STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **14**, November 2025, pp 2487–2502.
Published online in International Academic Press (www.IAPress.org)

Automated Initialization Method in Convolutional Neural Networks using BO Applied to Plant Disease Classification

Saloua Lagnaoui [1,*], Khaoula Boumais [2], Zakariae En-naimani [3], Khalid Haddouch [1]

[1] *Laboratory Engineering, Systems and Applications, ENSA University of Sidi Mohamed Ben Abdellah, Fez, Morocco*
[2] *Laboratory AI, Data Science and Emerging Systems, Sidi Mohammed ben Abdallah University, ENSA, Fez, Morocco*
[3] *Laboratory Computer Science, Artificial Intelligence and Cyber Security (2IACS), ENSET, Mohammedia, University of Hassan II Casablanca, Morocco*

Abstract The Convolutional Neural Network (CNN) stands out as the most effective deep learning model for image classification due to its utilization of convolutional kernels for feature extraction. The initialization methods of these kernels significantly impact a CNN performance, with proper initialization leading to faster convergence and improved overall performance, while poor initialization can hinder the learning process. Our paper introduces a novel algorithm called BO-IKM, which leverages Bayesian Optimization to determine the optimal kernel initialization methods for each convolutional layer in a CNN. This systematic approach enhances the models accuracy and precision by identifying the best initializers. We validated the effectiveness of BO-IKM using the Plant Pathology 2020 and Plant Disease Recognition datasets, a challenging image classification problem. The results were compelling, showing significant improvements in accuracy and precision for CNN models optimized with BO-IKM compared to those using standard initialization methods. These findings underscore the potential of BO-IKM to enhance CNN performance across various image classification tasks.

Keywords Convolutional Neural Network, Bayesian Optimization, Initialization Kernels Methods, Plant Disease Classification

DOI: 10.19139/soic-2310-5070-2380

1. Introduction

Agriculture is crucial in the global economy, providing food and raw materials for various industries. However, one of the significant challenges faced by the agricultural sector is the accurate and timely identification of plant diseases. Early detection and precise classification of plant diseases are vital to prevent crop loss and ensure food security [1]. Traditional methods of plant disease detection, which often rely on expert visual inspection, can be time-consuming, labor-intensive, and prone to human error. In recent years, deep learning algorithms have emerged as powerful tools for enhancing precision in plant disease classification. These algorithms can analyze large datasets, learn intricate patterns, and make accurate predictions [2]. Among the various deep learning models, Convolutional Neural Networks stand out due to their exceptional performance in image classification tasks. CNNs have been successfully applied to various fields, including medical imaging, autonomous driving, and facial recognition. In agriculture, CNNs have shown promise in detecting and classifying plant diseases from images of leaves, stems, and fruits[3, 4, 5].

*Correspondence to: Saloua Lagnaoui (Email: saloua.lagnaoui@usmba.ac.ma). Laboratory Engineering, Systems and Applications, ENSA University of Sidi Mohamed Ben Abdellah, Fez, Morocco

ISSN 2310-5070 (online) ISSN 2311-004X (print)

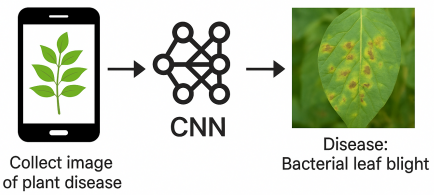


Figure 1. CNN for plant disease classification

CNNs are particularly well-suited for plant disease classification because of their ability to automatically learn and extract features from raw image data. Unlike traditional machine learning methods that require manual feature extraction, CNNs use convolutional layers to identify patterns such as edges, textures and shapes directly from images. This capability enables CNNs to achieve high accuracy in distinguishing between healthy and diseased plants, as well as among different types of diseases [6]. For example, in a study involving the classification of apple leaf diseases, a CNN model achieved remarkable accuracy by learning from a large dataset of apple leaf images. Similarly, CNNs have been used to detect diseases in crops such as wheat, rice, and tomatoes with impressive results. These models not only assist farmers in identifying diseases but also provide valuable insights for taking timely and appropriate remedial actions [7, 8].

Despite their advantages, CNNs come with their own set of challenges. One of the primary issues is the complexity of the model. As CNNs grow deeper and more intricate to capture finer details, they require significant computational resources and time for training. This complexity can lead to overfitting, where the model performs exceptionally well on the training data but fails to generalize to unseen data [9, 10, 11]. Overfitting is a critical problem in deep learning, as it undermines the models ability to make accurate predictions on new, real-world data. A crucial factor influencing the performance of CNNs is the initialization of the kernels in the convolutional layers. The choice of initialization methods can significantly impact the models convergence speed and overall accuracy. Poor initialization can cause the model to converge slowly or get stuck in suboptimal solutions, while good initialization can lead to faster convergence and better performance [12, 13, 14].

To address the issue of kernel initialization, our proposed algorithm, BO-IKM, leverages Bayesian Optimization to identify the optimal initialization methods for each convolutional layer in the CNN. Bayesian Optimization is a powerful technique for optimizing hyperparameters in machine learning models. By systematically exploring the search space of possible initializers, BO-IKM aims to find the best configuration that enhances the models accuracy and precision while reducing the risk of overfitting.

The BO-IKM algorithm was applied to the Plant Pathology 2020 and Plant Disease Recognition datasets, which present a challenging image classification problem involving various plant diseases. The results obtained using BO-IKM were compelling, demonstrating significant improvements in both accuracy and precision compared to CNN models that used standard initialization methods. Specifically, BO-IKM optimized the initialization methods for each convolutional layer, leading to faster convergence and more robust learning. The proposed algorithm not only improved the overall performance of the CNN models but also highlighted the importance of carefully selecting kernel initialization methods to enhance model efficiency. By reducing the complexity and mitigating overfitting, BO-IKM provides a valuable tool for developing high-performance CNN models in agricultural applications.

In the end, the BO-IKM algorithm represents a significant advance in the field of deep learning for plant disease classification. By leveraging Bayesian optimization to optimize kernel initialization methods, BO-IKM improves the accuracy and precision of CNN models, providing a robust solution to the challenges encountered in agricultural disease detection. The promising results obtained on the Plant Pathology 2020 and Plant Disease Recognition datasets underline the potential of this approach to improve crop health monitoring and contribute to sustainable agricultural practices.

