A Comprehensive Benchmark of State-of-the-Art Hyperparameter Optimization Frameworks for Convolutional Neural Networks

[1]N. Latha, [2]T. Divya, [3]Dr. Poornima Devan, [4]Radha mam,
[1,2]Assistant Professor, Department of CSE, Arasu Engineering College, Kumbakonam, India,
[3]Assistant Professor, Department of CSE, Sathyabama Institute of Science and Technology, Chennai - 600119.
[4]Radha R, Assistant Professor, MCA Department, Ethiraj College for Women (Autonomous), Chennai, India.

latha.sakthi2000@gmail.com[1], divyathiru777@gmail.com[2], poorniramesh2011@gmail.com[3], radha_r@ethirajcollege.edu.in[4]

Abstract—Convolutional Neural Networks (CNNs) have emerged as the backbone of many high-performance machine learning models, driving advances in image recognition, object detection, and beyond. However, their success is critically dependent on the fine-tuning of hyperparameters, a process that often demands significant time and computational resources. Selecting the right hyperparameters, such as learning rates, batch sizes, and network architectures, can significantly impact model convergence, generalization, and overall performance. This paper offers a detailed and comprehensive benchmark of leading Hyperparameter Optimization (HPO) frameworks tailored for CNNs, with a focus on balancing accuracy, computational efficiency, and ease of use.

We explore and rigorously compare a diverse range of state-of-the-art optimization techniques, including traditional approaches like Grid Search and Random Search, advanced probabilistic methods such as Bayesian Optimization, and emerging resource-aware algorithms like Hyperband and the Tree-structured Parzen Estimator (TPE). We evaluate each approach using diverse CNN architectures and datasets, analyzing critical factors such as optimization speed, resource efficiency, and model accuracy. Through this comparative study, we reveal unexpected strengths and weaknesses of each framework, offering new insights into the subtle trade-offs between exploration and exploitation, computational overhead, and scalability. Our findings provide a fresh perspective on optimizing CNN performance, guiding practitioners in selecting the most effective HPO strategy for specific applications, and paving the way for more intelligent, adaptive deep learning models.

Keywords: *optimization, hyperparameters, machine learning, deep learning, learning rate, batch size, architecture and kernel size.*

1. Introduction

There has been a revolution in the field of machine learning brought about by convolutional neural networks (CNNs), particularly in areas such as image categorization, object detection, and video analysis. CNNs are a useful tool for extracting complicated patterns from high-dimensional data sources due to their hierarchical architecture, which was designed to automatically learn spatial hierarchies of information. The efficacy of CNNs is largely dependent on the appropriate selection of hyperparameters. These hyperparameters include learning rate, batch size, and the number of layers. Each of these hyperparameters plays a key role in determining the convergence, accuracy, and cognitive efficiency of the network. Practitioners have typically relied on trial-and-error methods to determine the ideal configurations for hyperparameter tuning, which has generally been a manual procedure that is frequently arbitrary. For this reason, efficient hyperparameter optimization (HPO) has arisen as a crucial topic of research as CNN architectures continue to get more complicated and are implemented in situations with limited resources. While attempting to strike a balance between the demand for high model performance and the constraints of computational resources, the issue lies in navigating the enormous hyperparameter search space [1,2].

Using complex search algorithms that intelligently explore and exploit the hyperparameter space, hyperparameter optimization frameworks that are considered to be state-of-the-art try to automate this process. In spite of their widespread application, traditional methods such as Grid Search and Random Search are frequently ineffective because of the exhaustive or solely stochastic character of their methodologies [3]. Bayesian Optimization, Hyperband, and the Tree-structured Parzen Estimator (TPE) are examples of more sophisticated techniques that utilize probabilistic models, resource-aware mechanisms, and adaptive algorithms to optimize hyperparameters with less iteration and less computer overhead. Bayesian Optimization is a methodology that was developed in the 1990s. These frameworks not only improve the accuracy of models, but they also reduce the amount of time it takes for models to converge and to scale, making them extremely useful for contemporary deep learning processes [4].

There are not enough extensive studies that compare the performance of HPO frameworks across a variety of CNN architectures and real-world datasets, despite the fact that HPO frameworks are becoming increasingly popular. There is a lack of assistance for practitioners regarding which framework is best suitable for their particular requirements because the

majority of the research that is currently available either concentrates on a single HPO approach or analyzes performance on a limited selection of activities. In addition, the trade-offs between speed, resource utilization, and model accuracy critical factors in practical applications of machine learning [5].

