

# TALOS: optimization of the CNN for the detection of the tomato leaf diseases

Shruthi Kikkeri Subramanya, Naveen Bettahalli, Naveen Kalenahalli Bhoganna

Department of Electronics and Communication Engineering, BGS Institute of Technology, Adichunchanagiri University, Karnataka, India

## Article Info

### Article history:

Received Jun 5, 2024

Revised Oct 7, 2024

Accepted Oct 28, 2024

### Keywords:

CNN

Hyperparameter

Optimization

TALOS

Tomato leaf disease

## ABSTRACT

Early detection of plant diseases using convolutional neural network (CNN) is crucial for maximizing crop yield and minimizing economic losses. Manual inspection, the frequent technique, is inefficient and error prone. While CNN's offer potential for accurate and quick disease recognition, their performance is highly dependent on effective hyperparameter tuning. This process is time consuming, resource intensive, and needs significant expertise due to the vast hyperparameter space, since it can be hard to figure out which is ideal for optimal performance. An effective optimization tool, tunable automated hyperparameter learning optimization system (TALOS), is proposed, which automates the tuning of hyperparameters by systematically exploring the hyperparameter space and evaluates different combinations of parameters to find the optimal configuration that maximize the model's performance. The performance of this approach is recognizable through its exploration of five different hyperparameters across a search space of 32 combinations, yielding optimal parameters by the second round. Using 3030 tomato leaf images from a benchmark data set, the model achieves a remarkable 94.7% validation accuracy with 33647 trainable parameters. Thus, automated hyperparameter tuning approach not only optimizes model performance but also reduces manual effort and resource requirements, paving the way for more effective and scalable solutions in agricultural technology.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



## Corresponding Author:

Naveen Bettahalli

Department of Electronics and Communication Engineering, BGS Institute of Technology

Adichunchanagiri University

B.G. Nagara, Karnataka, India

Email: naveenb@bgsit.ac.in

## 1. INTRODUCTION

JayKordich's statement, "All life on earth emanates from the green of the plant," highlights the critical role of plants as the primary source of oxygen production, supporting aerobic life forms' survival. Additionally, they play a crucial role in maintaining ecological balance, regulating the earth's climate, and supporting our planet's intricate web of life. Due to the endemic diseases in plants, numerous plants are on the verge of becoming extinct [1], [2]. The challenge of accurately identifying the plant diseases is crucial due to the significant impact of these diseases can have on agriculture. Traditional methods, while useful in specific contexts, are often limited by their manual, labor-intensive nature, and dependency on expert knowledge. Visual inspections are subjective and inconsistent, microscopy requires specialized skills and is time consuming, and culturing is not applicable to all pathogens. Moreover, traditional machine learning (ML) approaches, while automated, often fail to handle the complexity and variability of disease symptoms effectively [3]-[5].

Deep learning (DL) is emerging as a powerful technology, especially convolutional neural networks (CNN's), which have been increasingly utilized. CNN can automatically extract relevant features from large and complex datasets, eliminating the need for manual feature extraction (FE) [6]-[8]. However, the application of CNNs is not without challenges. The performance of the DL model is crucial for achieving optimal performance which mainly depends on the quantity and quality of the images in the dataset, the robust design of the models, and the optimization of the hyperparameters. Firstly, a significant challenge in training the DL models is getting a high-quality dataset that are large, diverse, accurate, and well pre-processed [9], with balanced classes to prevent bias. This process is crucial but computationally expensive and it may cause over fitting [10] where the model performs better on the training data compared to validation/test data. Mathematically, the over fitting can be represented as follows:

$$E_{\text{train}} \ll E_{\text{test}} \quad (1)$$

where  $E_{\text{train}}$  is the training dataset error, and  $E_{\text{test}}$  is the error on test or validation datasets.

Second, the designing a robust model [11] involves choosing an appropriate architecture, configuring the layers effectively, selecting the proper activation function, and incorporating regularization to improve stability and generalization of the models. Lastly, hyperparameter Tuning is essential to optimize the model's performance which involves adjustment of the parameters such as batch size, number of epochs, learning rate, and the choice of optimizer [12]. Traditional methods like grid search [13], while exhaustively, are computationally expensive as the model complexity increases; random search [14], while more efficient for large datasets, lacks the certainty of finding the best configuration. Also, the manual tuning, though providing insights, is subjective, time-consuming, and prone to errors, particularly with complex models [15]. Therefore, finding an efficient and effective tuning strategy for large and intricate models remains a significant challenge in model optimization [16].

Reviewing the relevant literature helps to identify major contributors' work and findings, guiding potential advancements in the field by summarizing the recent progressions in hyperparameter tuning, including algorithms like grid search, random search, and Bayesian optimization, which aim to enhance optimization efficiency and performance.

The work proposed in [17] highlights the whale optimization algorithm (WOA), to optimize the hyperparameters in neural networks. It achieved a notable accuracy of 80.60% and 89.85% on Reuters datasets and Fashion MNIST, respectively. The research proposed in [18] is a 14-layered deep CNN (14-DCNN) to identify diseases from a dataset of 147,500 images of 58 plant leaf classes. The model is trained for 1,000 epochs and then optimized using random search with coarse-to-fine hyperparameters searching. The stated DCNN model provides a 99.9655% high accuracy, 99.7966% recall, 99.7999% weighted average precision, and 99.7968 F1-score. The methodology in [19] employs a WOA with hybrid principal component analysis (PCA) to identify diseases from a dataset of 18159 of 10 classes of tomato leaves from the PlantVillage dataset. Grid search is adopted to tune and find the optimal hyperparameters, which enhances model performance. The model provides 99% of training accuracy and 86% testing accuracy at the 15<sup>th</sup> epoch. Pandian *et al.* [20] the DCNN model with five convolutional layers is trained with the augmented dataset of the PlantVillage dataset, which originally contains images of 55448 with 39 different classes is subjected to deep convolutional generative adversarial networks (DCGAN) augmentation techniques resulting in a set of 240,000 images. Hyperparameters are set using a random search, resulting in an accuracy of 98.41% on the test dataset.