2. Convolutional Neural Networks

In Convolutional Neural Networks, different types of layers play distinct roles in processing and extracting features from input data, typically images [15, 16, 17]. Here are detailed definitions of the convolutional layer, pooling layer, and fully connected layer:

1. Convolutional Layer
   The convolutional layer is the core building block of a CNN. It consists of a set of learnable filters that slide over the input data to produce feature maps. Each filter detects specific features such as edges, textures, or patterns. The operation is defined by a mathematical convolution, which involves:

$$Z_k^i = \sigma\left(\sum_{l=1}^{N_{i-1}} K_{l,k}^i X_l^{i-1}\right), \tag{1}$$

   where denotes the convolution product, σ is the activation function, and K is the trained kernel.
2. Pooling Layer
   The pooling layer reduces the spatial dimensions of the input, thereby decreasing the number of parameters and computations in the network. This layer helps in making the detection of features invariant to scale and orientation. The two common types of pooling are:
   - Max Pooling:

$$Y_{i,j} = \max\{X_{m,n}, m \in [i.s, i.s + p - 1], n \in [j.s, j.s + p - 1]\} \tag{2}$$

   - Average Pooling:

$$Y_{i,j} = \frac{1}{P^2} \sum_{m=i.s}^{i.s+p-1} \sum_{n=j.s}^{j.s+p-1} X_{m,n} \tag{3}$$

3. Fully Connected Layer
   The fully connected (FC) layer is typically used in the final stages of a CNN to produce the final output. Each neuron in an FC layer is connected to every neuron in the previous layer, just like in a traditional neural network.

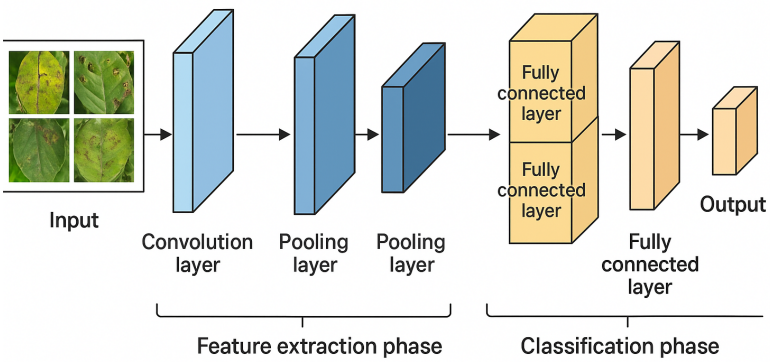


Figure 2. CNN Architecture

3. Bayesian Optimization

Bayesian Optimization (BO) is a powerful and efficient strategy for optimizing black box functions that are expensive to evaluate, particularly gaining attention in fields like machine learning for hyperparameter tuning. It utilizes a probabilistic model, typically a Gaussian Process (GP), to model the objective function, defined by its mean function $\mu(x)$ and covariance function $k(x, x^{'})$. The key steps include fitting a GP model to observed data, optimizing an acquisition function, and iteratively evaluating and updating the model. BOs main advantages are its sample efficiency, flexibility in handling various types of objective functions, uncertainty quantification, and its ability to find global optima. However, it faces challenges in scalability with high dimensional search spaces and computational complexity. Despite these limitations, BO is extensively applied in hyperparameter tuning, scientific experimentation, robotics, and Automated Machine Learning. These applications highlight BOs capability to significantly enhance model performance and optimize complex, expensive to evaluate functions, making it a crucial tool for researchers and practitioners [18, 19, 20].

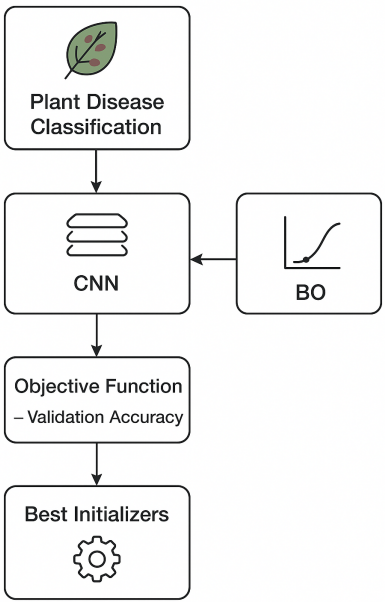


Figure 3. Bayesian Optimization for Initializer Kernels

4. Proposed Algorithm

The proposed algorithm, BO-IKM (Bayesian Optimization for Initializer Kernels Methods), aims to improve the performance of Convolutional Neural Network models by optimizing the initialization methods for the convolutional kernels. Initializing the kernels properly can lead to faster convergence during training and better overall model performance [21, 22, 23, 24, 25, 26]. The BO-IKM algorithm systematically searches for the best initialization methods using Bayesian Optimization, an efficient and effective approach for hyperparameter tuning. The BO-IKM algorithm 2 enhances Convolutional Neural Network performance by systematically optimizing the initialization methods of convolutional kernels. This approach is pivotal as the proper initialization of kernels significantly influences CNN convergence speed and overall model effectiveness. BO-IKM leverages Bayesian Optimization, a methodical search technique that efficiently explores the space of initialization strategies, including random, Glorot [7], He [27], and orthogonal initializers [28]. By minimizing the validation accuracy through Bayesian Optimization, BO-IKM identifies the optimal set of initializers for each convolutional layer. This systematic tuning improves model accuracy and precision and ensures computational efficiency by reducing the

number of evaluations needed. Applied to challenging datasets like Plant Pathology 2020 and Plant Disease Recognition, BO-IKM has demonstrated substantial performance gains over traditional initialization methods, underscoring its potential to advance CNN capabilities in image classification tasks.