By presenting a comprehensive assessment of the most advanced HPO frameworks, with a particular emphasis on how these frameworks might be used to CNNs, the purpose of this research is to fill in these gaps. We evaluate classic and new HPO techniques in a methodical manner, evaluating them across a variety of dimensions, such as the accuracy of the model, the amount of time it takes for the model to converge, the amount of computational resources employed, and the scalability of the system [6]. We intend to provide a comprehensive evaluation that will assist academics and practitioners in selecting the HPO framework that is best suitable for their needs. This will be accomplished by utilizing a number of CNN architectures, including ResNet, VGG, and Inception, and testing on real-world datasets that range in size and complexity [7].

Our investigation of the underlying mechanisms that are responsible for the effectiveness of these frameworks is in addition to the evaluation of their performance. In this study, we investigate the various approaches that are utilized to control the exploration-exploitation trade-off, optimize resource allocation, and scale to larger datasets and longer networks. This study offers useful insights into how HPO frameworks can be customized to unique CNN-based applications, which ultimately leads to more efficient and successful model optimization processes. These insights are provided by offering a balanced assessment of the strengths and limits of these frameworks [8].

The remaining sections of the paper are structured as follows: In the second section, an overview of HPO and its methods is presented. In Section 3, the hyper parameters optimization techniques are discussed. In Section 4, the HPO frameworks, including Bayesian Optimization, Optuna, HyperOpt, and Keras Tuner, are broken down and discussed. The different performance evaluations are discussed in Section 5. The conclusion of the paper is found in Section 6.

2. Hyperparameter Optimizations

The HPO process is an essential method for selecting the optimal combinations of hyperparameters that will lead to the highest possible performance for machine learning

models. There have been a number of automatic HPO algorithms created in order to tune the hyperparameters in order to design efficient machine learning models. First, we will discuss the standard hyperparameters, and then we will move on to the HPO approaches.

2.1. Hyperparameters

The variables of a model that are responsible for determining its appearance and behavior are called hyperparameters. In order to address the computational complexity of the models, hyperparameters have recently garnered a significant amount of attention. Every machine learning and deep learning algorithm has its own unique set of hyperparameters that need to be adjusted during the tuning process as shown in Figure 1. Numerous studies provide a comprehensive explanation of the hyperparameters. As far as hyperparameters are concerned, there are two primary categories: model hyperparameters and optimizer hyperparameters [9].

