Tomato leaf disease detection using Taguchi-based Pareto optimized lightweight CNN

Bappaditya Das, C. S. Raghuvanshi
Department of Computer Science and Engineering, Faculty of Engineering and Technology, Rama University, Kanpur, India

Article Info

Article history:

Received Mar 12, 2024
Revised Sep 30, 2024
Accepted Oct 7, 2024

Keywords:

Deep learning
Hyperparameters tuning
Leaf disease detection
Multiobjective
Taguchi method

ABSTRACT

The prospect of food security becoming a global danger by 2050 due to the exponential growth of the world population. An increase in production is indispensable to satisfy the escalating demand for food. Considering the scarcity of arable land, safeguarding crops against disease is the best alternative to maximize agricultural output. The conventional method of visually detecting agricultural diseases by skilled farmers is time-consuming and vulnerable to inaccuracies. Technology-driven agriculture is an integral strategy for effectively addressing this matter. However, orthodox lightweight convolutional neural network (CNN) models for early crop disease detection require fine-tuning to enhance the precision and robustness of the models. Discovering the optimal combination of several hyperparameters might be an exhaustive process. Most researchers use trial and error to set hyperparameters in deep learning (DL) networks. This study introduces a new systematic approach for developing a less sensitive CNN for crop leaf disease detection by hyperparameter tuning in DL networks. Hyperparameter tuning using a Taguchi-based orthogonal array (OA) emphasizes the S/N ratio as a performance metric primarily dependent on the model's accuracy. The multi-objective Pareto optimization technique accomplished the selection of a robust model. The experimental results demonstrated that the suggested approach achieved a high level of accuracy of 99.846% for tomato leaf disease detection. This approach can generate a set of optimal CNN models' configurations to classify leaf disease with limited resources accurately.

Corresponding Author:

C. S. Raghuvanshi
Department of Computer Science and Engineering, Faculty of Engineering and Technology
Rama University
Kanpur, 209217 Uttar Pradesh, India
Email: drcsraghuvanshi.fet@ramauniversity.ac.in

1. INTRODUCTION

Plants are essential for human civilization as they generate food and provide protection against harmful radiation. Tomato is a popular, nutrient-dense vegetable with pharmacological properties [1]. The extensive use of tomatoes escalates demand, resulting in an annual consumption of about 160 million tons [2]. Tomatoes are a highly profitable crop for agricultural households and can have a significant impact on reducing poverty [3]. As per FAO, plant diseases alone accounted for 14% of agricultural production losses, leading to an annual trade deficit of $200 billion [2]. Timely identification of plant diseases can minimize the use of pesticides, thereby safeguarding consumer health and the environment. Traditional visual diagnosis of pests and pathogens is more time-consuming and complex. Farmers face the formidable challenge of frequently monitoring their plants to prevent the spread of disease. Therefore, developing an automated, rapid, and accurate leaf disease detection system is imperative for the early identification of diseases and holds immense importance.

Convolutional neural networks (CNNs) have emerged as a powerful tool for automated leaf disease detection [4]. Their success in accurately detecting diseases has fuelled a surge in research, focusing on developing novel CNN architectures or applying existing models to various crops [4]–[9]. The extensive cultivation of tomatoes and the availability of large, publicly accessible datasets containing diverse disease categories have made tomato leaf disease detection a popular area of deep learning (DL) research [10]. Both the restructured deep residual dense network (RRDN) model [11] and improved faster region convolutional neural network (Faster RCNN) [12] employed deep residual networks for feature extraction. Researchers have developed several efficient, lightweight CNN models using DL to classify tomato leaf diseases. ToLeD [13], a CNN with a small parameter count of 0.2 M, achieved a maximum testing accuracy of 91%, where validation accuracy was improved by adjusting through hyperparameter tuning of the epoch, batch size, learning rate, dropout rate, number of convolution layers, and pooling layers. The INAR-SSD model [14], [15], combining rainbow concatenation with the SSD algorithm and the Inception module, achieved 98.49% and 78.80% accuracy for classifying five common leaf diseases of tomato and apple, respectively. Bhujel *et al.* [16], developed a 20-layered lightweight CNN model (lw_resnet20_cbam) by incorporating the convolutional block attention module (CBAM), spatial attention (SA), squeeze and excitation (SE), and dual attention (DA) modules into the ResNet-20 architecture to classify tomato leaf diseases. The model attained a Top-1 accuracy of 99.51% with a validation loss of 0.0155. A customized CNN was developed using DenseNet201 as the base architecture, followed by adding three convolutional layers and a flattening layer [17].