Algorithm 1 Bayesian Optimization for CNN Tuning

Step 1: Define the Search Space
Define the set of hyperparameters $\mathcal{H}$ and their possible values.
Step 2: Construct the CNN Model
Construct the CNN model using hyperparameters from the search space $\mathcal{H}$.
Step 3: Define the Objective Function
Given a set of hyperparameters $\mathbf{H} \in \mathcal{H}$, define the objective function:

$$\text{Objective}(\mathbf{H}) = -\text{Validation Accuracy} \tag{4}$$

Step 4: Perform Bayesian Optimization
Step 4.1: Initialize Trials
Initialize trials: $\mathcal{T} \leftarrow \text{Trials}()$
Set the maximum number of evaluations N
Step 4.2: Gaussian Process Modeling
Use a Gaussian Process to model the objective function. The GP is defined by its mean function $\mu(x)$ and covariance function $k(x, x')$.

- **Mean Function** $\mu(x)$: Represents the expected value of the objective function at point x.
- **Covariance Function** $k(x, x')$: Measures the similarity between points x and x', providing the correlation between the objective function values at these points.

$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x')) \tag{5}$$

Step 4.3: Optimize the Acquisition Function
Use the GP model to define an acquisition function (e.g., Expected Improvement, Probability of Improvement, or Upper Confidence Bound) to determine the next point to evaluate.
Optimize the acquisition function to find the next set of hyperparameters $\mathbf{H}$ to evaluate:

$$\mathbf{H}_{next} = \arg\max_{\mathbf{H} \in \mathcal{H}} \text{Acquisition}(\mathbf{H}|\mathcal{T}) \tag{6}$$

Step 4.4: Update the GP Model
Evaluate the objective function at $\mathbf{H}_{next}$, add the result to trials $\mathcal{T}$, and update the GP model.
Step 4.5: Iterate
Repeat steps 4.2 to 4.4 until the maximum number of evaluations N is reached.
Step 4.6: Find the Best Set of Hyperparameters
Find the best set of hyperparameters:

$$\mathbf{H}^* = \arg\min_{\mathbf{H} \in \mathcal{H}} \text{Objective}(\mathbf{H}) \tag{7}$$

Step 5: Train the Final Model with the Best Hyperparameters
Construct and train the final model using $\mathbf{H}^*$
Compile the model
Train the model on the entire training set
Evaluate the model on the test set

Algorithm 2 BO-IKM: Bayesian Optimization for Initializer Kernels Methods

Step 1: Define the Search Space
Define the set of initializers:

$$\mathcal{I} = \{\text{random\_uniform}, \text{random\_normal}, \text{glorot\_uniform}, \text{glorot\_normal}, \text{he\_uniform}, \text{he\_normal}, \text{orthogonal}\}$$

Define the search space for the initializers:

$$\mathcal{S} = \{\text{initializer}_i \in \mathcal{I} \mid i = 1, 2, \ldots, 6\}$$

Step 2: Construct the CNN Model
Construct the CNN model using initializers from the search space $\mathcal{S}$.
Step 3: Define the Objective Function
Given a set of initializers $\mathbf{I} \in \mathcal{S}$, define the objective function:

$$\text{Objective}(\mathbf{I}) = -\text{Validation Accuracy} \tag{8}$$

Step 4: Perform Bayesian Optimization
Initialize trials: $\mathcal{T} \leftarrow \text{Trials}()$
Set the maximum number of evaluations: $N = 50$
Find the best set of initializers:

$$\mathbf{I}^* = \arg\min_{\mathbf{I} \in \mathcal{S}} \text{Objective}(\mathbf{I}) \tag{9}$$

Step 5: Train the Final Model with the Best Initializers
Construct and train the final model using $\mathbf{I}^*$
Compile the model with Adam optimizer and sparse categorical crossentropy loss
Train the model on the entire training set
Evaluate the model on the test set

5. Datasets

5.1. Plant Pathology 2020 Dataset

The Plant Pathology 2020 dataset is a collection of images describing various apple leaf diseases. The dataset was part of a competition hosted by Kaggle, which challenged participants to develop models for accurately identifying plant diseases. The dataset consists of high-quality images of apple leaves, annotated with labels indicating different disease categories [29, 30]. The primary objective is to classify each image into one of the following categories:

Table 1. Summary of the Plant Pathology 2020 Dataset

Category	Description	Number of Images
Healthy	Images of healthy apple leaves without any signs of disease.	1,821
Scab	Images showing apple leaves affected by apple scab disease.	592
Rust	Images displaying symptoms of rust disease on apple leaves.	622
Multiple Diseases	Images where apple leaves exhibit symptoms of more than one disease.	91

Figure 4. Images of each class in the Plant-pathology-2020 datasets

5.2. Plant Disease Recognition Dataset

The Plant Disease Recognition Dataset is a curated collection of RGB images classified into three main classes: *Healthy*, *Powdery*, and *Rust*. It consists of high-resolution images of plant leaves captured under various natural conditions and exhibiting a wide range of disease manifestations. The dataset is organized into training, validation, and testing directories, each containing subfolders for the respective classes. This structure facilitates supervised learning tasks such as image classification and feature learning [31, 32]. The diversity in lighting conditions, leaf orientations, and disease severity provides a robust benchmark for evaluating deep learning models on their ability to generalize and accurately distinguish between healthy and diseased plants. The data set is especially useful in agricultural research and precision farming applications, where early and accurate disease detection is crucial to yield protection.

Table 2. Summary of the Plant Disease Recognition Dataset

Category	**Description**	**Number of Images**
Healthy	Leaves without visible signs of disease	4560
Powdery	Leaves infected with powdery mildew	3600
Rust	Leaves affected by rust disease	3840