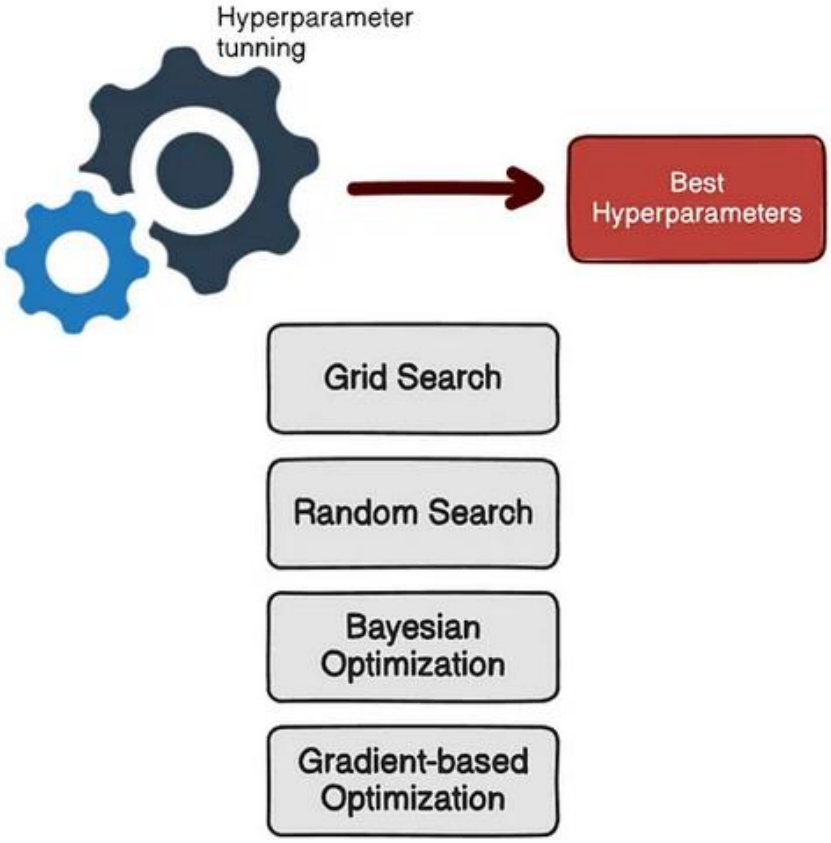

Fig. 1 Process of hyper parameters optimization

While the training optimization algorithms are the optimizer hyperparameters, the model hyperparameters are responsible for designing the structure of the model. A type of model-specific hyperparameter known as the activation function is responsible for producing a robust perception of important and non-linear complicated functions that are used to convert

the input signals. Sigmoid, Softmax, Tanh, and Rectified Linear Units (ReLU) are examples of types of activation functions that are frequently utilized [10]. It is recommended that the ReLU activation function be used as the default activation function since it offers a solution to the vanishing gradient problem and converges six times more quickly than the Tanh function.

A hyperparameter that is well-known for its ability to quantify the rate and speed at which networks learn is referred to as the learning rate (LR). The weights of the hidden layers, which are the location where the overall structure of networks that directly extract complicated information are represented, can be adjusted with the use of LR [11]. For the purpose of ensuring that the local minimum is not overlooked, it is necessary to choose the optimal LR values. LR annealing is a method that was established not too long ago in order to locate the optimal value during the training process. However, the majority of the times, the users are the ones that manually adjust the LR settings. Creating fake copies of the training data in order to carry out particular functions such as transformation, rotation, cropping, and so on is an example of the data production approach known as data augmentation. The neural networks are able to improve their performance during the learning process because to this [12].

There is a significant amount of importance placed on optimization techniques in the process of tuning neural networks (NNs). They make it possible for the networks to learn and improve their performance. Adaptive momentum estimation (Adam), Root Mean Square Propagation (RMSprop), and Gradient Descent (GD) are three optimizers that are frequently utilized in deep learning [13]. The Gradient Descent algorithm ensures that the cost function is minimized by continuously updating the parameters of the models until they reach the standard bias values at each stage. The gradient is adjusted in an iterative manner in order to obtain the local minima, which is accomplished by updating the weights and bias. The RMSprop algorithm optimizes the gradient by creating a balance between the momentum (step size) and decreasing the step size for gradients that are very large [14]. The Adaptive Gradient Descent (AdaGrad) and RMSprop optimizers are combined into Adam, which is another state-of-the-art optimizer. Adam integrates the two optimizers. Additionally, it serves as the default optimization technique for the training process, and it computes the adaptive LR for each parameter [15].

3. Hyperparameters Optimization Techniques

An overview of the HPO approaches, including grid search, random search, Genetic Algorithm (GA), Bayesian Optimization (BO) and Tree Parzen Estimator (TPE) is provided in this section. As shown in Figure 2, the hyper parameters optimization techniques can be classified into five types.

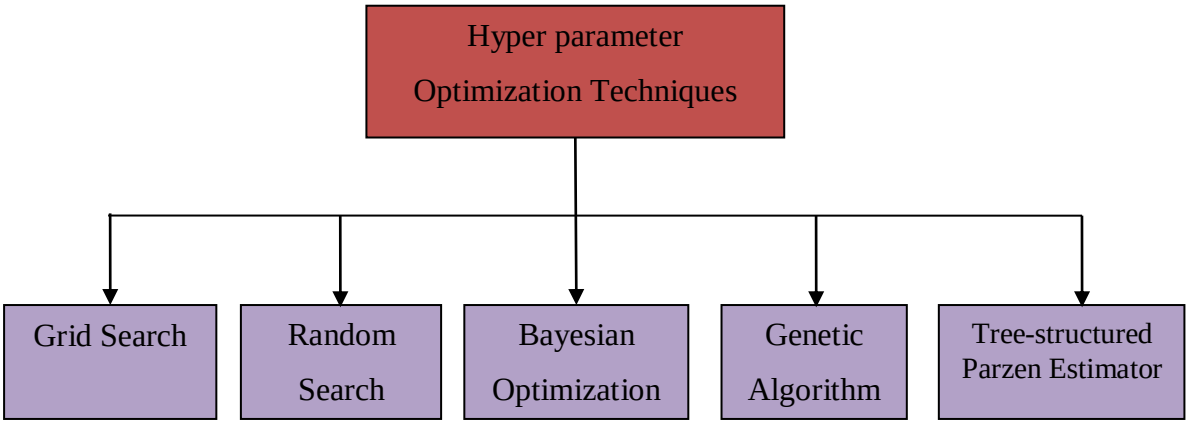

Fig. 2 Categories of hyper parameters optimization

3.1.1. Grid Search (GS)

The grid search algorithm is one of the most straightforward approaches of optimizing hyperparameters. A grid of hyperparameter values must be specified, and the model must be evaluated for each and every conceivable combination. The process can be formalized as follows:

- It is necessary to define the hyperparameters and the ranges that correspond to them.
- It is necessary to generate a Cartesian product of all possible hyperparameter combinations.
- Cross-validation or a hold-out validation set can be used to train and evaluate the model for each particular combination.
- Select the combination yielding the best performance metric (e.g., accuracy, F1-score).

The number of possible combinations increases at an exponential rate in proportion to the number of hyperparameters and the values that are assigned to them, which results in expensive computation times. The significance of various hyperparameters is not taken into account by it; parameters with less significance may result in the consumption of resources that are not necessary [16].

3.1.2. Random Search (RS)

The random research technique has been empirically shown to outperform grid search, particularly in situations where only a subset of hyperparameters significantly impacts model performance. This is because it can yield high-quality hyperparameter configurations with a significantly smaller number of evaluations, which makes it particularly useful in high-dimensional search spaces where optimal configurations may be sparsely located. Because of this, Random Search is a more efficient alternative to grid search. It does this by sampling hyperparameter values randomly from defined distributions rather than exhaustively evaluating every possible combination [17].

3.1.3. Bayesian Optimization (BO)

Bayesian optimization is a popular and advanced method that models the performance of hyperparameter configurations over time using a probabilistic surrogate model, often a Gaussian Process (GP). Bayesian optimization uses previous evaluation results to predict which hyperparameters to evaluate next, using an acquisition function like Expected Improvement or Upper Confidence Bound. This method finds an appropriate balance between exploring new hyperparameter regions and taking advantage of known promising configurations, and it converges to optimal settings with much fewer evaluations than traditional methods [18].

3.1.4. Genetic Algorithm (GA)

Genetic algorithms (GAs) are inspired by the principles of natural selection and evolutionary biology, evolving a population of hyperparameter configurations over generations [19]. The process generally follows these steps:

- Initialize a random population of hyperparameter configurations.
- Evaluate the performance of each configuration.
- Select configurations based on fitness scores for reproduction, applying crossover (mixing configurations) and mutation (randomly altering configurations).
- Repeat the evaluation and selection process over multiple generations.

3.1.5. Tree-structured Parzen Estimator (TPE)

The Tree-structured Parzen Estimator (TPE) is an additional advanced Bayesian optimization technique that models the probability distributions of hyperparameter configurations. It uses these distributions to differentiate between configurations that produce good and bad performance outcomes. TPE constructs separate models for the likelihood of observing good

and bad configurations and chooses the next hyperparameter set for evaluation based on these distributions. This allows it to concentrate the search on promising regions of the hyperparameter space, which significantly improves the efficiency of the optimization process in comparison to more conventional methods such as grid search [20].

4. Hyperparameters Framework

The automatic tools that are used to modify the hyperparameters of machine learning models are known as hyperparameter optimization frameworks. In most cases, each tool comes with a collection of optimization strategies as well as an intuitive user interface that allows for the definition of search space, the evaluation of objective functions, and the monitoring of the performance of mathematical models. In order to solve difficult machine learning challenges, these frameworks are in high demand across the industry. As shown in Figure 3, the following HPO frameworks have been utilized such as Optuna, HyperOpt, Ray tuner, OptiML and Keras tuner.