The proposed work in [21] is a contextual mask auto-encoder optimized with a dynamic differential annealed optimization algorithm (PDI-CMAE-DDAOA) for the early detection of plant diseases. PDI-CMAE-DDAOA achieves higher accuracy 23.34%, 34.33%, and 32.07%, F1-score 46.67%, 57.56%, sensitivity 36.67%, 36.33%, and 23.21%, and 43.21%, and specificity 56.67%, 67.56%, and 23.21% compared to these existing models such as PDI-DENN, PDI-CAE-CNN, and PDI-EN-CNN, respectively. Dudi and Rajesh [22] present a method that uses the sharksmll-based-WOA (SS-WOA) to optimize the CNN's activation function for maximum classification accuracy. Compared to NB and SVM, the accuracy of the supplied SS-WOA-CNN is 7.14% and 5.63% higher respectively. Halim *et al.* [23] considered the CNN architectures such as Xception and DenseNet with the AiSara tuning algorithm, resulting in a 23% improvement in accuracy compared to typical tuning techniques. The models on the PlantVillage and PlantDoc datasets are evaluated in the study; DenseNet121 and Xception obtained accuracy of 89.60% and 85.94% on PlantVillage and 81.51% on PlantDoc, respectively, without hyperparameter adjustment. Accuracy increased to 94.75% and 91.03% on PlantVillage and 84.84% and 87.66% on PlantDoc with AiSara tweaking. Akkuş *et al.* [24] evaluates the effectiveness of two CNN models, ResNet18 and AlexNet in detecting the severity of sariopsis leaf spot disease in grape leaves which are gathered from the PlantVillage dataset. The study explores the impact of tuning hyperparameters, including epochs, data augmentation, and mini-batch size. ResNet18 reached an accuracy of 87.6% with a mini-batch size of 64, 10

epochs, and with data augmentation. AlexNet achieved the highest classification accuracy of 90.31% with a mini-batch size of 32, and 50 epochs, and without data augmentation.

A successful implementation of DL for the identification of plant diseases depends on several factors, including dataset's quality, the architecture of the model, and the optimization of hyperparameters. Despite using techniques such as WOA, PCA, FOA [25] and random search, challenges remain in addressing these inter dependencies efficiently. A holistic approach incorporating robust data set, advanced model architectures, and efficient hyperparameter optimization is essential. The proposed method tackles these challenges by considering the following sequential steps:

- This study proposes a disease identification system for tomato plants utilizing a customized CNN architecture designed to provide a reduced parameter.
- The training of a CNN model is performed with a limited dataset of tomato leaf images.
- The study employs tunable automated hyperparameter learning optimization system (TALOS) for hyperparameter tuning, resulting in efficiently optimizing model performance while minimizing manual intervention and computational resources, thus meeting its primary objective.

The remaining section of the research work is organized according to the preceding structure. Section 2 presents a proposed model for a classification system based on CNN and TALOS optimization. Section 3 shows the results and discussion. Section 4 conclude the work.

## 2. METHOD

A customized CNN is used with a hyperparameter optimization tool to identify tomato plant leaf diseases using a minimal number of images. TALOS is a hierarchical optimizer that tunes the hyperparameters without manually testing each value. An illustration of the model for categorizing tomato leaf diseases is shown in Figure 1. The methodology is performed in the following steps:

- a. Dataset collection and pre- processing
- b. TALOS setup
- c. Define the CNN architecture
- d. TALOS integration
- e. TALOS experiment
- f. Selecting optimal parameters

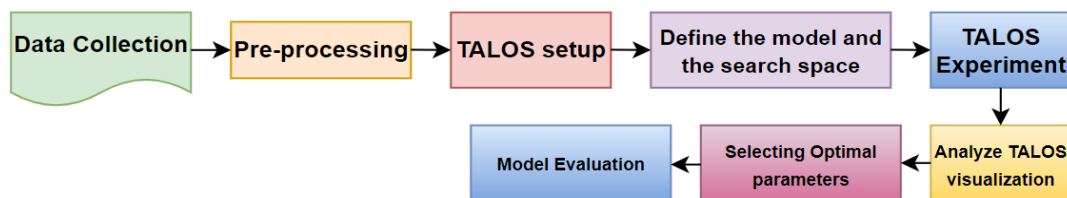


Figure 1. TALOS-CNN based tomato leaf disease classification system

### 2.1. Data collection and pre-processing

The tomato leaves dataset employed in this study is obtained from the PlantVillage dataset. The late blight, yellow leaf curl, and healthy leaf of the tomato are considered for experimentation, as shown in Figure 2. Each class contains an image of 900 in the trained set and 110 in the testing set, balanced and indicating an even distribution of attributes across all categories as it prevents any single class from dominating. The dataset's images were pre-processed so the suggested model could extract their necessary attributes. The images were normalized and resized to 256×256 pixels in the first stage.



Figure 2. Sample leaf images of the tomato leaves from the training dataset

**2.2. Defining the CNN architecture**

This study incorporates a CNN as a disease detection approach. The CNN architecture comprises: the FE stage and classification stage. The proposed CNN architecture consists of layers such as batch normalization, Conv2D, MaxPooling, and dropout layers in the FE part, followed by dense and dropout layers in the classification part. The customized CNN architecture is shown in Figure 3, with the corresponding parameters correlated to the layers displayed in Table 1. The architecture integrates four batch normalization layers and four convolution layers with a stride of (1, 1), preserving fine spatial features crucial for effective FE. Pooling layers decrease output dimensionality, reducing computational complexity for subsequent layers; employing max-pooling with a 2x2 pool size optimizes this process. Despite the dense layer’s significant parameter count (9248), its role in capturing complex feature relationships remains pivotal. However, this parameter count is comparatively modest. The SoftMax activation function, applied to the dense layer’s output, transforms it into a probability distribution. This architecture balances FE, regularization, and parameter efficiency, offering robust performance in tomato leaf disease detection while mitigating computational overhead. The configuration of a neural network model is depicted in Table 2.

