IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Hyperparameter Optimization Of Machine Learning Methods Using Random Search And Bayesian Optimization To Recognize Digits

[1]Shubhangi Topale, [2]Saniya Samant, [3]Vrushali Samant, [4]Shlesha Patil, [5]Dr. Manisha Mali

[1,2,3,4]Student, [5]Professor

[1]Department of Computer Engineering,
[1]Vishwakarma Institute of Information Technology, Pune, India

Abstract: Hyperparameter optimization plays a significant role in enhancing the performance of machine learning models. In this paper, we present applications of Random search and Bayesian optimization for hyperparameter tuning for four machine learning models like Support Vector Machines (SVM), Decision Trees, K-nearest neighbors (KNN) and Random Forest. We provide a comparative analysis of these methods and investigate their impact on model accuracy, training efficiency, resource consumption, and computational efficiency. Results demonstrate the variation in accuracy of these optimization strategies such as Bayesian Optimization, significantly reduces the search time and improves accuracy. By marking these optimizations, we aim to highlight best practices for achieving high-performance models across the MNIST dataset for handwritten digit recognizer.

Index Terms - hyperparameter optimization, digit recognition, Bayesian optimization, RandomsearchCV optimization.

I. INTRODUCTION

Machine learning models are of immense worth towards the solving of many complex problems across any domains. One of the widely explored problems in machine learning is Handwritten Digit Recognition; the performance of any model is highly dependent on the algorithm selected and their hyperparameter tuning. Recognizing a handwritten digit is of immense importance for several applications, such as bank check processing, postal code reading, etc. Hyperparameters are critical settings that significantly influence how well a machine learning model is able to perform. Otherwise, if the hyperparameters are not chosen rightly, the model might not work well. In fact, poor choices in the hyperparameters can make models overfit, meaning that they do extremely well on training data but fail to generalize to new data; underfit, which are designed so badly that they fail to learn important patterns; or anything in between, which can be said to be worse than the above cases because of lower performance and accuracy.

With all this in mind, the paper investigates how hyperparameter optimization algorithms make possible high performance machine learning models of handwritten digit recognition. For this it employs Random Search where hyperparameter configurations are randomly sampled from given ranges, and Bayesian Optimization- that involves much deeper approach.

We run four unique approaches: Decision tree, K Nearest Neighbours (KNN), Random Forest and Support Vector Machine (SVM). Each of these models will have certain characteristics which may impact how accurate the model might be at identifying handwritten digits. We take as our input feature a dataset that is freely available on Kaggle, consisting of pixel values from images of handwritten digits. What we want to do here is quantify just how much improvement in performance comes from not doing any tuning of hyperparameters as compared to tuning hyperparameters by evaluation of model performance before and after doing said optimization. Below is our main contribution:

1. Comparison of model performance with and without hyperparameter tuning
2. The performance comparison of random search and Bayesian optimization with four ML models for the best identification in handwritten digit recognition
3. Comparative analysis of Decision Tree, KNN, Random Forest, and SVM models based on performance

II. Related Work

A novel approach to hyperparameter optimization of CNNs is presented in Algorithm [8]. Methods classically used, such as grid search and random search, are inefficient for large models or complex networks. Here, the authors devise a genetic algorithm of variable length where an entire model is evolved over multiple generations to increase the depth progressively and fine-tune hyperparameters. This method is well suited to deep learning tasks because this algorithm does not restrict the number of layers, instead offering flexible layer expansion that separates it from other fixed-length GAs. Experimental results on the CIFAR-10 dataset clearly depicted how this algorithm efficiently searches for the best available hyperparameters, which eventually lowers the complexity and time associated with it. Thus, this algorithm has proven particularly beneficial for researchers with limited GPU resources.