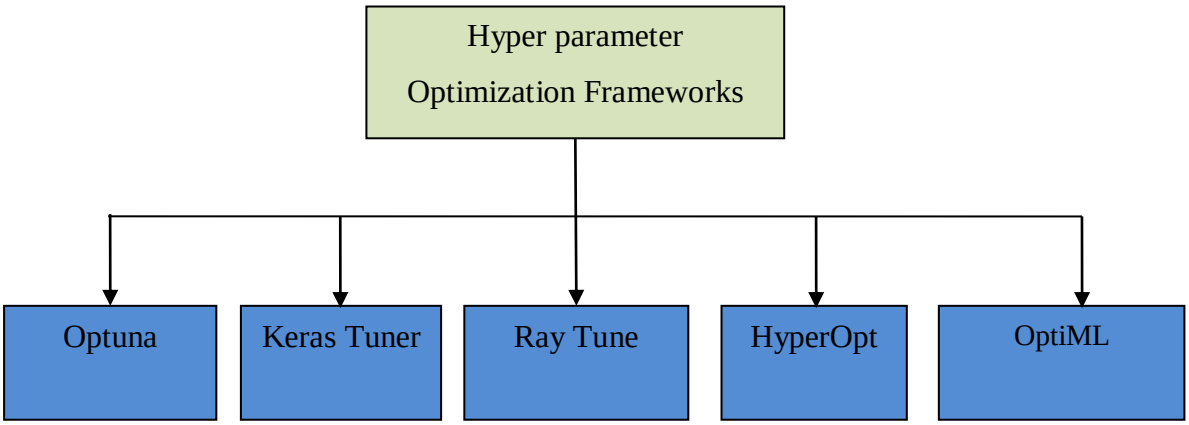

Fig. 3 Categories of hyper parameters frameworks

The general Convolutional Neural Network (CNN) based hyper parameters optimization has been shown in Figure 4.

4.1. Optuna

Optuna is a framework for hyperparameter optimization that is both flexible and efficient. It makes use of an automatic optimization technique that is based on tree-structured Parzen estimators (TPE) in order to intelligently suggest hyperparameter configurations. It is especially useful for complex projects that require fine-tuned models since it provides features such as visualization tools that assist users in analyzing the optimization process and a user-friendly interface that enables easy connection with current machine learning libraries. These features make it particularly relevant for identify the best parameters [21].

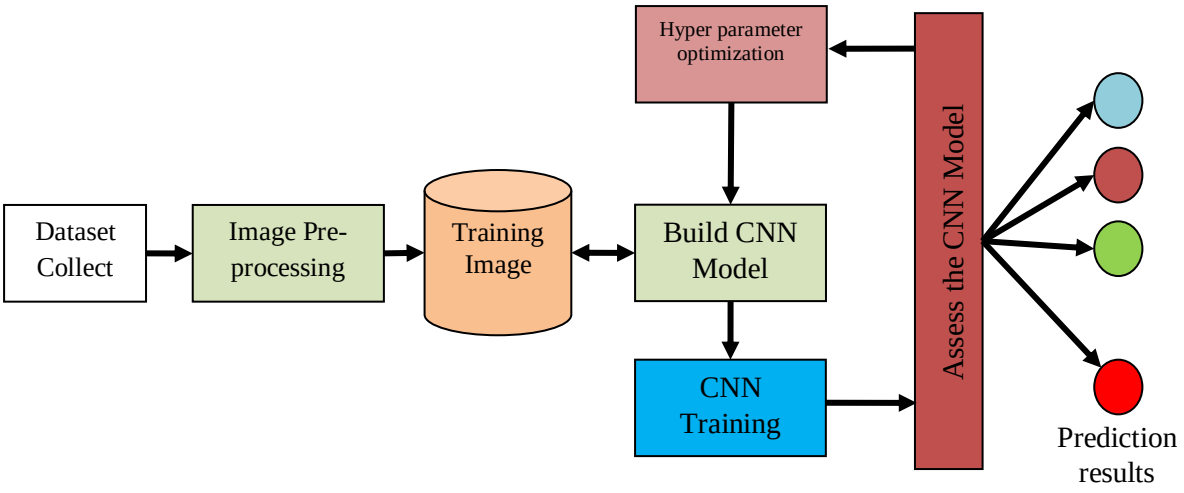

Fig. 4 General architecture of CNN based hyper parameters optimization

4.2. Keras Tuner

Keras Tuner is a library that was developed exclusively for the purpose of tuning hyperparameters in Keras models. It is a straightforward and user-friendly tool. It makes use of a variety of optimization techniques, such as Random Search and Bayesian Optimization, and provides straightforward application programming interfaces (APIs) for defining hyperparameter search spaces. In addition to providing functions such as early pausing and learning rate scheduling, Keras Tuner interacts smoothly with Keras, making it possible for users to easily tune their models with little setup [22].

