grid Search, Random Search, and Bayesian Optimization. Grid Search performs an exhaustive search of all hyperparameter combinations, evaluates the model each time and produces the combination with the highest accuracy. This ensures optimal values but at the cost of a very large amount of computation, particularly not feasible for high-dimensional hyperparameter spaces. Random Search is more efficient than Grid Search by randomly sampling hyperparameters within a specified range and evaluates the model with each combination and provides the one with the highest accuracy. This does not guarantee to provide optimal results and struggles in large search spaces. Bayesian Optimization presents a probabilistic method with the exploration-exploitation trade-off. This method is more improved, but it requires a good understanding about the underlying probabilistic model and may not be that effective in scalability with large data or high-dimensional search spaces.

C. *Proposed Neural Network-Based Approach*

Due to the drawbacks of these conventional methods, this research explores the application of meta-learning and neural networks for hyperparameter optimization. Rather than performing an exhaustive search and evaluating the model each time, a trained neural network predicts optimal hyperparameter combinations considering the dataset features which drastically cuts down the computational overhead. This method makes use of a meta-dataset created from multiple real-world datasets so that the model learns to generalize over variety of datasets. Through the recognition of trends of the previous hyperparameter trials, the neural network can effectively suggest settings that are expected to yield high accuracy and hence makes the optimization process adaptive and faster.

The remainder of this paper is structured as follows. Section II (Literature Review) reviews existing research on hyperparameter optimization with a focus on different techniques and their limitations. Section III (Methodology) presents the proposed methodology. Section IV (Experiments and Results) discusses the experimental setup including dataset preparation, meta-learning framework, and model training and performance evaluation. Section V (Discussion) provides a discussion of the results, and Section VI (Conclusion and Future Work) concludes the study with potential future research directions.

II. Literature Review

A. Importance of Hyperparameters in Random Forest Models

Hyperparameters Random Forest (RF) models depend on a set of key hyperparameters whose influence can considerably change performance in most machine learning tasks. Probst & Boulesteix (2018) state, the number of trees (n_estimators) as a key parameter with many performance gains being within the initial 100 trees and then plateauing [4]. Trithipkaiwanpon & Taetragool (2021) demonstrate that the min_samples_leaf parameter significantly affects classification accuracy and F1-score on a set of datasets [1].

Bernard et al. (2009) discusses that the count of randomly drawn features (max_features) at every node is of great significance in constructing precise RF classifiers. Even though default values are close to optimal, sometimes they can be exceedingly sub-optimal based on the dataset [2]. Probst et al. (2018) note that other hyperparameters, such as the number of observations sampled per tree, splitting rules, and minimum node size, also affect model stability and accuracy [4]. Although RF models can perform well under default hyperparameters, Probst (2019) describes that hyperparameter tuning can achieve substantial improvements, particularly when dealing with complex datasets [3].

Huang & Boutros (2016) highlight that incorrect hyperparameter selection can also impact variable importance measurements, resulting in inaccurate interpretations in feature selection problems [5]. A number of tuning techniques have been investigated to optimize RF hyperparameters and Probst et al. (2018) have identified model-based optimization (MBO) as a prominent method [4]. As RF models are sensitive to parameter selection, effective hyperparameter tuning techniques must be developed to maximize predictive performance while preserving computing efficiency.

B. Hyperparameter Optimization Techniques

Hyperparameter tuning is key to boosting the accuracy of machine learning algorithms. Muzayanah et al. (2024) state that Grid Search provides high accuracy at a high computational cost as it performs an exhaustive search over all the possible combinations of hyperparameters [6]. Stuke et al. (2020) show that Random Search provides a trade-off between accuracy and computational efficiency because it selects random combinations instead of attempting all available combinations [7].

The resarch paper written by Putatunda & Rama (2018) observe Bayesian Optimization with Hyperopt to be better than Grid and Random Search both in accuracy and computation time [8]. However, in Random Forest models for diabetes prediction, Muzayanah et al. (2024) observed Grid Search to yield marginally better accuracy than Bayesian Optimization but at significantly more time [6]. Stuke et al. (2020) discuss that for kernel ridge regression in computational chemistry, Random Search and Bayesian Optimization scaled more efficiently with the number of hyperparameters, performing the same or better than Grid Search [7].