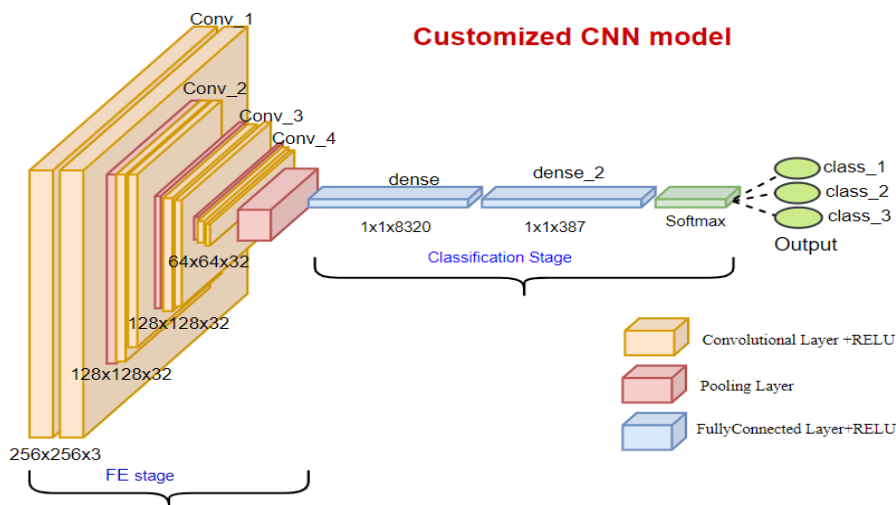


Figure 3. CNN model architecture for tomato leaf disease recognition

Table 1. Key parameters for CNN’s model

Sl.no.	Factors	Defined value
1	Filter size	3x3
2	Number of filters	32
3	Pooling	MaxPooling with a 2x2 pooling window and strides of 1x1
4	Activation function	ReLU (rectified linear unit) and ELU (exponential linear unit)
5	Loss function	Categorical cross-entropy

Table 2. Network summary

Sl.no.	Layers	Activation functions	Trainable parameters
1.	batchnormalization	256*256*3	12
	conv2d_1	256,256,32	896
	max_pooling2d	255*255*3	0
2.	batchnormalization	255*255*3	128
	conv2d_2	255*255*3	9248
	max_pooling2d_1	254*254*3	0
3.	batchnormalization	254*254*3	128
	conv2d_3	254*254*3	9248
	max_pooling2d_2	253*253*3	0
4.	batchnormalization	253*253*3	128
	conv2d_4	253*253*3	9248
	max_pooling2d_3	252*252*3	0
	FC	128	4224
	Dense	128	387
<b>Total parameters: 33,647</b>			

### 2.3. TALOS integration with CNN's

Fine-tuning a CNN model typically involves manually setting hyperparameters followed by grid or random search approaches to examine alternative configurations systematically. This method can be computationally demanding, especially when dealing with complex models and enormous datasets [26]. After defining the CNN model, TALOS, a Python module designed to optimize hyperparameters, is an assistant that explores different hyperparameter combinations for the model is employed. For the range of permissible values for each hyperparameter, TALOS tests the various combinations of these values and prioritizes those that seem promising for improving CNN's performance in identifying tomato leaf diseases.

### 2.4. Selecting optimal parameters

Once the set of hyperparameters and their search space is defined, TALOS experiment is initiated, after all the iterations, TALOS provides useful visualizations to track and identifies the optimal set of hyperparameters that leads to the best results [27]. Figure 4 shows the flow diagram of the Hyperparameter optimization technique using the TALOS. The Google Collaboratory platform (Colab), built around Jupyter Notebooks [14] is utilized to execute the experiment. The implementation also includes the study to examine how varying filter sizes in CNN models [28] affect accuracy in tomato leaf disease detection, demonstrating the critical role of hyperparameter optimization in enhancing performance. The following section provides an algorithm for optimizing a CNN model for detecting tomato leaf diseases using TALOS.