4.3. Ray Tune

Ray Tune is a hyperparameter tuning library that is developed for distributed computing and machine learning. It is a scalable hyperparameter tuning library that is a component of the Ray ecosystem. With Ray Tune's support for a variety of search methods, such as random search, Bayesian optimization, and Hyperband, users are able to explore hyperparameter spaces in parallel across numerous nodes in an effective manner. As a result of its capacity to manage large-scale machine learning processes and its seamless integration with well-known deep learning frameworks like TensorFlow and PyTorch, it is a good option for applications that require a considerable amount of processing power [23].

4.4. HyperOpt

Another library that is extensively used for hyperparameter optimization is called Hyperopt. It employs Bayesian optimization and TPE methods in order to search for the optimal

hyperparameters. In addition to supporting single-objective and multi-objective optimization jobs, it supplies users with the ability to create complex search spaces through the use of a clear syntax. The versatility of Hyperopt in terms of defining the search space, in conjunction with its capacity to operate in distributed contexts, makes it suited for a wide range of machine learning applications [24].

4.5. OptiML

In order to navigate huge search spaces in an effective manner, OptiML is a hyperparameter optimization system that makes use of multi-fidelity optimization strategy. OptiML is able to lower the overall computational cost associated with hyperparameter tweaking while retaining a high level of accuracy in identifying optimal configurations. This is accomplished by utilizing surrogate models to generate predictions about the performance of the model at certain resource levels (for example, training epochs) [25].

5. Performance Metrics

Evaluation metrics plays an important role for accessing the classification performance and improving model selection. We have used confusion matrices, accuracy, precision, recall and F1-score for evaluating the effectiveness of the proposed approach. TP, FP, TN, and FN class values are shown in Figure 5 for an classification and prediction system. If the classifier correctly predicts the class response at each instance, it is considered a "success," but if it does not, it is considered an "error." The error rate, which is a proportion of the errors made over the entire set of samples, is used to determine the classifier's overall performance.

	Positive	Negative
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negative (FN)	True Negative (TN)

Fig. 5 Confusion matrix process

For the purpose of evaluating the effectiveness of classification algorithms, it is feasible to derive statistical metrics such as Precision, Recall, and F-measure from the confusion matrix. These metrics are described as follows: Precision (P), which is often referred to as detection rate, is the ratio of instances that have been successfully labelled to the total number of

instances that have been labelled. In a certain class, it refers to the proportion of positive forecasts that were accurate. It is defined in the following manner:

$$\text{Precision (P)} = \frac{TP}{TP + FP} \tag{4}$$

The number of true positives is denoted by TP, and the number of false negatives is denoted by FN for a particular class. Total number of test samples for a particular class is equal to the sum of test samples and blanks. Whether it is recall (R) or sensitivity, the ratio of correctly classified images to the total number of instances in a class is referred to as recall. In addition to this, it is also referred to as the genuine positive rate, and it holds the capability of measuring the prediction model. The following is the definition of it:

$$\text{Recall (R)} = \frac{TP}{TP + FN} \tag{5}$$

For a particular class, the number of true positives is denoted by TP, while the number of false negatives is determined by FN. The total number of test samples for a certain class is equal to, TP plus FP. By calculating the harmonic mean of precision and recall, the F1-measure makes an effort to provide a single measurement of performance. It is possible for a classification algorithm to obtain high levels of both recall and precision. Consider the following explanation of the F1-measure:

$$F1 = 2.\frac{P.R}{P + R} \tag{6}$$

The Accuracy can be defined as below:

$$\text{Accuracy} = \frac{\text{TP+FP}}{\text{TP+FP+FN+TN}} \tag{7}$$

K-Fold Validation

The objective of K-Fold validation is to achieve the capability of simultaneously training and validating all of the images that are contained inside the model. There is a type of cross-validation known as k-fold cross-validation, which includes iterating a set of data k times. At the beginning of each round, we divide the dataset into k parts: one of these parts is utilized for validation, and the remaining k-1 parts are combined into a training subset for the purpose of evaluating the model, as visualized in Figure 6.