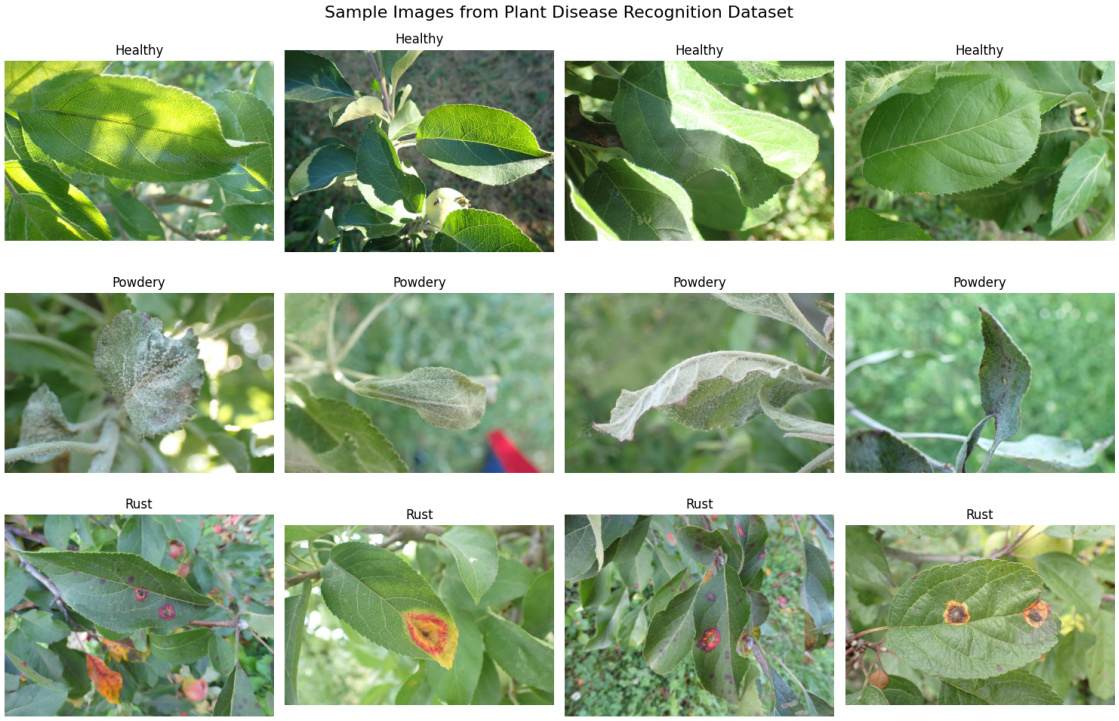


Figure 5. Images of each class in the Plant Disease Recognition Dataset

6. Results and Discussion

In evaluating computational efficiency, we analyzed the average training time per epoch, the overall wall-clock optimization time. All experiments were carried out on a GPU to ensure consistency across models. The integration of BO-IKM introduces an additional computational overhead, increasing the total training time by approximately **15%** compared to conventional initialization schemes. Nevertheless, this moderate increase is counterbalanced by superior convergence stability and improved predictive performance. The models under investigation included LeNet-5, BCNN, VGG19, and MobileNetV2, each trained with the Adam optimizer (learning rate = 0.001), a batch size of 32, regularization Dropout, and categorical cross-entropy loss. The Bayesian Optimization framework explored a search space encompassing seven initialization strategies: random uniform, random normal, glorot uniform, glorot normal, he uniform, he normal, orthogonal, while employing the Expected Improvement acquisition function combined with an RBF Gaussian Process kernel as the surrogate model. This configuration enabled BO-IKM to adapt initialization schemes in a layer-wise manner, leading to consistent gains in classification accuracy and F1-scores, while maintaining computational feasibility for practical deployment.

6.1. Plant Pathology 2020 Dataset

The results of applying the BO-IKM algorithm to three CNN models: Baseline CNN with 8 layers (BCNN), LeNet-5 [33], and MobileNet [34] show significant improvements in performance measures such as training accuracy, validation accuracy, training loss, validation loss, accuracy and F1 score.

In Table 3, the BCNN achieved a high training accuracy of 98% and a validation accuracy of 74%. This model also demonstrated a low training loss of 0.07 and a higher validation loss of 1.05, indicating some overfitting. The precision and F1-score were moderate at 0.72 and 0.73, respectively. LeNet-5 and MobileNet models showed different performance levels, with LeNet-5 achieving 90% training accuracy and 40% validation accuracy, while

Table 3. The results of CNNs models with BO-IKM algorithm for plant pathology 2020 dataset

Models	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss	Precision	F1-Score
BCNN	0.98	0.74	0.07	1.05	0.72	0.73
LeNet-5	0.90	0.40	0.22	9.85	0.41	0.40
MobileNet	0.96	0.87	0.09	0.50	0.85	0.86

MobileNet achieved 96% training accuracy and 87% validation accuracy. Both models exhibited varying degrees of overfitting as evidenced by their higher validation losses compared to training losses.

Table 4. The results of CNN models with BO-IKM algorithm

Models	Best Loss	Best Initializers
BCNN	0.74	'initializer_1': 'glorot_uniform', 'initializer_2': 'orthogonal', 'initializer_3': 'random_uniform', 'initializer_4': 'glorot_uniform', 'initializer_5': 'he_normal', 'initializer_6': 'random_normal', 'initializer_7': 'random_normal', 'initializer_8': 'orthogonal'
LeNet-5	0.48	'initializer_1': 'random_uniform', 'initializer_2': 'random_normal', 'initializer_3': 'he_normal', 'initializer_4': 'glorot_uniform', 'initializer_5': 'he_normal', 'initializer_6': 'random_uniform'
MobileNet	0.60	'initializer_1': 'glorot_uniform', 'initializer_2': 'glorot_uniform', 'initializer_3': 'glorot_uniform', 'initializer_4': 'random_uniform', 'initializer_5': 'random_uniform', 'initializer_6': 'glorot_normal', 'initializer_7': 'glorot_normal', 'initializer_8': 'glorot_uniform'

Table 4 provides insights into the best loss achieved by each model along with the specific initializers that contributed to these results. For instance, the BCNN achieved the best validation loss of 0.74 with a combination of initializers such as GlorotUniform, Orthogonal, RandomUniform, HeNormal, and RandomNormal. Similar trends were observed for LeNet-5 and MobileNet, each leveraging specific sets of initializers tailored to their architectures.

Tables 5, 6, and 7 and Figures 6, 7 delve into the specific convolutional and dense layers within each model and their corresponding best initializers identified by the BO-IKM algorithm. For example, in the BCNN Table 5, notable initializers like GlorotUniform, Orthogonal, RandomUniform, and HeNormal were assigned to different convolutional and dense layers, highlighting the algorithms ability to optimize initialization methods at a granular level.