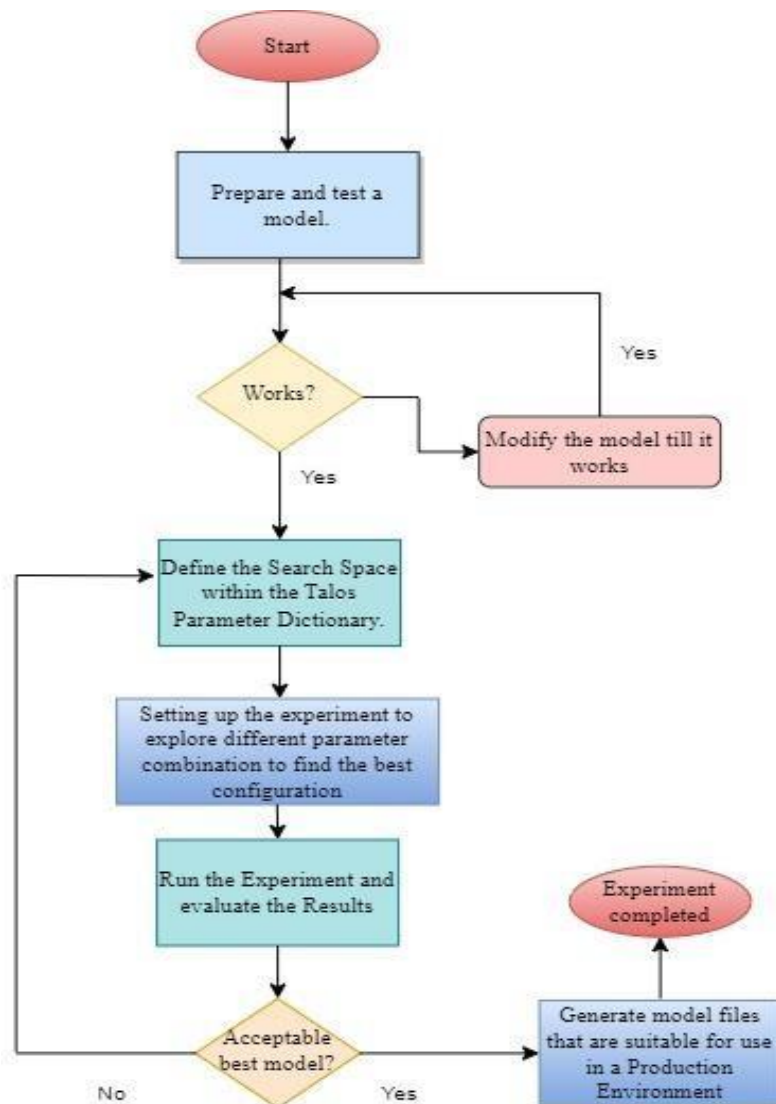


Figure 4. Hyper-parameter optimization technique using the TALOS

Input:

- Tomato leaf disease dataset
- Desired hyperparameter search space

Output:

- Best model configuration
  - Number of rounds in the scanning process ( $m^n=2^5$ , m: search options, n: total number of hyperparameters)
  - Highest accuracy given parameters set
1. Install required libraries: 'setup-tools' and 'TALOS', image processing libraries: OpenCV, PIL, data manipulation libraries: NumPy
  2. Data preprocessing
    - Load the tomato leaf disease dataset into Collab.
    - Pre-process images.
    - Convert categorical labels to one-hot encoded format.
    - Split data into training, validation, and testing sets.
  3. Define CNN model architecture
    - Create a custom CNN model with base architecture and parameters to be optimized.
    - Define the input dimensions and hyperparameters for the model.
  4. Define the search space for hyperparameters, including the ranges and values to be explored.
  5. Run TALOS experiment
    - Execute the TALOS scan to perform hyperparameter optimization on the CNN model.
  6. Result analysis
    - Print the results of the TALOS scan, including the top-performing configurations.
    - Identify the model ID that yields the best validation accuracy.
    - Load the best model.
    - Retrieve the total number of rounds completed in the TALOS scan and highest validation accuracy.

**3. RESULTS AND DISCUSSION**

The section showcases the results from automated tuning experiment so hyperparameters, which aim to increase the performance of the CNN model for plant leaf disease identification. Filters in CNNs are known for their efficient detection of important features such as textures, edges, and patterns in the images and allow the model to learn and detect complex details required for accurate classification tasks. Three scenarios of were explored using TALOS to customize the CNN with varying filter sizes and dense neurons, aiming to identify the most effective configuration for optimal accuracy. The results are summarized in Table 3. Filter sizes of 3x3 regularly out performed those of 5x5, most likely because they were better at capturing features. In addition to these, the main hyperparameters under investigation included the activation function, the optimizer technique, dropout, the Conv\_dropout, and dense neuron count, each hyperparameter tested across a defined range.

Table 3. Analysis of varying filter size and dense neuron on CNN

Filter size	Dense neurons	Accuracy	Trainable parameters
5x5	[32,64]	85%	56271
3x3	[512]	87%	47271
3x3	[128,256]	94.5%	37615

Table 4 shows the list of hyperparameters and their space coverage for analyzing the constraints of the TALOS tool. Once all ranges have been defined, the TALOS tool experiments combine all possible 32 combinations and for each configuration, TALOS trains a model and evaluates its performance on the dataset. Based on the evaluation results, TALOS identifies the hyperparameter set which results in best performance. Generally, the configuration with maximum validation accuracy and minimum validation loss is preferred. Hence, Model ID 25 is considered as the best model configuration with 94.5% accuracy and 94.7% validation accuracy. The hyperparameters that resulted in high accuracy can be found in the output column of Table 4. This approach drastically reduces computational time by tracking the performance of each trial. Thus, TALOS is able to provides the insights in to the effect of different hyperparameter settings on model performance.

Table 4. Definition of hyperparameters space and the corresponding result

Hyperparameters	Range	Output
Activation function	[ReLU, ELU]	ReLU
Optimizer	[Adam, SGD]	Adam
dropout,	[0.25, 0.5]	0.25
Conv_dropout	[0.25, 0.5]	0.25
Dense neuron	[128, 256]	128

The proposed CNN's performance is visualized in Figure 5 and Figure 6 which indicates the confusion matrix while Figures 6(a) and 6(b) depict the validation accuracy and validation loss of CNN respectively. Tables 5-7 compare the proposed TALOS-CNN's model performance with the state-of-the-art model developed in the work [14] and predefined models, reveals satisfactory performance. Although its accuracy is slightly lower, the model demonstrates competitive results in terms of total parameters and total elapsed time. TALOS can generate the visualization to provide insights into the optimization process as depicted in the Figure 7.