Traditional methodologies for hyperparameter tuning, such as grid or random search [3], often turn out to be inefficient when there is less than comprehensive exploration of the search space when it is high-dimensional. More powerful HPO methods include Bayesian Optimization, Evolution Strategies, and Hyperband among others that rely on much more efficient, automated ways of adjusting machine learning models. Results are quite impressive in this regard because systematic optimization of hyperparameters might lead to highly improved model performance with regard to more accurate and reliable predictions. The paper emphasizes the injection of HPO into a pipeline of machine learning to better assess and generalize models. Automated HPO approaches with Bayesian optimization have surpassed other methods by providing a balance within the search space. Hence, these techniques show much promise for further advancement in applications ranging from various fields where computational efficiency coupled with robust model outcomes is ensured.

This paper [4] uses an open-source public Drupal dataset to compare four HPO methods for six machine learning-based SVP models. Different experiments are performed through various Python libraries, such as Sklenar, skopt, hyperopt, and tpot..

They present time-efficient HPO, based on A2C-based RL combined with early stopping in an OpenAI Gym environment in this paper [1]. XGBoost, SVC, and random forest are the ml models used in this paper.

Hyperparameter Optimisation of Support Vector Machine:

Related Work Hyperparameter Optimization for SVMs Hyperparameter Optimization for SVM [6] is crucial for applications such as electricity load forecasting. SVM is effective at dealing with data sets that do not have a linear structure, however in the paper, it is dependent upon the appropriate adjustment of the hyperparameters. Methods such as grid and random search are perhaps more the norm but inappropriate for large complicated data sets. Metaheuristic methods such as Differential Evolution provide a more viable alternative through optimization over hyperparameters. DE has been proven to be especially useful in enhancing SVM's performance for applications that are regression and forecasting. Since it is an adaptive modification of DE, the Adaptive Differential Evolution (ADE) builds upon DE with real-time adjustment of control parameters to improve convergence speed and accuracy. Therefore, ADE made for a very efficient approach in fine-tuning SVM for dynamic problems like electricity load forecasting, wherein precise short-term predictions amount to great importance for grid stability and cost efficiency. Studies have proven that SVM combined with ADE increases the accuracy highly more than traditional methodologies for optimization. It is a very popular and effective method to further improve the models of machine learning, in particular, when the data contains complexity and heterogeneity.

[5] Traditional MNL models used for WTMC, base upon socio-economic variables. These however cannot impact the intricacy involved in travel behavior as they work under linearity assumptions. On the contrary, SVM, KNN, and Decision Trees have shown to increase the accuracy level based on non-linear relationship modeling and far larger datasets. Hyperparameter tuning has become the problem of all these models as it is usually done manually and time consuming. Toward this end, BO has been adopted in automating tuning to improve the model and its accuracy, especially for models concerned with the minority classes. Optimized ML models with BO hold high promise for informing transportation policy and improving urban transport systems.

A research paper on hyperparameter optimization of CNNs [7] explains the structure and elements of Convolutional Neural Networks. In doing so, it also explains how the various layers such as convolutional, pooling, and fully connected layers work in tandem. Other hyperparameters like the size of the kernel, stride, activation functions, and regularizing techniques are brought into focus to a network. Each one of these plays a significant role in the performance of the network while balancing feature extraction and computational efficiency. It also discusses different metaheuristic algorithms such as GA, PSO, and SA that have been applied with various CNN hyperparameters for hyperparameter tuning in comparison to the traditional grid and random search advantages and limitations. The sequential model-based optimization techniques are also highlighted and particularly Bayesian Optimization, which improves efficiency by learning from past results. In this paper, other numerical optimization approaches will be discussed, for example, Stochastic Gradient Descent (SGD), which lags behind because of drawbacks like local minima and slow convergence. Finally, a comparison will be made of different hyperparameter optimization methods in terms of their performance on certain datasets, for example, MNIST, and CIFAR-10. It further identifies challenges in CNN HPO, mainly having "relatively high computational costs" and expounds on future directions in HPO research that mainly comprise AutoML and reinforcement learning to fortify optimization techniques.