Table 5. The results of the BCNN model with BO-IKM algorithm

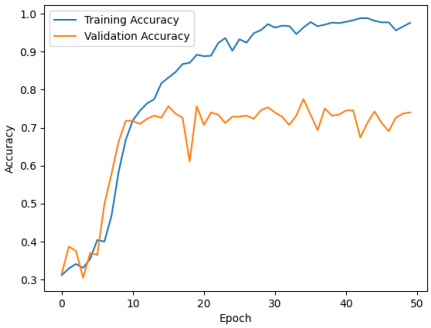
Layers	Best Initializers
$conv2d\_300$	Initializer: GlorotUniform
$conv2d\_301$	Initializer: Orthogonal
$conv2d\_302$	Initializer: RandomUniform
$conv2d\_303$	Initializer: GlorotUniform
$conv2d\_304$	Initializer: HeNormal
$conv2d\_305$	Initializer: RandomNormal
$dense\_100$	Initializer: RandomNormal
$dense\_101$	Initializer: Orthogonal

Table 6. The results of LeNet-5 models with BO-IKM algorithm

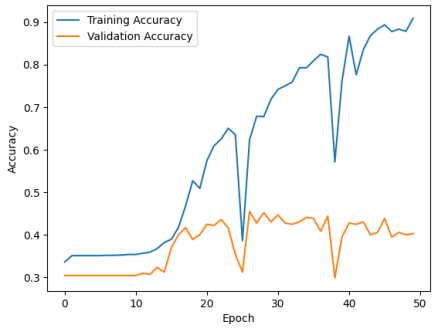
Layers	Best Initializers
$conv2d\_204$	Initializer: RandomUniform
$conv2d\_205$	Initializer: RandomNormal
$dense\_306$	Initializer: HeNormal
$dense\_307$	Initializer: GlorotUniform
$dense\_308$	Initializer: HeNormal

Table 7. The results of MobileNet models with BO-IKM algorithm

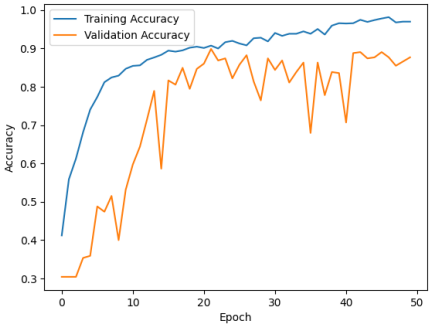
Layers	Best Initializers
$conv2d\_200$	Initializer: GlorotUniform
$conv2d\_201$	Initializer: GlorotUniform
$conv2d\_202$	Initializer: RandomUniform
$conv2d\_203$	Initializer: GlorotNormal
$dense\_50$	Initializer: GlorotUniform



(a) BCNN model



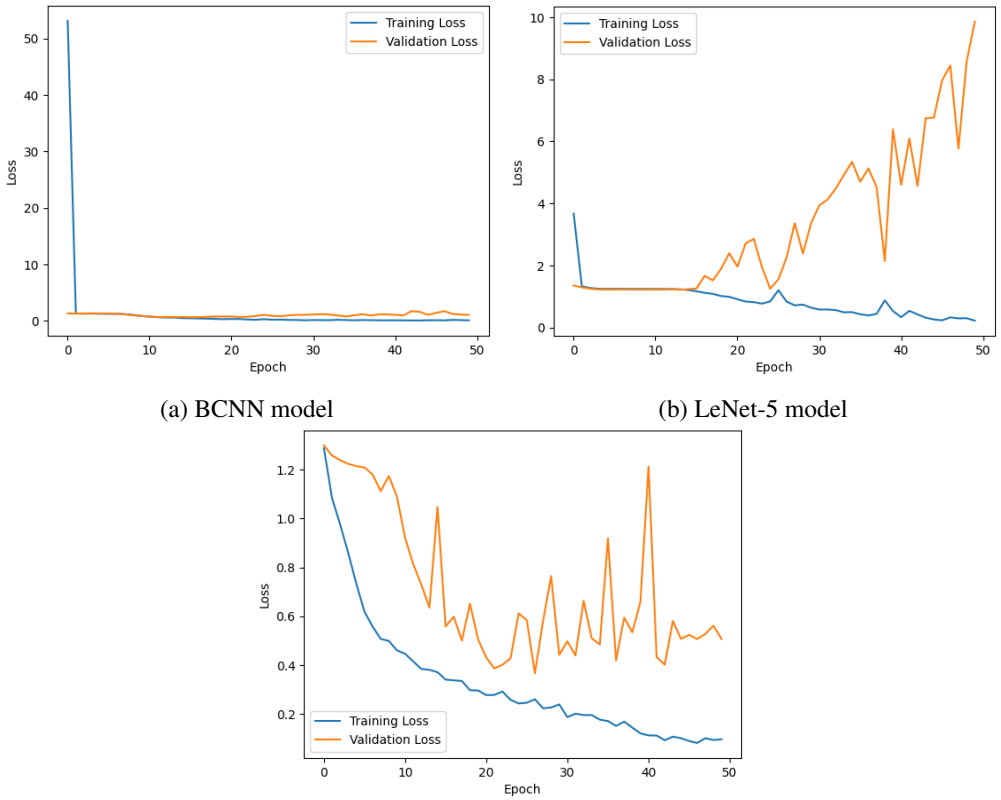
(b) LeNet-5 model



(c) MobileNet model

Figure 6. Accuracy results for each class of CNN models with the BO-IKM algorithm

Overall, the results underscore the efficacy of BO-IKM in enhancing CNN performance across various architectures and datasets, particularly emphasizing improvements in accuracy, precision, and robustness against overfitting. These findings suggest that optimized kernel initialization methods through Bayesian Optimization can significantly benefit CNN models in image classification tasks.



(a) BCNN model (b) LeNet-5 model

(c) MobileNet model

Figure 7. Loss results for each class of CNN models with the BO-IKM algorithm

6.2. Plant Disease Recognition Dataset

The application of the proposed BO-IKM algorithm to four convolutional neural network architectures BCNN, LeNet-5 [33], VGG19 [35], and MobileNet [34] demonstrated notable improvements in multiple performance metrics. These include increases in training and validation accuracy, as well as reductions in training and validation loss. Furthermore, the algorithm contributed to improved overall classification accuracy and the F1 score, indicating its effectiveness in optimizing CNN models for plant disease recognition tasks.