This model achieved the highest validation accuracy of 98.26% in diagnosing tomato leaf diseases on the PlantVillage dataset. Hyperparameter tuning was utilized for EfficientNet-based transfer learning to achieve 89% accuracy and 0.235 loss in identifying five classes of cassava leaf diseases [18]. An experimental approach optimized hyperparameters, such as batch size, epochs, learning rate, optimizer, and loss function. Integrating channel, spatial, and pixel attention using ResNet50, multi-feature fusion network (MFFN), and the adaptive attention mechanism achieved 99.8% validation accuracy for tomato leaf disease classification [19]. Trials were conducted for 100 epochs using various combinations of channel attention module (CAM), position attention module (PAM), and cross-position attention module (CPAM) with a batch size of 4 and a fixed learning rate of 0.0003. Optimal batch size and learning rate values can significantly decrease the training time of the model [20], whereas adjusting the ratios of the dataset for training, testing, and validation improves the model's accuracy. Since ResNet50 outperforms visual geometry group (VGG)16 and VGG19 in detecting leaf diseases, an online application [21] for recommending initial treatment by utilizing ResNet50 achieved the highest accuracy of 98.98%. Datasets of varying sizes were used to assess the validation accuracy and loss of the model. The principal component analysis (PCA) technique using VGG16 [22] and the two-stage transfer learning approach employing VGGNet [23] achieve high accuracy in detecting tomato leaf diseases. This research uses semantic segmentation to distinguish precisely between disease-affected and healthy regions. Cutting-edge approaches, such as the proposed U-Net design with skip connections and dilated convolutions, ensure accurate separation. Researchers employed the Taguchi methodology to optimize hyperparameters within a CNN model for accurate breast histopathology image classification [24]. A similar approach was applied to determine the optimal architectural configuration for a DL network for malware detection [25]. Six of the nine control variables were assigned two levels, while the number of filters per convolutional operation was assigned three levels. The authors utilized ANOVA to evaluate model performance and identify significant parameters based on larger-is-better criteria. A generalized Taguchi method was proposed for optimizing hyperparameters in multi-objective CNN models [26]. The method involved defining a performance functional vector, employing extended orthogonal arrays(OAs), and computing a performance index to identify optimal parameter settings.

Optimizing hyperparameters is crucial yet challenging in developing DL models. The traditional trial-and-error method for determining optimal hyperparameter configurations for CNNs is exhaustive and time-consuming. Despite their established impact on DL model performance, optimizing CNN hyperparameters for detecting tomato leaf diseases requires further exploration. This paper proposes a framework that integrates Taguchi-based hyperparameter fine-tuning and multi-objective Pareto optimization to develop a lightweight CNN model for accurately detecting tomato leaf disease. The significant contribution of this research is as:

i) To preprocess the image, we perform various augmentations, such as rotation, scaling, flipping, brightness adjustment, normalization, color enhancement, and noise reduction.

ii) We designed a lightweight CNN model with less than three million parameters, achieving superior accuracy in tomato leaf disease detection compared to previous classical CNN models. This model's memory efficiency makes it suitable for deployment in resource-constrained environments, such as mobile or embedded devices.
iii) We employed a systematic approach to optimize hyperparameters rather than relying on trial and error. In our performance review, we consider validation accuracy and loss factors rather than depending exclusively on the standard S/N ratio based solely on accuracy. We expand the population to a fixed size by adjusting the developed model's hyperparameters.
iv) We developed the most robust and least vulnerable model by making a trade-off between multi-objectives using Pareto front optimization.

The rest of the paper is structured as follows: section 2 outlines the proposed methodology. Section 3 provides detailed explanations and discussions of the research findings. Finally, section 4 concludes with different application areas of our research.

2. METHOD

This comprehensive approach ensures accurate crop disease identification through a streamlined process, as depicted in Figure 1.