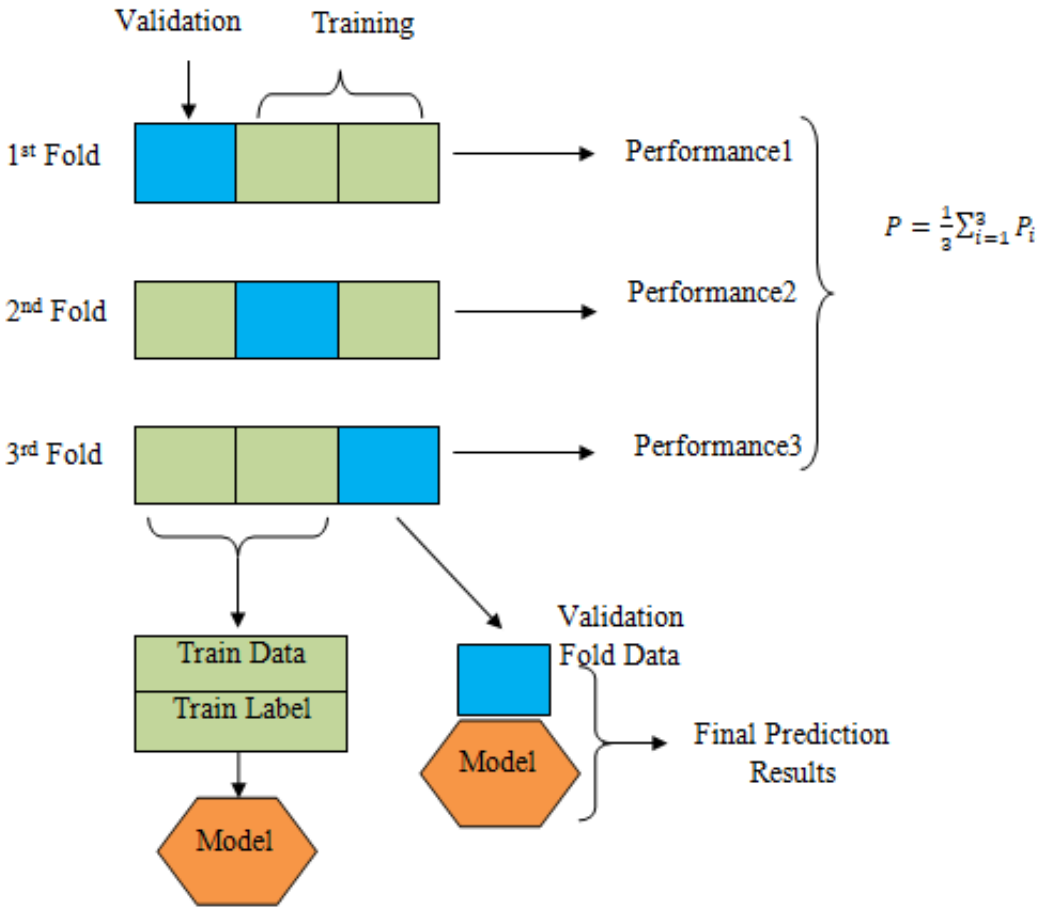

Fig. 6 The concepts of K-Fold cross validation

6. Conclusion

In this paper, a rigorous and nuanced evaluation of Hyperparameter Optimization (HPO) frameworks that have been specifically customized for Convolutional Neural Networks (CNNs) is presented. The study also highlights crucial trade-offs in terms of accuracy, computational efficiency, and usability. The strengths, limitations, and ideal applications of each framework are revealed through the examination of a variety of optimization techniques. These techniques include the traditional Grid and Random Search methods, as well as more advanced Bayesian Optimization and resource-efficient algorithms such as Hyperband and Tree-structured Parzen Estimator (TPE). The results of our research indicate that selecting the most appropriate HPO approach can significantly improve both the performance of the model and its computational feasibility. This has the potential to direct practitioners towards making more informed decisions when adjusting CNNs. These findings not only contribute to a more in-depth comprehension of HPO techniques, but they also pave the way for future breakthroughs in CNN models that are flexible, efficient, and high-performing.