Table 8. The results of CNNs models with BO-IKM algorithm for plant disease recognition dataset

Models	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss	Precision	F1-Score
BCNN	0.88	0.87	0.97	1.05	0.88	0.87
LeNet-5	0.79	0.73	1.05	1.85	0.76	0.73
VGG19	0.92	0.90	0.83	0.85	0.91	0.90
MobileNet	0.69	0.70	1.08	0.98	0.71	0.70

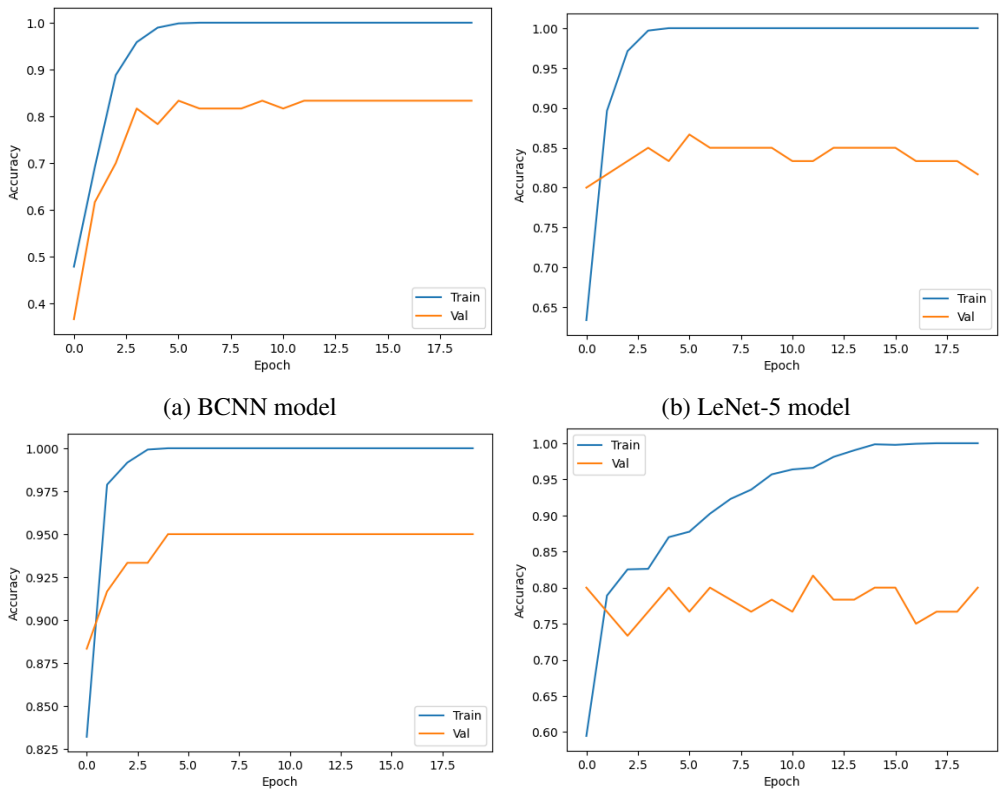
The experimental results presented in Tables 8, 9 and Figures 8,9demonstrate the effectiveness and performance gains achieved by the CNN models proposed, enhanced with the BO-IKM Bayesian Optimization-based initializer

Table 9. The results of CNN models with BO-IKM algorithm for plant disease recognition dataset

Models	Best Accuracy	Best Loss	Best Initializers
BCNN	0.88	0.87	'initializer_1': 'he_uniform', 'initializer_2': 'orthogonal', 'initializer_3': 'orthogonal', 'initializer_4': 'glorot_uniform', 'initializer_5': 'he_normal', 'initializer_6': 'orthogonal', 'initializer_7': 'random_normal', 'initializer_8': 'orthogonal'
LeNet-5	0.87	0.39	'initializer_1': 'orthogonal', 'initializer_2': 'orthogonal', 'initializer_3': 'he_uniform', 'initializer_4': 'glorot_uniform', 'initializer_5': 'glorot_uniform', 'initializer_6': 'he_normal'
VGG16	0.95	0.14	'initializer_1': 'random_uniform', 'initializer_2': 'random_normal', 'initializer_3': 'he_normal', 'initializer_4': 'glorot_uniform', 'initializer_5': 'he_normal', 'initializer_6': 'random_uniform' 'initializer_7': 'glorot_uniform', 'initializer_8': 'orthogonal', 'initializer_9': 'orthogonal', 'initializer_10': 'random_uniform', 'initializer_11': 'orthogonal', 'initializer_12': 'glorot_normal', 'initializer_13': 'orthogonal'
MobileNet	0.82	0.41	'initializer_1': 'glorot_uniform', 'initializer_2': 'orthogonal', 'initializer_3': 'glorot_uniform', 'initializer_4': 'he_uniform', 'initializer_5': 'orthogonal', 'initializer_6': 'he_normal', 'initializer_7': 'orthogonal', 'initializer_8': 'glorot_uniform'

kernel method in the context of recognition of plant diseases. The integration of BO-IKM led to notable improvements in model accuracy, loss minimization, and predictive consistency across multiple architectures, ranging from classical models like LeNet-5 to more modern and deeper networks such as VGG16 and MobileNet. In the table of results 8, VGG19 achieved the highest validation accuracy of 90% and an F1 score of 0.90, followed by BCNN 87% and LeNet-5 73%. However, when BO-IKM was applied Table 9, the performance of all models improved significantly, with VGG16 achieving a remarkable 95% accuracy and a substantially reduced validation loss of 0.14, demonstrating the powerful synergy between deep architectures and optimal initialization strategies. This indicates that proper initialization plays a crucial role in network convergence and generalization, especially in deeper architectures. In particular, the accuracy of LeNet-5 improved from 73% to 87% and its loss decreased from 1.85 to 0.39, showing that even simple models can benefit from data-driven kernel initialization. Similarly, MobileNet, which initially showed modest performance, improved to 82% accuracy with

a lower validation loss of 0.41, proving that BO-IKM is also highly beneficial for lightweight architectures used in resource-constrained environments. The best-performing initializer combinations frequently included orthogonal, glorot, and he initializers, indicating their effectiveness in maintaining variance stability and gradient flow across layers. Overall, these findings validate the proposed BO-IKM framework as a highly adaptable and efficient method for kernel initialization, enhancing the learning dynamics of CNNs, and significantly improving model performance across a wide range of network configurations. This positions BO-IKM as a practical and scalable approach for real-world plant disease recognition tasks and potentially for other computer vision applications that require optimized deep learning architectures.