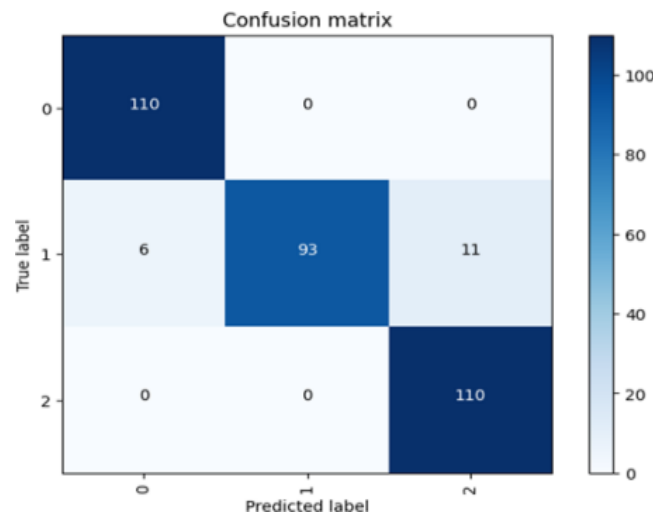


Figure 5. Confusion matrix

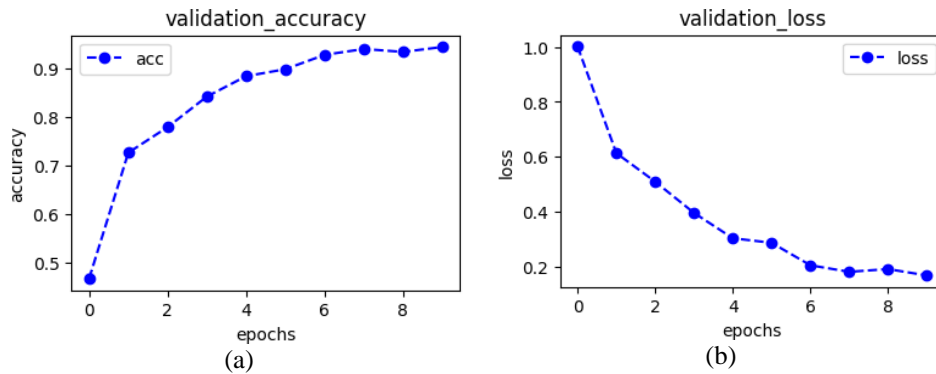


Figure 6. Validation performance of the CNN during training (a) validation accuracy curve and (b) validation loss curve

#### - Discussion

The findings of this study reveal that hyperparameter tuning is critical for optimizing CNN performance in tomato leaf disease detection. It explored three scenarios using TALOS, each with varying filter sizes and dense neuron configurations. The best-performing model, achieved an impressive accuracy of 94.5% with a 3×3 filter and dense neurons in the range of [128, 256]. This result underscores the importance

of careful hyperparameter adjustment, as filter sizes of 3×3 consistently outperformed the 5×5 filters, likely due to their superior ability to capture relevant features. The TALOS tool facilitated this optimization process by efficiently evaluating 32 different combinations of hyperparameters, with Model ID 25 achieving a 94.5% accuracy and a 94.7% validation accuracy.

When comparing the results with those of previous studies [29], [30] have similarly which also highlight the importance of hyperparameter tuning in achieving optimal CNN performance and note that smaller filter sizes often outperform larger ones in image classification tasks. However, this study uniquely demonstrates the practical application of the TALOS optimization tool, which systematically explores and identifies the best hyperparameter combinations. Unlike previous works that often relied on manual or heuristic methods, this approach leverages automated optimization to achieve superior results with greater efficiency. Tables 5-7 compare this work with previous studies in terms of model complexity, total parameters, training time, and dataset size.

A strength of our study lies in the comprehensive evaluation of different hyperparameter configurations using a robust optimization tool, which contrasts with the more limited scope of previous research that typically explored fewer configurations. Nonetheless, our model’s accuracy, while high, is slightly lower than some state-of-the-art models reported in the literature. This suggests that while our approach is effective, there is room for improvement, particularly in integrating more sophisticated techniques or additional layers of optimization. Unexpectedly, the model with the largest dense neuron configuration in case 2 did not outperform the smaller configuration in case 3, indicating that higher model complexity does not necessarily translate to better performance, which aligns with findings in some previous studies.