A critical area of machine learning (ML) in optimising model performance, generalisation, and efficiency is hyperparameter tuning [2]. Parameters like the learning rate and model architecture directly impact the results and therefore, their optimization is crucial. Even everyday approaches such as grid search and random search are available, but advanced methods like Bayesian optimization save much more computational resources and even allow balancing exploration and exploitation. Challenges arise in managing computational costs, risk of overfitting, and ensuring that the models and datasets grow with scalability. Interpretability is also necessary as the comprehension of hyperparameter effects can be a factor for improving trust in ML systems. Future research may attempt to increase the efficiency of tuning by possibly using techniques such as NAS, as well as better strategies for resource allocation to various parts of large-scale models.

Methodology

The methodology aims at evaluating the performance of Hyperparameter optimization techniques in improving different machine learning models used for Handwritten digit recognition. The study entails several models in action including Decision tree, SVM, KNN and Random Forest with and without hyperparameter optimization.

Data Collection:

The dataset of interest in this study is sourced from Kaggle which contains the pixel values of the handwritten digits (0 9) in gray scale. Each image measures 28x28 pixels mostly black and white and the pixel values vary from 0 for white to 255 for black where the degree of blackness increases. The training set covers 785 columns: one labelled as ‘label’ shows the digit which has been written and 784 with respect to the pixel values. Since we take dataset from Kaggle, it is of a pre processed nature.

Hyperparameter Optimization:

A procedure that involves the identification of most appropriate and adequate parameters which will help enhance the given model performance. Improvement of a model requires optimization of the solution deployed. In this research, we decompose hyperparameter optimization as applied computational methods in machine learning, using Random Search, and Bayesian Optimization approach.

A) Random Search:
Random search randomly samples hyperparameters within specified limits in search of best hyperparameter combination.

B) Bayesian Optimization:
However, it tends to be a more improved process as it takes advantage of what has been done before to improve on the selection of hyperparameters.

Machine Learning Models:
Four machine learning models were employed: Decision Tree, K-Nearest Neighbors (KNN) and support vector machine and Random Forest.

1.DecisionTree:
For the task of handwritten digit recognition, a decision tree classifier is adopted using the Scikit-learn python library. In the first process we assess accuracy of the model using default parameters. In the second process, we evaluate the model with ten compositions of hyperparameters.

Several key parameters can be highlighted:

- Criterion: This refers to the function that can be used to tutor the quality of the split functions like Gini impurity or Information Gain or the information gain.
- Max-depth: This parameter specifies the limit on the number of levels the tree can have.
- Min_sample_split: This parameter specifies the minimum sample number needed for an internal node to be split.
- Min_sample_leaf: Specify the minimum number of samples to be present in a leaf node.
- Max_features: maximum number of such features in a candidate set, when the best split is sought.

For these different sets of parameters tested on the model, we also perform Random search and Bayesian Optimization technique for the most probable hyperparameters on smaller subset of data. Subsequently, the parameters derived from the optimization processes performed on the reduced dataset are utilized to carry out the analysis on the entire dataset. The accuracy of the model is measured again enabling determination of the improvement of the model's performance as a result of hyperparameter tuning.

2. Support Vector Machine (SVM):
The paper further discusses techniques in SVM of machine learning specifically in regard to How Important is Hyperparameter Tuning. In our first test we hit a great 97% accuracy without any hyperparameter optimization. Once the hyperparameter tuning had been applied, we observed that with high usage of resources and high run time of the model, improvements were only minor if any enhancement in accuracy was sought after. This work considers, however problems of performance versus computational efficiency as far as the possible benefits of hyperparameter optimization are concerned. It becomes apparent when it makes more sense to optimize for performance, rather than computational efficiency, and how the results of the paper give recommendations in what parts of practice.

Some of the hyperparameters chosen for optimization are as follows:

- C (Regularization Parameter): This controls how much to regularize, i.e., the trade-off between maximizing the margin and minimizing classification error. Larger C tries to classify all training examples correctly, which may result in overfitting.
- Kernel: The kernel function is that which determines how SVM transforms the input data to higher dimensions for better classification
- Gamma (γ): It is critical for digit recognition to adjust gamma for model complexity control.

- Degree: If a polynomial kernel is being used, the degree of the polynomial defined is the value of this parameter

- Max Iterations: It defines the maximum number of iterations that the algorithm runs.