(a) BCNN model (b) LeNet-5 model

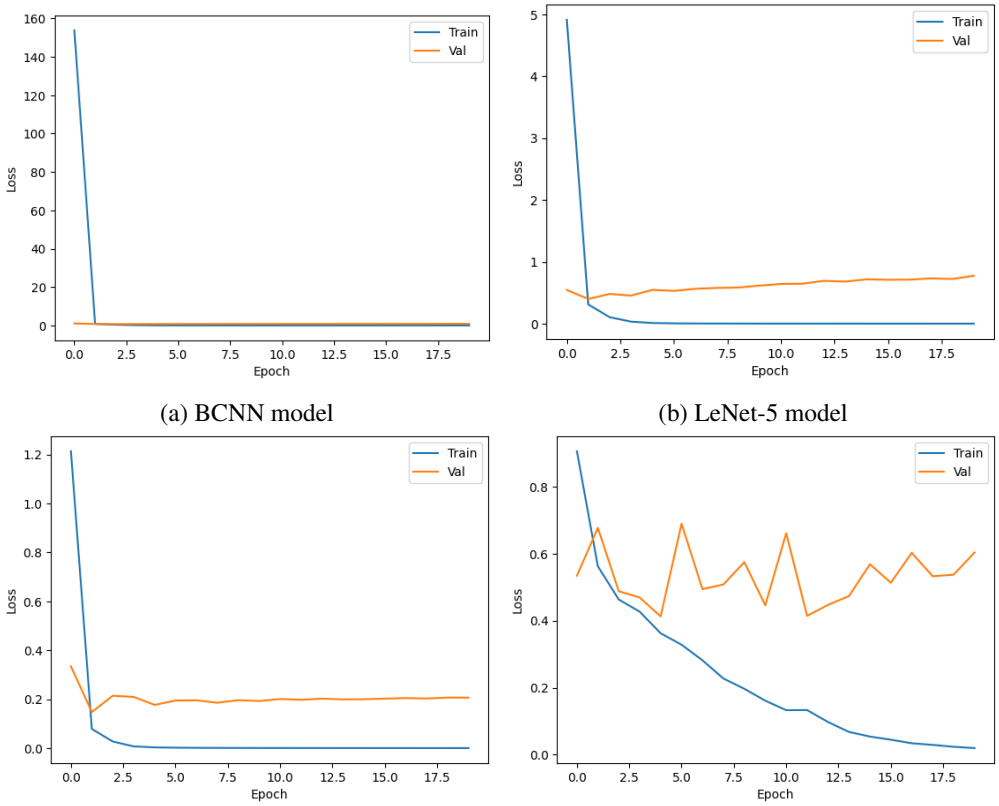
(c) VGG16 model (d) MobileNet model

Figure 8. Accuracy results for each class of CNN models with the BO-IKM algorithm for Plant Disease Recognition Dataset

Highlighting the novelty and practical importance of the proposed BO-IKM, Table 10 presents a comparative overview of BO-IKM and several well-established initialization and optimization techniques used in deep learning. While traditional methods such as Glorot and He initialization are widely adopted for their simplicity and effectiveness in stabilizing the gradient flow, they apply a uniform strategy to all layers without considering architecture-specific or data-based adaptation. Other approaches, such as orthogonal initialization and metaheuristic-based optimizers, offer certain performance advantages but often lack layer-level granularity or require high computational overhead. In contrast, BO-IKM introduces a systematic and efficient layer-specific initialization framework that leverages Bayesian optimization to search for optimal kernel initializers, thereby improving convergence and generalization. This comparison highlights BO-IKM unique contribution to bridging the gap between static initializer design and intelligent, data-driven initialization tailored to plant disease recognition and beyond.

Table 10. Comparison of the BO-IKM algorithm with CNN-Based Methods for Plant Disease Classification

Method	**Accuracy**	**Reference**
BO-IKM	95% (VGG16)	-
Xavier Initialization	88% (VGG)	[9]
He Initialization	90% (VGG19)	[27]
Orthogonal Initialization	89% (ResNet)	[28]
Genetic CNN	91% (Custom CNN)	[36]



(a) BCNN model

(b) LeNet-5 model

(c) VGG16 model

(d) MobileNet model

Figure 9. Loss results for each class of CNN models with the BO-IKM algorithm for Plant Disease Recognition Dataset

7. Conclusion

The proposed BO-IKM algorithm leverages Bayesian Optimization to optimize kernel initialization methods in Convolutional Neural Networks, significantly enhancing model performance. Applied to the ”Plant Pathology 2020” dataset, BO-IKM demonstrated substantial improvements in training and validation accuracy, training and validation loss, precision, and F1-score compared to CNNs using standard initialization methods. Specifically, BO-IKM systematically explored the space of possible initializers to identify the optimal configurations for each

convolutional layer, resulting in faster convergence and improved learning efficiency. These results underscore BO-IKMs effectiveness in boosting CNN performance for image classification tasks, particularly in the challenging domain of plant disease detection. This approach not only enhances the accuracy of disease identification but also offers a robust method for optimizing deep learning models across various complex datasets.