According to Jia Wu et al. (2019), the suggested Bayesian Optimization approach in their study determines the optimal hyperparameters for popular models like Random Forest, Neural Networks, and Multi-Grained Cascade Forests while also taking time efficiency into account [9]. A study by Louise Bloch et al. (2020) explain that Bayesian Optimization outperforms Grid Search in terms of computing efficiency when it comes to tuning the hyperparameters of Random Forest and XGBoost models, resulting in equivalent or better performance on a test that predicts Alzheimer's disease [10]. Based on Subhasis Dasgupta et al. (2024), Tree-Structured Parzen Estimator (TPE) works better for classification issues, but Random Search is more efficient for regression problems. According to the results, there is no one ideal way to tune hyperparameters because the best option varies depending on the model type and task specifications [11].

The insights from this literature review highlight the limitations of existing hyperparameter optimization techniques, especially regarding computational efficiency and adaptability to diverse datasets. These challenges emphasize the necessity of a data-driven, novel approach for hyperparameter optimization in Random Forest models which is explored in the following section.

III. Methodology

The proposed approach uses a neural network trained on a meta-dataset to optimize the hyperparameters of the Random Forest model. The process consists of the following steps.

First, several datasets with different characteristics are chosen to build the meta-dataset. Every dataset is characterized by properties such as,

- number of samples
- number of features
- class imbalance ratio
- number of categorical and numerical features

A pre-determined grid of hyperparameters is established for each dataset consisting important RF hyperparameters such as number of trees (n_estimators), maximum tree depth (max_depth), minimum samples per split (min_samples_split), and number of features to consider at each split (max_features). The chosen datasets are utilized to train RF models for various combinations of hyperparameters, and Grid Search is used to test their performance. The accuracy obtained for each setting is recorded, creating the meta-dataset.

The neural network model is trained on the meta-dataset to learn the relationship between dataset characteristics, hyperparameter combinations, and their corresponding accuracies. Dataset characteristics and hyperparameter settings are input to the model, and the model output is the predicted accuracy.

On a new test dataset, the trained neural network predicts the accuracy of all the various combinations of hyperparameters in a grid of predefined values. The combination with the best predicted accuracy is chosen as the optimal hyperparameter set. While preserving good model performance, this method lowers computing costs when compared to exhaustive search techniques.

IV. Experiments and Results

An experiment was carried out to evaluate the performance of the proposed neural network-based approach for hyperparameter optimization in RF models and compared it with conventional approaches including Grid Search and Bayesian Optimization for accuracy and efficiency.

A. Meta-Dataset Creation

A In the experiment carried out, 10 real-world binary classification datasets with a variety of characteristics were chosen to create a strong meta-dataset to feed later to the neural network to be trained for hyperparameter optimization. To guarantee thorough coverage of various data distributions, these datasets were selected based on distinct variations in the number of samples, features, class imbalance ratios, number of categorical and numerical features. There, number of samples range from 700 to 8000 and number of features range from 4 to 20 features across the selected datasets. And all the 10 datasets are in csv file format.

An overview of the selected datasets is given in Table I, which includes information on the number of features, samples, categorical feature count, and class imbalance ratios.

TABLE I. Datasets Used for Meta-Dataset Creation

Dataset Name	**No. of Samples**	**No. of Features**	**Class Imbalance Ratio**	**Categorical Feature Count**
Heart Disease	1025	13	1.0541	8
Titanic Survival	712	9	1.4722	5
Diabetes Diagnosis	768	8	1.8657	0
BankNote Authentication	1348	4	1.2098	0
Water Quality	7999	20	7.7708	0
Wine	1359	11	1.1234	0
Age Prediction	3524	5	1.0000	0
Gender Classification	3233	7	1.2296	5
Brain Stroke	4981	11	9.0847	8
AIDS Classification	5000	18	2.1666	11

Before being used for hyperparameter evaluation, each dataset was put through a thorough preprocessing to guarantee consistency across all samples. Some of the preprocessing techniques used are listed below.

- *Handling Missing Values*: Mean imputation was used to treat datasets with missing values for categorical and numerical variables respectively.
- *Feature Scaling*: To guarantee consistency across various datasets, numerical features were normalized.
- *Categorical Encoding*: one-hot encoding was used for categorical features where appropriate.
- *Elimination of Duplicates*: To avoid redundancy in the training data, any duplicate entries were found and eliminated.
- *Data Shuffling*: Data was shuffled to guarantee randomness in the distribution of the data and avoid bias in model training.

The following hyperparameters related to Random Forests were considered for the evaluation.