3.K-Nearest Neighbors (KNN):
The K-Nearest Neighbors algorithm is used for predicting and classifying. The classification is based on the labels of nearest neighbors. The KNN is used experimentally in two stages: baseline model and optimized model. The baseline model is trained based on default hyperparameters. In the optimized model, the hyperparameter is optimized to enhance its performance. It employs Random Search and Bayesian Optimization to find the optimal hyperparameters that increase the classification accuracy.

For the selected optimization, hyperparameters are

- 'n_neighbors' or Number of Neighbors: It defines the number of nearest neighbors to be chosen to classify. This is typically settled using cross validation. Typical values range from k = 3 up to k = 10.

- 'P' (Distance metric): The KNN approach uses the distance between data points to find the nearest neighbors. This parameter allows variations in distance metrics; with `p = 1` this indicates Manhattan distance, and `p = 2` Euclidean distance.

- 'weights` (Weight Function): This hyperparameter controls whether allneighbors contribute equally to the prediction or whether closer neighbors should have more influence. can either be set to 'uniform' or 'distance'.Uniform: All neighbors are assigned equal weight. Distance based: The nearer ones get greater weight, while distant ones count very less.

First, Random Search is performed to search hyperparameter combinations by trying various values for 'n_neighbors' between 3 and 10 as well as for 'p' in the distance metric. Bayesian Optimization further refines the model by being more specific in the search for hyperparameters as it uses information gathered from performance due to Random Search, which yields accuracy more finely grained. Now, we will train the KNN model with default hyperparameters in order to set a baseline using default hyperparameters such as n_neighbors = 5, p = 2, and uniform weights. This will be used as a baseline to assess if there is really any significant effect of hyperparameter tuning. Thus, it is trained with the best hyperparameters that have been optimized through Random Search and Bayesian Optimization. The performance was estimated by means of 5-fold cross-validation to ensure robustness without overfitting

4.Random Forest:
In this paper we discovered the use of Random Forest classifier techniques in machine learning model of digit recognition. In First model we use default random forest technique and we achieved 94% accuracy .In our second model we use different parameters to increase accuracy. Then after we applied hyperparameter tuning, From which we got 97% accuracy we were able to observe high resource utilization and run time of the model.

The hyperparameters selected for optimization include:

- n_estimators (Number of Trees): controls number of trees in the forest. If we Increase this value it can reduce variance, but it can also increases computation time.

- max_depth (Tree Depth): It is number of maximum depth of the trees. Deep trees may overfit but Shallow trees may underfit.

- min_samples_split: This controls the minimum number of samples required to split an internal node,This factore affect the complexity of model.

- min_samples_leaf: The minimum number of samples required to be at a leaf node. Higher values prevent overly small leaf nodes, improving generalization. controlling how large or small leaf nodes can be.

III. RESULT AND DISCUSSION

We present the result and analysis from our experiments on handwritten digit recognition, using different machine learning models: Decision Tree, K-Nearest Neighbors (KNN), Support Vector Machine (SVM) and Random Forest. The performance is evaluated with and without using hyperparameter optimization.

Table.1 presents the accuracy performance of various machine learning models without optimization. Figure 1 shows a graph illustrating the accuracy trends of these models before any optimization techniques were applied. Subsequently, Figure 2 highlights the accuracy improvements achieved when using random search optimization across different models. Finally, Figure 3 displays the accuracy performance of these models after applying Bayesian optimization, showcasing further enhancements in model accuracy through this more refined optimization method. These figures provide a comprehensive comparison of model performances with and without optimization.