round_epochs	loss	accuracy	val_loss	val_accuracy	dense1_neuron	activation	conv_dropout	optimizer	dropout	
0	10	0.124471	0.960269	0.558389	0.796857	32	relu	0.25	<class 'keras.optimizers.legacy.adam.Adam'>	0.25
1	10	0.157838	0.943434	0.164887	0.932660	32	relu	0.25	<class 'keras.optimizers.legacy.adam.Adam'>	0.50
2	10	0.326553	0.890909	0.530065	0.819304	32	relu	0.25	<class 'keras.optimizers.legacy.gradient_desce...	0.25
3	10	0.379372	0.861616	0.661374	0.710438	32	relu	0.25	<class 'keras.optimizers.legacy.gradient_desce...	0.50
4	10	0.147502	0.953199	0.155801	0.940516	32	relu	0.50	<class 'keras.optimizers.legacy.adam.Adam'>	0.25
5	10	0.194205	0.930303	0.327024	0.843996	32	relu	0.50	<class 'keras.optimizers.legacy.adam.Adam'>	0.50
6	10	0.412820	0.836027	0.592955	0.846240	32	relu	0.50	<class 'keras.optimizers.legacy.gradient_desce...	0.25
7	10	0.420556	0.843434	0.639109	0.810326	32	relu	0.50	<class 'keras.optimizers.legacy.gradient_desce...	0.50
8	10	0.142948	0.951515	0.233683	0.921437	32	elu	0.25	<class 'keras.optimizers.legacy.adam.Adam'>	0.25
9	10	0.171989	0.933670	0.210467	0.915825	32	elu	0.25	<class 'keras.optimizers.legacy.adam.Adam'>	0.50
10	10	0.308430	0.876431	0.914302	0.656566	32	elu	0.25	<class 'keras.optimizers.legacy.gradient_desce...	0.25
11	10	0.347046	0.859259	0.704932	0.721661	32	elu	0.25	<class 'keras.optimizers.legacy.gradient_desce...	0.50
12	10	0.173298	0.936364	0.141422	0.946128	32	elu	0.50	<class 'keras.optimizers.legacy.adam.Adam'>	0.25
13	10	0.171635	0.938721	0.272429	0.895623	32	elu	0.50	<class 'keras.optimizers.legacy.adam.Adam'>	0.50
14	10	0.345053	0.861279	1.429729	0.576880	32	elu	0.50	<class 'keras.optimizers.legacy.gradient_desce...	0.25
15	10	0.385368	0.842424	0.625754	0.780022	32	elu	0.50	<class 'keras.optimizers.legacy.gradient_desce...	0.50
16	10	0.106558	0.966667	0.295195	0.883277	64	relu	0.25	<class 'keras.optimizers.legacy.adam.Adam'>	0.25
17	10	0.126860	0.955219	0.212944	0.925926	64	relu	0.25	<class 'keras.optimizers.legacy.adam.Adam'>	0.50
18	10	0.282924	0.896970	0.821124	0.506173	64	relu	0.25	<class 'keras.optimizers.legacy.gradient_desce...	0.25
19	10	0.319535	0.881145	0.732369	0.648709	64	relu	0.25	<class 'keras.optimizers.legacy.gradient_desce...	0.50
20	10	0.112875	0.960606	0.675002	0.753086	64	relu	0.50	<class 'keras.optimizers.legacy.adam.Adam'>	0.25
21	10	0.148799	0.943098	0.252984	0.903479	64	relu	0.50	<class 'keras.optimizers.legacy.adam.Adam'>	0.50
22	10	0.372131	0.854545	0.623623	0.815937	64	relu	0.50	<class 'keras.optimizers.legacy.gradient_desce...	0.25
23	10	0.423650	0.823906	0.649630	0.793490	64	relu	0.50	<class 'keras.optimizers.legacy.gradient_desce...	0.50
24	10	0.134229	0.952525	0.190472	0.923681	64	elu	0.25	<class 'keras.optimizers.legacy.adam.Adam'>	0.25
25	10	0.154532	0.945791	0.147076	0.947250	64	elu	0.25	<class 'keras.optimizers.legacy.adam.Adam'>	0.50
26	10	0.316136	0.872391	0.744439	0.686869	64	elu	0.25	<class 'keras.optimizers.legacy.gradient_desce...	0.25
27	10	0.324090	0.873401	0.942594	0.654321	64	elu	0.25	<class 'keras.optimizers.legacy.gradient_desce...	0.50
28	10	0.158519	0.942088	0.282282	0.904602	64	elu	0.50	<class 'keras.optimizers.legacy.adam.Adam'>	0.25
29	10	0.150342	0.944781	0.338260	0.867565	64	elu	0.50	<class 'keras.optimizers.legacy.adam.Adam'>	0.50
30	10	0.347989	0.856229	0.607176	0.726150	64	elu	0.50	<class 'keras.optimizers.legacy.gradient_desce...	0.25
31	10	0.368451	0.846128	0.739520	0.656566	64	elu	0.50	<class 'keras.optimizers.legacy.gradient_desce...	0.50

Figure 7. Visualization of the TALOS experiment



Table 5. Overview of the literature survey and comparison with proposed work

Method	Dataset	DL models	Optimizer/Technique	Accuracy	Summary
[17]	Reuters (11228 images), Fashion MNIST (70000 images)	Customized deep NN	WOA	89.85% (Fashion MNIST), 80.60%(Reuters)	Can be sensitive to initialization, might get stuck in local optima.
[18]	147,500 images of plant leaf classes	14-DCNN	Random search	99.97%	Inefficient, often misses optimal hyperparameter combinations.
[19]	18159 images of tomato leaf	Deep NN	WOA with hybrid PCA	86%(testing)	Similar to WOA, can be sensitive to initialization.
[20]	240,000 augmented images plants	DCNN	Random search	98.41%	Inefficient, might miss optimal hyperparameter combinations.
[21]	54309 images, PlantVillage dataset	PDI-CMAE	PDI-CMAE-DDAOA	98%	Complex algorithm, computationally expensive.
[22]	1125 images of Swedish leaf dataset, 4503 images of mendeley data	CNN	SS-WOA	97%	Specific to activation function optimization, limited applicability.
[23]	54306 images of PlantVillage, 2598 images of PlantDoc	Xception, DenseNet	AiSara	23%improvement	Specific to certain architectures.
[24]	54309 images of PlantVillage,	ResNet18, AlexNet	Manual fine-tuning hyperparameters	90.31% (AlexNet), 87.6% (ResNet18)	Time consuming, requires expert knowledge.
[25]	10071 images, PlantVillage dataset	SVM	fruit fly (FOA)	91.1%	Converges prematurely to sub optimal solutions, when dealing with complex optimization problems.
Proposed work	3030 images of PlantVillage	CNN	TALOS tool	94.5%	Accelerated hyperparameter tuning, reduced manual intervention, optimize time and cost

Table 6. Comparison of proposed work in terms of filter

Reference	Kernel size	Accuracy (%)	Precision	Recall	F1-score	Parameters	Elapsed time
[14]	3×3	97.41	0.9760	0.9728	0.9744		6hr 28min 23sec
[14]	5×5	97.61	0.9766	0.9758	0.9762	1334533	8hr 24min 3sec
[14]	3×3	99.59	0.9963	0.9984	0.9984		6hr 58min 27sec
[14]	5×5	99.66	0.9968	0.9965	0.9967		8hr 38min 27sec
Proposed work	3x3	94.5	0.9867	0.9867	0.9867	37,647	3hr 18min 45sec

Table 7. Comparison of proposed TALOS-CNN with the existing models

Models	Number of parameters	Epochs required
VGG16	39443043	15
Inception V3 [13]	24,937,283	15
MobileNetV2 [13]	2,422,339	15
14-DCNN [18]	17,928,571	1000
CNN1 [13]	5,108,426	50
CNN2 [13]	494,218	50
Proposed CNN	33647	10