- *Number of Trees (n_estimators)*: This parameter regulates the number of decision trees in the forest. The accuracy is generally better the more trees there are, but at a computational expense.
- *Maximum Depth of Trees (max_depth)*: This hyperparameter limits the depth of all decision trees to prevent overfitting. A predefined range of depth values was taken to balance model complexity and generalization.
- *Minimum Samples to Split (min_samples_split)*: This parameter controls the minimum number of samples that are required to split a node, to avoid overfitting by only splitting when significant.
- *Feature Selection Strategy (max_features)*: Various strategies like sqrt (square root of the total features) and log2 (logarithm base 2 of the total features) were tried out to identify the most appropriate feature selection for every split.

A hyperparameter grid was established for every dataset, and Random Forest models were trained with the help of the Grid Search method utilizing a variety of hyperparameter combinations. Then the respective accuracies were recorded. About 1,300 samples made up the resulting meta-dataset, which represented the dataset's characteristics such as sample size, feature count, class imbalance ratio, number of categorical and numerical features, various combinations of selected hyperparameters, and corresponding accuracy values. The training of a neural network-based model for hyperparameter optimization was made easier by this structured approach, which made it possible to automatically choose the best configurations depending on dataset characteristics.

B. Neural Network Model Architecture and Training

A neural network model was designed for this experiment in order to predict the performance of given hyperparameter combinations based on dataset properties. The model's architecture was chosen to accurately represent the relationship between hyperparameter performance and dataset characteristics. Fig. 1 displays the designed neural network architecture.

- *Input Layer*: The dataset's attributes (number of samples, number of features, class imbalance ratio, number of categorical and numerical features) and selected hyperparameters (number of estimators, maximum tree depth, feature selection strategy, and minimum samples needed to split an internal node) are all included in the feature vector that the model receives.
- *Hidden Layers*: Non-linearity was introduced using three densely connected hidden layers triggered by Rectified Linear Unit (ReLU) activation. Batch Normalization was used to stabilize training and converge quickly. To avoid overfitting, dropout layers were inserted at each hidden layer.
- *Output Layer*: The accuracy of the given hyperparameter combination was predicted using a single neuron with a sigmoid activation function, with the output bound between 0 and 1.

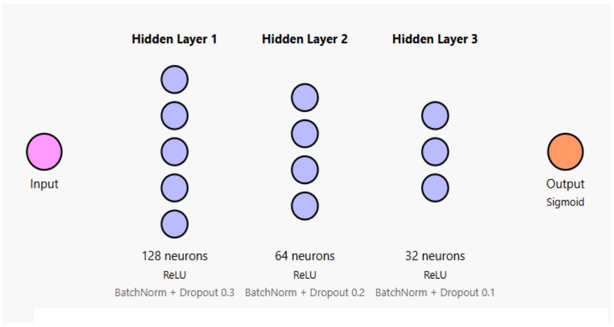

Fig. 1. Architecture of the designed neural network

The meta-dataset was shuffled before feeding it to the model for training. This ensured randomness during training, preventing any bias due to sequential data ordering.

Adam optimizer was used to modify the learning rate for effective convergence and the model was trained for 50 epochs. Mean Squared Error (MSE) served as the loss function and Mean Absolute Error (MAE) was utilized as a performance evaluating measure.

The meta-dataset was split 80-20 into training and validation sets to ensure there was enough data for both assessment and learning. During training, dropout layers randomly deactivated neurons to assist avoid overfitting. Batch normalization further increased stability. To maximize training efficiency, early stopping callback tracked validation loss and stopped training after ten consecutive epochs if no improvement was seen. Furthermore, if validation loss plateaued, the learning rate was lowered by a factor of 0.5, guaranteeing a refined learning procedure.

MAE and accuracy plots were generated during training to monitor model performance. The plots, as seen in Fig. 2 and Fig. 3, offer insight into training dynamics. The MAE plot demonstrates a gradual decrease in training curve, reflecting a gradual decrease in prediction error over epochs. Validation MAE follows the same decreasing pattern. MAE also demonstrates some fluctuations, which reflect minimal variations in training and validation performance while following an overall satisfying pattern. The accuracy plot display rising curves for both training and validation accuracy, which corresponds to progressive learning and achieved validation accuracy was 90%. Validation accuracy closely tracks the training accuracy, which means the model does not seem to be overfitting and can adapt effectively to new data.