Table no. 1 Comparison of Model Performance

Model Name	Without Hyperparameter	With Best Hyperparameter (among 10 trials)	Random Search Optimization	Bayesian Optimization
Decision Tree	85%	82%	87%	87%
KNN	96%	97%	97%	96%
Random Forest	96%	97%	94%	95%
SVM	97%	Computationally high	Computationally high	Computationally high

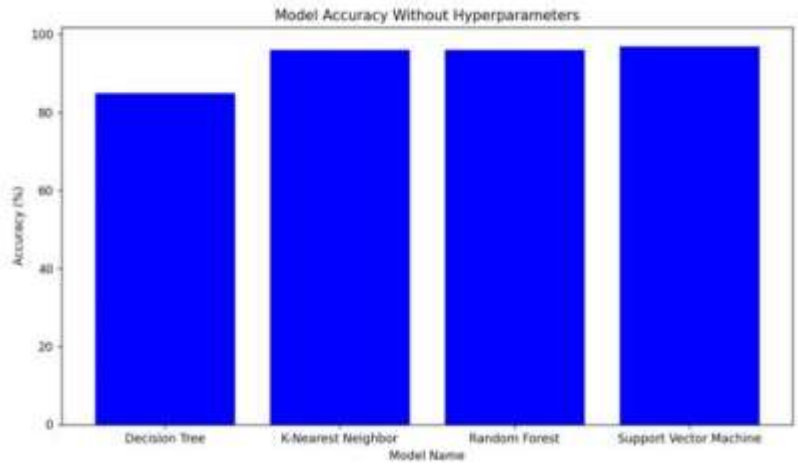

Figure No. 1 Model Accuracy Without Hyperparameter Optimization

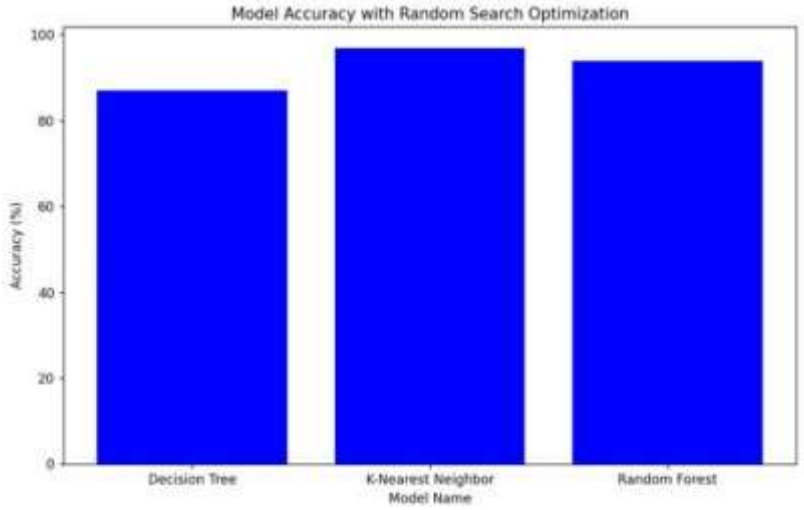

Figure no.2 Model Accuracy with Hyperparameter Optimization (Random Search)

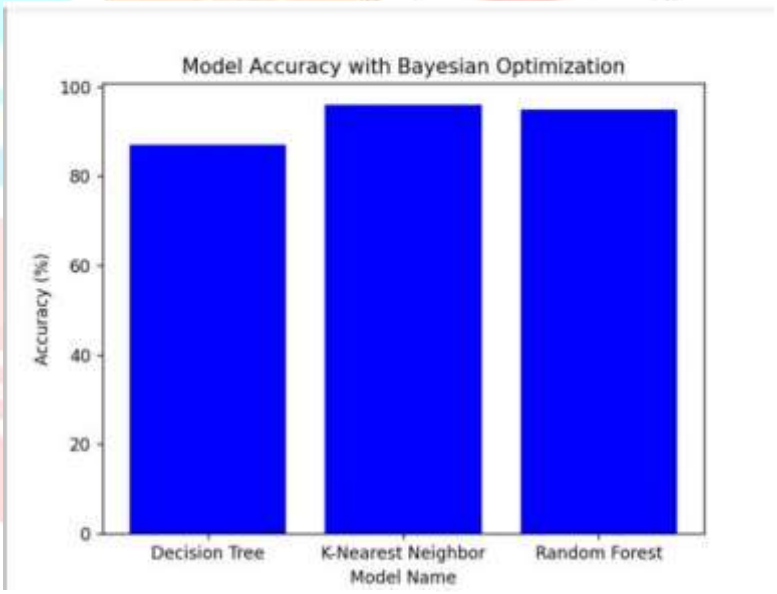

Figure no.3 Model Accuracy with Hyperparameter Optimization (Bayesian Optimization)