#### 4. CONCLUSION

The deployment of cutting-edge deep CNN models for early plant disease detection has made tremendous strides in recent years. As a result, it has become a prominent and active research field, promising early disease detection and improved precision in managing agricultural diseases. Efficient identification of the diseases results in reduced losses, better crop management and improved food security. However, issues like manual tuning of the hyperparameters, the necessity of large datasets, and high computational complexity persist. This research addresses the earlier concerns by introducing the automated hyperparameter tuning technique employed with a customized CNN model. Trained using a dataset of 3030 tomato leaf images. Firstly, the CNN model is defined efficiently to train with a limited number of images. Next, TALOS is set by specifying the hyperparameters and range, and the experiment is set for different filter sizes, 5×5 and 3×3, with varying dropout rates of neurons. However, TALOS can be computationally intensive, the benefits overcome the drawbacks. Because TALOS narrows the search space of hyperparameters, which enables more effective tuning compared to manual methods and can be adapted to handle large models. This makes the TALOS suitable for real-world applications despite its computing needs. The model achieved an impressive average accuracy of 95.92% with 3×3 filters in classifying healthy and diseased plants using tomato leaf images with comparative parameters 33,647 in a period of 3 hours 28minutes. The batch size, number of training epochs, and dropout rate significantly influenced the model's

performance. Compared to the previous architectures, the proposed design substantially reduces the number of parameters. Future research should focus on extending the work to various deep-learning architectures with more classes in different crops to improve the models' robustness and dependability in the future to various deep-learning architectures with more classes in different crops.




## REFERENCES

- [1] R. Gowthami, N. Sharma, R. Pandey, and A. Agrawal, "Status and consolidated list of threatened medicinal plants of India," *Genetic Resources and Crop Evolution*, vol. 68, no. 6, pp. 2235–2263, Aug. 2021, doi: 10.1007/s10722-021-01199-0.
- [2] E. Nic Lughadha *et al.*, "Extinction risk and threats to plants and fungi," *Plants People Planet*, vol. 2, no. 5, pp. 389–408, Sep. 2020, doi: 10.1002/ppp3.10146.
- [3] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma, "A survey of content-based image retrieval with high-level semantics," *Pattern Recognition*, vol. 40, no. 1, pp. 262–282, Jan. 2007, doi: 10.1016/j.patcog.2006.04.045.
- [4] M. K. R. Gavhale and P. U. Gawande, "An overview of the research on plant leaves disease detection using image processing techniques," *IOSR Journal of Computer Engineering*, vol. 16, no. 1, pp. 10–16, 2014, doi: 10.9790/0661-16151016.
- [5] G. Dhingra, V. Kumar, and H. D. Joshi, "Study of digital image processing techniques for leaf disease detection and classification," *Multimedia Tools and Applications*, vol. 77, no. 15, pp. 19951–20000, = 2018, doi: 10.1007/s11042-017-5445-8.
- [6] Y. Li, J. Nie, and X. Chao, "Do we really need deep CNN for plant diseases identification?," *Computers and Electronics in Agriculture*, vol. 178, p. 105803, Nov. 2020, doi: 10.1016/j.compag.2020.105803.
- [7] G. Saleem, M. Akhtar, N. Ahmed, and W. S. Qureshi, "Automated analysis of visual leaf shape features for plant classification," *Computers and Electronics in Agriculture*, vol. 157, pp. 270–280, Nov. 2019, doi: 10.1016/j.compag.2018.12.038.
- [8] L. Wang, J. Zhao, and R. Mortier, "Neural network," in *Power Systems*, vol. 28, 2007, pp. 75–159.
- [9] J. Wang, Z. Mo, H. Zhang, and Q. Miao, "Ensemble diagnosis method based on transfer learning and incremental learning towards mechanical big data," in *Measurement: Journal of the International Measurement Confederation*, vol. 155, 2020.
- [10] G. Geetharamani and A. P. J., "Identification of plant leaf diseases using a nine-layer deep convolutional neural network," *Computers and Electrical Engineering*, vol. 76, pp. 323–338, Jun. 2019, doi: 10.1016/j.compeleceng.2019.04.011.
- [11] V. Ferrari, C. Sminchisescu, M. Hebert, and Y. Weiss, "ShuffleNet V2: practical guidelines for efficient CNN architecture design," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11218, pp. vii–ix, 2018, doi: 10.1007/978-3-030-34201264-9.
- [12] S. I. Prottasha and S. M. S. Reza, "A classification model based on depthwise separable convolutional neural network to identify rice plant diseases," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 4, pp. 3642–3654, Aug. 2022, doi: 10.11591/ijece.v12i4.pp3642-3654.
- [13] H. Ulutaş and V. Aslantaş, "Design of efficient methods for the detection of tomato leaf disease utilizing proposed ensemble CNN model," *Electronics (Switzerland)*, vol. 12, no. 4, p. 827, Feb. 2023, doi: 10.3390/electronics12040827.
- [14] R. Valarmathi and T. Sheela, "Heart disease prediction using hyper parameter optimization (HPO) tuning," *Biomedical Signal Processing and Control*, vol. 70, p. 103033, Sep. 2021, doi: 10.1016/j.bspc.2021.103033.
- [15] X. Liu and C. Wang, "An empirical study on hyperparameter optimization for fine-tuning pre-trained language models," in *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference, 2021*, pp. 2286–2300, doi: 10.18653/v1/2021.acl-long.178.
- [16] B. Zhang *et al.*, "On the importance of hyperparameter optimization for model-based reinforcement learning," *Proceedings of Machine Learning Research*, vol. 130, pp. 4015–4023, Feb. 2021, [Online]. Available: <http://arxiv.org/abs/2102.13651>.
- [17] A. Brodzicki, M. Piekarski, and J. Jaworek-Korjakowska, "The whale optimization algorithm approach for deep neural networks," *Sensors*, vol. 21, no. 23, p. 8003, Nov. 2021, doi: 10.3390/s21238003.
- [18] J. A. Pandian, V. D. Kumar, O. Geman, M. Hnatiuc, M. Arif, and K. Kanchanadevi, "Plant disease detection using deep convolutional neural network," *Applied Sciences (Switzerland)*, vol. 12, no. 14, p. 6982, Jul. 2022, doi: 10.3390/app12146982.
- [19] T. R. Gadekallu *et al.*, "A novel PCA-whale optimization-based deep neural network model for classification of tomato plant diseases using GPU," *Journal of Real-Time Image Processing*, vol. 18, no. 4, pp. 1383–1396, Aug. 2021, doi: 10.1007/s11554-020-00987-8.
- [20] J. A. Pandian *et al.*, "A five convolutional layer deep convolutional neural network for plant leaf disease detection," *Electronics (Switzerland)*, vol. 11, no. 8, p. 1266, Apr. 2022, doi: 10.3390/electronics11081266.
- [21] M. Prasannakumar and K. Latha, "Plant disease identification using contextual mask auto-encoder optimized with dynamic differential annealed optimization algorithm," *Microscopy Research and Technique*, vol. 87, no. 3, pp. 484–494, Mar. 2024, doi: 10.1002/jemt.24451.
- [22] B. Dudi and V. Rajesh, "Optimized threshold-based convolutional neural network for plant leaf classification: a challenge towards untrained data," *Journal of Combinatorial Optimization*, vol. 43, no. 2, pp. 312–349, 2022, doi: 10.1007/s10878-021-00770-w.
- [23] A. Halim, C. Chow, M. Amabel, S. Achmad, and R. Sutoyo, "The impact of hyperparameter tuning in convolutional neural network on image classification model: a case study of plant disease detection," in *2023 5th International Conference on Cybernetics and Intelligent Systems, ICORIS 2023*, Oct. 2023, pp. 1–6, doi: 10.1109/ICORIS60118.2023.10352209.
- [24] E. Akkuş, U. Bal, F. Ö. Koçoğlu, and S. Beyhan, "Hyperparameter optimization of pre-trained convolutional neural networks using adolescent identity search algorithm," *Neural Computing and Applications*, vol. 36, no. 4, pp. 1523–1537, Feb. 2024, doi: 10.1007/s00521-023-09121-8.
- [25] E. Gangadevi, R. S. Rani, R. K. Dhanaraj, and A. Nayyar, "Spot-out fruit fly algorithm with simulated annealing optimized SVM for detecting tomato plant diseases," *Neural Computing and Applications*, vol. 36, no. 8, pp. 4349–4375, Mar. 2024, doi: 10.1007/s00521-023-09295-1.
- [26] R. Chauhan, K. K. Ghanshala, and R. C. Joshi, "Convolutional neural network (CNN) for image detection and recognition," in *ICSCCC 2018 - 1st International Conference on Secure Cyber Computing and Communications*, Dec. 2018, pp. 278–282, doi: 10.1109/ICSCCC.2018.8703316.
- [27] O. C. Omankwu, M. C. Okoronkwo, and Kanu, "Cardiovascular (heart) diseases prediction using deep learning neural network model," *Journal of Science and Technology Research*, vol. 6, no. 1, pp. 2024–2052, 2024, [Online]. Available: <https://doi.org/10.5281/zenodo.10802375>.
- [28] S. P. Singh, K. Pritamdas, K. J. Devi, and S. D. Devi, "Custom convolutional neural network for detection and classification of rice plant diseases," *Procedia Computer Science*, vol. 218, pp. 2026–2040, 2022, doi: 10.1016/j.procs.2023.01.179.