7. References

1. S. Abreu, “Automated Architecture Design for Deep Neural Networks,” ArXiv, 2019.
2. P.Deepan and L.R. Sudha, “Object Classification of Remote Sensing Image Using Deep Convolutional Neural Network”, The Cognitive Approach in Cloud Computing and Internet of Things Technologies for Surveillance Tracking Systems, pp.107-120, 2020. https://doi.org/10.1016/B978-0-12-816385-6.00008-8
3. J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for Hyper-Parameter Optimization,” in Advances in Neural Information Processing Systems, Granada, Spain, vol. 24, pp. 2546-2554, 2011.
4. P. Deepan and L.R. Sudha, “Fusion of Deep Learning Models for Improving Classification Accuracy of Remote Sensing Images”, Journal of Mechanics of Continua and Mathematical Sciences, Vol.14, pp.189-201, 2019, ISSN: 2454-7190.
5. P. Probst, A.-L. Boulesteix, and B. Bischl, “Tunability: Importance of Hyperparameters of Machine Learning Algorithms,” J. Mach. Learn. Res., vol. 20, no. 53, pp. 1−32, 2019.
6. P.Deepan and L.R. Sudha, “Remote Sensing Image Scene Classification using Dilated Convolutional Neural Networks”, International Journal of Emerging Trends in Engineering Research, Vol. 8, No.7, pp.3622-3630, 2020, ISSN: 2347-3983.
7. H. J. P. Weerts, A. C. Mueller, and J. Vanschoren, “Importance of Tuning Hyperparameters of Machine Learning Algorithms.” arXiv, Jul. 15, 2020.
8. P.Deepan and L.R. Sudha, “Comparative Analysis of Remote Sensing Images using Various Convolutional Neural Network”, EAI End. Transaction on Cognitive Communications, 2021. ISSN: 2313-4534, doi: 10.4108/eai.11-2-2021.168714.
9. S. Agrawal, “Hyperparameters in Deep Learning,” Medium, May 19, 2021. https://towardsdatascience.com/hyperparametersin-deep-learning-927f7b2084dd (accessed Mar. 18, 2022).
10. Shrestha, A. and A. Mahmood, Review of deep learning algorithms and architectures. IEEE access, 2019. 7: p. 53040-53065.
11. P.Deepan and L.R. Sudha, “Deep Learning and its Applications related to IoT and Computer Vision”, Artificial Intelligence and IoT: Smart Convergence for Eco-friendly Topography, Springer Nature, pp. 223-244, 2021, https://doi.org/10.1007/978-981-33-6400-4_11.

12. Ghantasala, G. P., Sudha, L. R., Priya, T. V., Deepan, P., & Vignesh, R. R. An Efficient Deep Learning Framework for Multimedia Big Data Analytics. Multimedia Computing Systems and Virtual Reality, 99.
13. Xu, Y., et al., Batch normalization with enhanced linear transformation. arXiv preprint arXiv:2011.14150, 2020.
14. Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. Advances in Neural Information Processing Systems.
15. R. Elshawi, M. Maher, and S. Sakr, "Automated Machine Learning: State-of-The-Art and Open Challenges," ArXiv190602287 Cs Stat, Jun. 2019.
16. P.Deepan, L.R. Sudha, K. Kalaivani and J. Ganesh, "Scene Classification of Remotely Sensed Images using Optimized RSISC-16 Net Deep Convolutional Neural Network Model", EAI Endorsed Transactions on Scalable Information Systems, Vol. ,2022, https://doi.org/10.4108/eai.1-2-2022.173292.
17. M.-A. Zöller and M. F. Huber, "Benchmark and Survey of Automated Machine Learning Frameworks," ArXiv190412054 Cs Stat, Jan. 2021.
18. Chen, Y., Huang, A., Wang, Z., Antonoglou, I., Schrittwieser, J., Silver, D., & de Freitas, N. (2018). Bayesian optimization in AlphaGo. arXiv:1812.06855.
19. John H. Holland, "Genetic Algorithms", *Sci. Am.*, vol. 267, no. 1, pp. 66-73, 1992.
20. Watanabe, S., & Hutter, F. (2023). c-TPE: Tree-structured Parzen estimator with inequality constraints for expensive hyperparameter optimization. arXiv:2211.14411.
21. Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In International Conference on Knowledge Discovery & Data Mining.
22. KerasTuner." Keras, Mar. 14, 2022. Accessed: Mar. 18, 2022. [Online]. Available: https://github.com/keras-team/kerastuner.
23. J. Zhang, Q. Wang, and W. Shen, "Hyper-parameter optimization of multiple machine learning algorithms for molecular property prediction using hyperopt library," Chin. J. Chem. Eng., vol. 52, pp. 115–125, Dec. 2022.
24. J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: a Python library for model selection and hyperparameter optimization," Comput. Sci. Discov., vol. 8, no. 1, p. 014008, Jul. 2015.
25. Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", *Proc. IEEE*, vol. 86, no. 11, pp. 2278-2323, 1998.