Table no.2 Accuracy for 10 different parameters in decision tree

criterion	max_depth	min_samples_split	min_samples_leaf	max_features	accuracy
Gini	10.0	2	1	Sqrt	0.7791
Gini	10.0	2	2	Sqrt	0.7732
Gini	20.0	5	1	Sqrt	0.8108
Gini	20.0	5	2	Sqrt	0.8076
Entropy	10.0	2	1	log2	0.7091
Entropy	10.0	2	3	log2	0.7197
Entropy	15.0	4	1	log2	0.7658
Entropy	15.0	4	2	Sqrt	0.8163
Gini	NaN	10	1	log2	0.7648
Entropy	NaN	10	3	Sqrt	0.8200

Table no. 3 Accuracy for 8 different parameters in random forest

N_estimator	Max_depth	Min_samples_split	Min_samples_leaf	Accuracy
50	10	2	1	0.94
100	15	3	2	0.96
150	20	4	3	0.96
200	25	5	4	0.96
200	None	2	2	0.96
100	None	10	5	0.96
300	30	3	2	0.96
400	35	4	3	0.96

Table no.4 Accuracy for 10 different parameters in KNN

n_neighbors	weights	p	accuracy
3	uniform	2	0.9651
4	distance	1	0.9595
5	uniform	2	0.9640
6	distance	2	0.9655
7	uniform	2	0.9639
8	distance	1	0.9586
9	distance	1	0.9561
17	distance	2	0.9590
10	distance	1	0.9568
6	uniform	2	0.9631

This result indicates that hyperparameter optimization plays a crucial role in improving the performance of machine learning models.

IV. Conclusion

In short, hyperparameter tuning is applied to enhance the performance of a machine learning model like Decision Tree, KNN, and Random Forest. On the other hand, complex models like SVM have higher accuracy by default, especially in tasks like handwritten digit recognition. The accuracy achieved by SVM is 97% without any hyperparameter optimization. But, applications of optimisation techniques like Random Search and Bayesian Optimization increase the computation cost with SVM. Whereas in decision tree model, accuracy was 85% without applying Hyperparameter optimization. If the hyper parameter optimizations are used with the decision tree model then its accuracy came around 87%. And KNN model which did not use hyper parameter optimization had the result accuracy around 96%. In the case of optimization of KNN model using Random Search hyperparameter optimization its accuracy gets improved by 1%, that is, it became 97%. Although the model seems simpler, optimization helps in making it achieve competitive performance. As case in the random search model the accuracy without hyperparameter optimization was 96% and by using hyperparameter optimization got the accuracy of 97%.

REFERENCES

[1] Albert Budi VI. References Efficient Christian "Accuracy-Time Hyperparameter Optimization Using Actor-Critic-based Reinforcement Learning and Early Stopping in OpenAI Gym Environment."

[2] Justus Akinlolu llemobayo "Hyperparamtere Tunning in Machine Learning: A Comprehensive Review"june

[3] Bernd Bischl "Hyperparameter Optimization: Foundations, AOptimization: Foundations, Algorithms, Bgorithms, Best Pst Practicesactices and Opnd Challenges."Deepalienges."

[4] Deepali Bassi "A Comparative Study on Hyperparameter Optimization Methods Prediction."2021 in Software Vulnerability

[5] Mahdi aghaabbasi "On Hyperparameter Optimization of Machine Learning Methods Using a Bayesian Optimization Algorithm to Predict Work Travel Mode Choice."

[6] M. Zulfiqar "Hyperparameter optimization of support vector machine using adaptive differential evolution for electricity load forecasting"

[7] Mohaimenul Azam Khan Raiaan"A systematic review of hyperparameter optimization techniques in Convolutional Neural Networks"

[8] Xueli Xiao "Efficient hyperparameter optimization in deep learning using a variable length genetic algorithm a preprint"june