- [29] S. Verma, A. Chug, and A. P. Singh, "Impact of hyperparameter tuning on deep learning based estimation of disease severity in grape plant," in *Advances in Intelligent Systems and Computing*, vol. 978 AISC, 2020, pp. 161–171.
- [30] A. L. C. Ottoni, A. M. Souza, and M. S. Novo, "Automated hyperparameter tuning for crack image classification with deep learning," *Soft Computing*, vol. 27, no. 23, pp. 18383–18402, Dec. 2023, doi: 10.1007/s00500-023-09103-x.

## BIOGRAPHIES OF AUTHORS






**Shruthi Kikkeri Subramanya**    completed her M.Tech degree in Digital Electronics and Communication at Malnad College of Engineering, Hassan, under Visveswaraya Technological University, Belgaum, with a dissertation titled "Text and Image Encryption based on Session". And Published 2 International journals on that. She had 4.5 years of experience as an assistant professor, Currently, she is a full-time research scholar at BGSIT, Adichunchanagiri University, B.G. Nagara, Karnataka, India since 2021, focusing her research on AI, particularly deep learning applied to agriculture, specifically in Plant Pathology. She published a paper in a IEEE Conference. She can be contacted at email: shruthibgsit@gmail.com.



**Dr. Naveen Bettahalli**    is working as an associate professor at Electronics and Communication Engineering department of BGS Institute of Technology, Adichunchanagiri University, B G Nagara, Mandya. He also worked as a Head of the department at Electronics and Communication Engineering of Jnanavikas Institute of Technology, Bangalore. He has 15 Experience includes 13 years in Teaching and 02 years of Industry. He has Contributed 26 referred article, including scopus indexed Journals, conferences, proceedings and book Chapters published by international and national publishers. He has applied for 05 patents. He has received sponsored research projects from VGST, AICTE. He is currently guiding 06 Ph.D. students and more than 15 M.Tech dissertation in the area of image processing, VLSI and embedded systems, artificial Intelligence. He can be contacted at email: naveenb@bgsit.ac.in.



**Dr. Naveen Kalenahalli Bhoganna**    is Professor in the Electronics and Communication Eng. BGSIT, Adichunchanagiri University, B G Nagara, India. He received his Ph.D. from VTU with the specialization VLSI design and embedded system. His research areas are VLSI, embedded systems, image processing, communication, and networking. He is a member of various profession bodies like IEEE, IEAE (InSc), ISTE, ISC, and IEL. During the last five years tenure at BGSIT he was leading the targeted interventions, counselling and testing, information, education and communication, mainstreaming, surveillance divisions and various projects being implemented as a part of KSCST and other agencies. He has published more than 50 papers published in various conference and journals. He can be contacted at email: naveenkb@bgsit.ac.in.