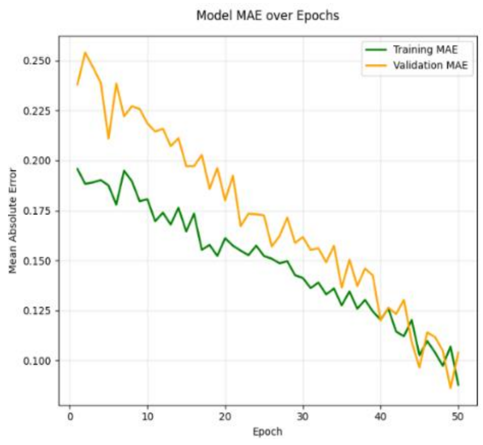

Fig. 2. Mean Absolute Error over Epochs

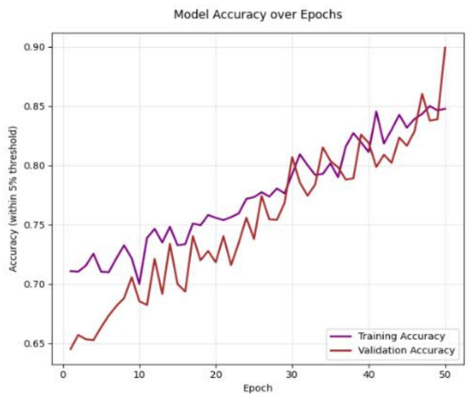

Fig. 3. Accuracy over Epochs

C. *Testing the Model Performance and Comparison*

Four binary classification datasets were used as test datasets to evaluate the proposed neural network-based approach for hyperparameter optimization in Random Forest models. A hyperparameter grid was defined for each datasets with selected hyperparameters. Tabel II shows a comparison of the achieved results of accuracy and computational time of the proposed method against Grid Search and Bayesian Optimization.

TABLE II. COMPARISON OF RESULTS ON TEST DATASETS

Dataset Name	**Grid Search**		**Bayesian Optimization**		**Proposed approach**	
	Accuracy	**Time (s)**	**Accuracy**	**Time (s)**	**Accuracy**	**Time (s)**
Liver Disease	0.9749	87.21	1.0000	6.91	0.9875	13.26
Diabetes Classificati-on	0.9695	260.98	0.9993	22.46	0.9878	13.43
Patient Treatment	0.9456	149.61	1.0000	15.35	0.9924	12.19
Loan Prediction	0.9613	54.21	1.0000	5.52	0.9688	10.82

V. Discussion

The experimental results in Table II reveal that the neural network-based hyperparameter optimization approach maintains competitive accuracy while efficiently cutting down on computational time when compared to Grid Search. As opposed to Grid Search, which involves continuously training Random Forest models, the proposed approach evaluates a predetermined hyperparameter grid, predicts accuracy for every combination, and chooses the one with the highest predicted accuracy. This simplifies the hyperparameter selection procedure while lowering computing overhead. Although Bayesian Optimization appears to be the most efficient and accurate method, the proposed approach maintains a strong competition in terms of efficiency, beating Bayesian Optimization in two of the four testing occasions by having less computational time.

The effectiveness of the proposed approach depends on the quality and diversity of the meta-dataset, which determines how well the neural network generalizes to new and unseen data. The use of a fixed hyperparameter grid limits flexibility of the method by preventing it from exploring configurations other than those in training, even if it produced consistent accuracy across test datasets.

VI. Conclusion and Future Work

This research proposed a neural network-based approach for hyperparameter optimization in Random Forest models as a replacement to exhaustive search methods for hyperparameter optimization. The model chooses the combination with the highest predicted accuracy after predicting accuracy values for all the hyperparameter combinations in a predefined hyperparameter grid. According to experimental results, this approach achieves competitive accuracy while outperforming Grid Search in terms of efficiency because it does not require iterative training for Random Forest models. Even though, Bayesian Optimization remains as the best approach, proposed method maintains a strong competition in terms of efficiency.

Future work can include making the meta-dataset even more generalizable, combining the proposed approach with Bayesian Optimization for even more precise optimization, and adapting the method for use with other models like XGBoost or deep learning models. Apart from that, rather than considering all potential combinations within a pre-defined grid, one can develop a more dynamic way to estimate the optimal hyperparameter combination within a single pass to the model, further enhancing efficiency. With these enhancements, neural network-based hyperparameter optimization would become a more adaptable and scalable solution for real world applications.

References

[1] T. Trithipkaiwanpon and U. Taetragool, “Sensitivity Analysis of Random Forest Hyperparameters,” 2021 18th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), pp. 1163–1167, May 2021, doi: https://doi.org/10.1109/ecti-con51831.2021.9454885.