REFERENCES

1. Lu, J., Tan, L., Jiang, H.: Review on convolutional neural network (cnn) applied to plant leaf disease classification. Agriculture 11(8), 707 (2021)
2. Fenu, G., Malloci, F.M.: Diamos plant: A dataset for diagnosis and monitoring plant disease. Agronomy 11(11), 2107 (2021)
3. Valarmathi, G., Suganthi, S., Subashini, V., Janaki, R., Sivasankari, R., Dhanasekar, S.: Cnn algorithm for plant classification in deep learning. Materials Today: Proceedings 46, 3684–3689 (2021)
4. Khan, B., Das, S., Fahim, N. S., Banerjee, S., Khan, S., Al-Sadoon, M. K., Islam, A. R. M. T. (2024). Bayesian optimized multimodal deep hybrid learning approach for tomato leaf disease classification. Scientific Reports, 14(1), 21525.
5. El Sakka, M., Ivanovici, M., Chaari, L., Mothe, J. (2025). A Review of CNN Applications in Smart Agriculture Using Multimodal Data. Sensors, 25(2), 472.
6. Francis, M., Deisy, C.: Disease Detection and Classification in Agricultural Plants Using Convolutional Neural Networks—a Visual Understanding, (2019). IEEE
7. Kawasaki, Y., Uga, H., Kagiwada, S., Iyatomi, H.: Basic study of automated diagnosis of viral plant diseases using convolutional neural networks, 638–645 (2015). Springer
8. Brahimi, M., Boukhalfa, K., Moussaoui, A.: Deep learning for tomato diseases: classification and symptoms visualization. Applied Artificial Intelligence 31(4), 299–315 (2017)
9. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256 (2010). JMLR Workshop and Conference Proceedings
10. Lagnaoui, S., En-naimani, Z., Haddouch, K. (2025). Stochastic PFRCosSim layer for solving filter redundancy problem in CNNs applied on plant disease classification. Evolutionary Intelligence, 18(1), 1-20.
11. Lagnaoui, S., En-Naimani, Z., Haddouch, K. (2022, October). The Effect of Normalization and Batch Normalization Layers in CNNs Models: Application to Plant Disease Classifications. In International Conference On Big Data and Internet of Things (pp. 250-262). Cham: Springer International Publishing.
12. Zhang, H., Feng, L., Zhang, X., Yang, Y., Li, J.: Necessary conditions for convergence of cnns and initialization of convolution kernels. Digital Signal Processing 123, 103397 (2022)
13. Narkhede, M.V., Bartakke, P.P., Sutaone, M.S.: A review on weight initialization strategies for neural networks. Artificial intelligence review 55(1), 291–322 (2022)
14. Wu, Jia, et al. "Hyperparameter optimization for machine learning models based on Bayesian optimization." Journal of Electronic Science and Technology 17.1 (2019): 26-40.
15. Wu, J.: Introduction to convolutional neural networks. National Key Lab for Novel Software Technology. Nanjing University. China 5(23), 495 (2017)
16. Kuo, C.-C. J. (2016). Understanding convolutional neural networks with a mathematical model. Journal of Visual Communication and Image Representation, 41, 406413
17. Zhang, Q., Zhang, M., Chen, T., Sun, Z., Ma, Y., Yu, B.: Recent advances in convolutional neural network acceleration. Neurocomputing 323, 37–51 (2019)
18. Snoek, J., Larochelle, H., Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems, 25.
19. Ibrahim, M. Q., Hussein, N. K., Guinovart, D., Qaraad, M. (2025). Optimizing Convolutional Neural Networks: A Comprehensive Review of Hyperparameter Tuning Through Metaheuristic Algorithms. Archives of Computational Methods in Engineering, 1-38.
20. Raiaan, M. A. K., Sakib, S., Fahad, N. M., Al Mamun, A., Rahman, M. A., Shatabda, S., Mukta, M. S. H. (2024). A systematic review of hyperparameter optimization techniques in Convolutional Neural Networks. Decision analytics journal, 100470.
21. Aslan, Muhammet Fatih. "Comparative Analysis of CNN Models and Bayesian Optimization-Based Machine Learning Algorithms in Leaf Type Classification." Balkan Journal of Electrical and Computer Engineering 11.1 (2023): 13-24.
22. Erkan, Ugur, Abdurrahim Toktas, and Deniz Ustun. "Hyperparameter optimization of deep CNN classifier for plant species identification using artificial bee colony algorithm." Journal of Ambient Intelligence and Humanized Computing 14.7 (2023): 8827-8838.
23. Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T. (2018). Rethinking the value of network pruning. arXiv preprint arXiv:1810.05270.
24. Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., Wilson, A. G. (2018). Averaging weights leads to wider optima and better generalization. arXiv preprint arXiv:1803.05407.
25. Wang, X., Jin, Y., Schmitt, S., Olhofer, M. (2023). Recent advances in Bayesian optimization. ACM Computing Surveys, 55(13s), 1-36.
26. Lagnaoui, S., Boumais, K., El Fallah, S., En-Naimani, Z., Haddouch, K., Matuzevi ˇcius, D.: Adaptive methods for kernel initialization of convolutional neural network model applied to plant disease classification. In: 2024 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream), pp. 1–6 (2024). IEEE
27. He, K., Zhang, X., Ren, S., Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).
28. Saxe, A. M., McClelland, J. L., Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. arXiv preprint arXiv:1312.6120.

29. Plant Pathology 2020 - FGVC7 - Kaggle,https://www.kaggle.com/c/plant-pathology-2020-fgvc7. Last accessed 2020
30. Thapa, R., Zhang, K., Snavely, N., Belongie, S., Khan, A.: The plant pathology challenge 2020 data set to classify foliar disease of apples. Applications in plant sciences 8(9), 11390 (2020)
31. Kaggle. Plant Disease Recognition Dataset. https://www.kaggle.com/datasets/rashikrahmanpritom/plantdisease-recognition-dataset. Last accessed 2021
32. Fu, L., Li, S., Sun, Y., Mu, Y., Hu, T., Gong, H. (2022). Lightweight-convolutional neural network for apple leaf disease identification. Frontiers in Plant Science, 13, 831219.
33. Wang, Z.F., Su, H.T., Chen, H.S., Hu, Z.Y., Wang, J.L.: A model of target detection in variegated natural scene based on visual attention. Applied Mechanics and Materials 333, 1213–1218 (2013)
34. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
35. Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
36. Xie, L., Yuille, A. (2017). Genetic cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1379-1388).