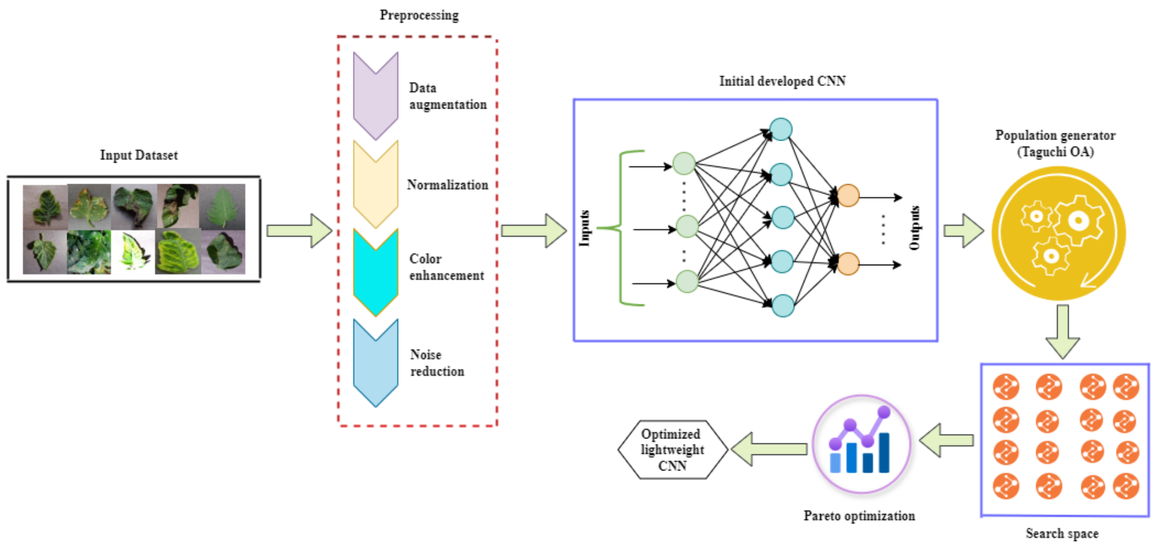

Figure 1. Process flow diagram of proposed model

2.1. Datatset preparation

A total of 18,160 tomato leaf images with laboratory backgrounds were used from 10 classes collected from the publicly available Kaggle dataset. These images were divided into training (13,083), validation (3,265), and testing (1,812) sets. Representative images from each class are shown in Figure 2. To ensure compatibility with our proposed model, all images were resized to 224×224 pixels.

2.2. Data preprocessing

2.2.1. Data augmentation

In preprocessing, data augmentation is a valuable machine learning (ML) technique that combats overfitting by generating diverse modified versions of existing data. This process often applied to images, involves flipping, cropping, rotation, resizing, and more transformations. Random rotation is a common approach that repositions objects in the frame by applying arbitrary clockwise or anticlockwise rotations. Scaling refers to resizing a digital image while maintaining its aspect ratio to ensure it does not appear distorted. Flipping or mirroring pixels horizontally or vertically creates a mirror effect and can increase dataset diversity. Enhancing image brightness through pixel intensity adjustments during preprocessing diversifies data.

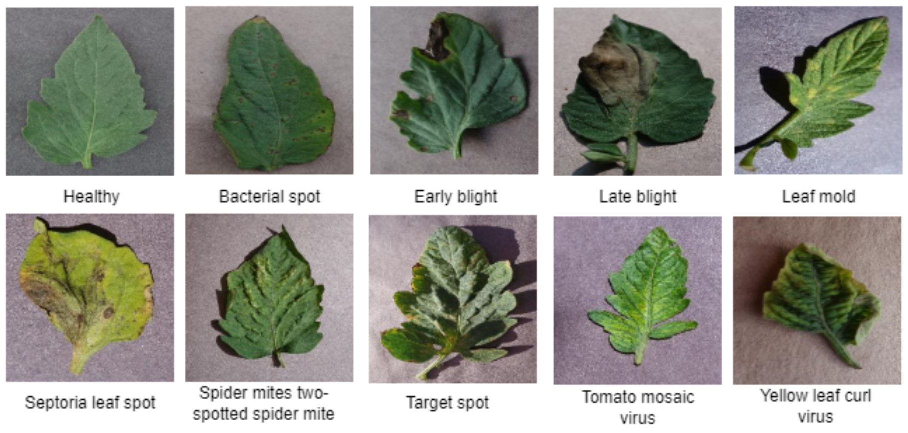

Figure 2. Representative tomato leaf image from each class

2.2.2. Normalization

Normalization in preprocessing adjusts pixel intensity ranges, which is beneficial for improving glare-damaged images by contrast or histogram stretching. It enhances ML algorithm efficiency and accuracy. The normalization process can be mathematically represented as:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

where x is the original pixel value, x_{min} and x_{max} are the minimum and maximum pixel values in the image, respectively, and $x_{norm}$is the normalized pixel value.