[2] S. Bernard, L. Heutte, and S. Adam, “Influence of Hyperparameters on Random Forest Accuracy,” Multiple Classifier Systems, pp. 171–180, 2009, doi: https://doi.org/10.1007/978-3-642-02326-2_18.

[3] M. Kim, “Supervised learning‐based DDoS attacks detection: Tuning hyperparameters,” ETRI Journal, vol. 41, no. 5, pp. 560‐573, Sep. 2019, doi: https://doi.org/10.4218/etrij.2019-0156.

[4] P. Probst, M. N. Wright, and A. Boulesteix, “Hyperparameters and tuning strategies for random forest,” Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 9, no. 3, Jan. 2019, doi: https://doi.org/10.1002/widm.1301.

[5] B. F. F. Huang and P. C. Boutros, “The parameter sensitivity of random forests,” BMC Bioinformatics, vol. 17, no. 1, Sep. 2016, doi: https://doi.org/10.1186/s12859-016-1228-x.

[6] R. Muzayanah, D. A. A. Pertiwi, M. Ali, and M. A. Muslim, “Comparison of gridsearchcv and bayesian hyperparameter optimization in random forest algorithm for diabetes prediction,” Journal of Soft Computing Exploration, vol. 5, no. 1, pp. 86–91, Apr. 2024, doi: https://doi.org/10.52465/joscex.v5i1.308.

[7] A. Stuke, P. Rinke, and M. Todorović, “Efficient hyperparameter tuning for kernel ridge regression with Bayesian optimization,” Machine Learning: Science and Technology, vol. 2, no. 3, p. 035022, Jun. 2021, doi: https://doi.org/10.1088/2632-2153/abee59.

[8] S. Putatunda and K. Rama, “A Comparative Analysis of Hyperopt as Against Other Approaches for Hyper-Parameter Optimization of XGBoost,” Proceedings of the 2018 International Conference on Signal Processing and Machine Learning - SPML ’18, 2018, doi: https://doi.org/10.1145/3297067.3297080.

[9] 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 Optimizationb,” Journal of Electronic Science and Technology, vol. 17, no. 1, pp. 26–40, Mar. 2019, doi: https://doi.org/10.11989/JEST.1674-862X.80904120.

[10] L. Bloch and C. M. Friedrich, “Using bayesian optimization to effectively tune random forest and XGBoost hyperparameters for early alzheimer’s disease diagnosis,” Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 285–299, 2021. doi:10.1007/978-3-030-70569-5_18.

[11] S. Dasgupta and J. Sen, “A Comparative Study of Hyperparameter Tuning Methods,” arXiv (Cornell University), Aug. 2024, doi: https://doi.org/10.48550/arxiv.2408.16425.

[12] N. Monica and P. Agrawal, “A Survey on Hyperparameter Optimization of Machine Learning Models,” Mar. 2024, doi: https://doi.org/10.1109/icdt61202.2024.10489732.

[13] Hijrah Nisya, S. N. Hertiana, and Yudha Purwanto, “Implementation of Hyperparameter Tuning Random Forest Algorithm in Machine Learning for SDN Security: An Innovative Exploration of DDoS Attack Detection,” pp. 321–326, Aug. 2024, doi: https://doi.org/10.1109/icoabcd63526.2024.10704521.

[14] A. Wulandari and M. S. Anggreainy, “Implementation of Random Forest Model with Hyperparameter Tuning for Diabetes Mellitus Classification,” 2024 7th International Conference of Computer and Informatics Engineering (IC2IE), pp. 1–5, Sep. 2024, doi: https://doi.org/10.1109/ic2ie63342.2024.10747821.

[15] E. Gunawan, N. I. Afkharinah, N. F. Masnur, K. Mutijarsa, A. Agustan, and S. Yulianto, “Optimization of Random Forest Algorithm for Paddy Growth Stage Classification Using Bayesian Optimization and Oversampling,” 2023 International Conference on Electrical Engineering and Informatics (ICEEI), pp. 1–6, Oct. 2023, doi: https://doi.org/10.1109/iceei59426.2023.10346666.

[16] M. Zhou, X. Zhong, Y. Sun, and L. Gan, “Prediction model of coal consumption based on random forest variable selection and random-grid hyperparametric optimization algorithm,” 2023 International Conference on Power System Technology (PowerCon), pp. 1–5, Sep. 2023. doi:10.1109/powercon58120.2023.10331350.