2.2.3. Color enhancement

Color enhancement reduces illumination and camera-related variations by enhancing color consistency. The specialized algorithms correct color discrepancies to improve data quality, aiding in accurate disease detection on agricultural crop images.

2.2.4. Noise reduction

Noise reduction, a typical digital image processing task, removes unwanted pixel value fluctuations, enhancing clarity and aiding visual analysis, often using filters and Gaussian blur. A Gaussian blur, achieved through a Gaussian function, is a standard graphics effect that reduces visual noise and detail. It differs significantly from bokeh or shadows, creating a smooth, translucent screen-like appearance. A Gaussian blur applies a weight to nearby pixels based on the two-dimensional Gaussian function given by (2).

$$g(X, Y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{X^2+Y^2}{2\sigma^2}} \tag{2}$$

Where X represents the horizontal axis, Y the vertical axis, and σ the standard deviation in a Gaussian distribution. The Gaussian function peaks at (0,0), and its magnitude diminishes with increasing X or Y.

2.3. Proposed lightweight CNN

Our proposed lightweight CNN architecture leverages efficient building blocks to achieve accurate tomato leaf disease classification. The architecture incorporates five distinct block types: ConvBlock, InceptionBlock, FireBlock, GhostBlock, and AttentionBlock as shown in Figure 3. A ConvBlock is a fundamental block consisting of three stacked ConvModules. Each ConvModule utilizes a two-dimensional (2D) convolutional layer followed by a 2D max-pooling layer. The convolutional layer extracts features by applying trainable filters to the input image, generating unique feature maps for different image locations. The subsequent max-pooling layer downsamples the feature maps while retaining the most significant information by selecting the maximum value within non-overlapping regions. This downsampling reduces model parameters, improves translation invariance, and promotes spatial regularization to mitigate overfitting.

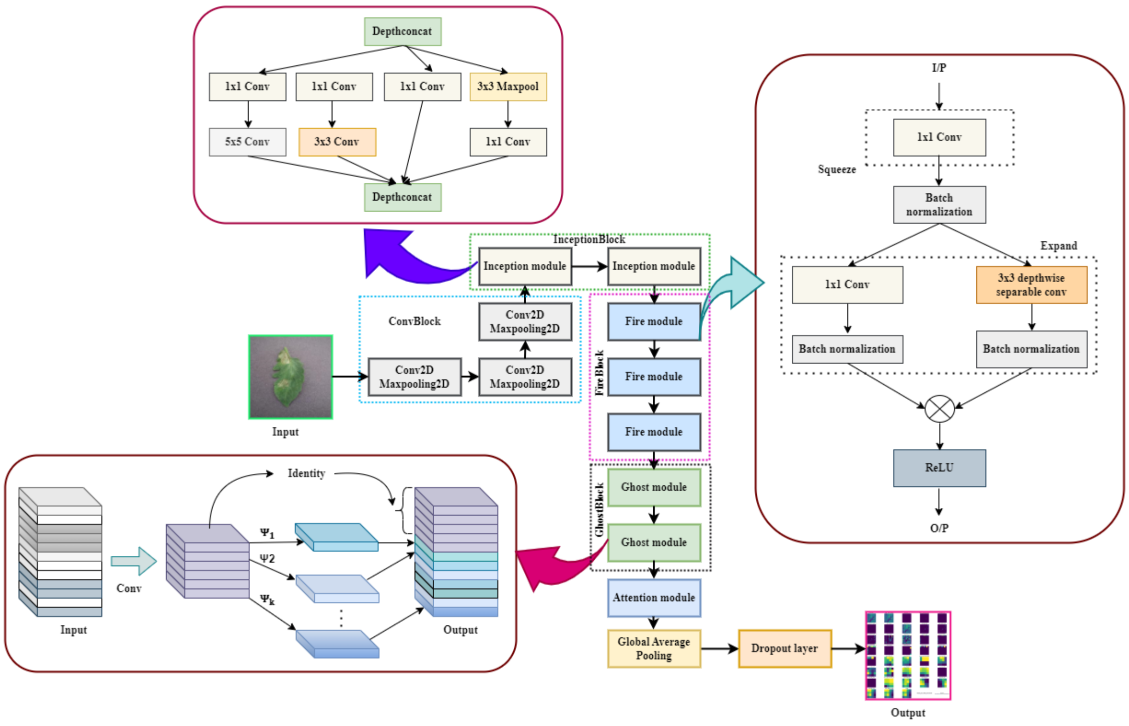

Figure 3. Proposed lightweight CNN architecture detailing inception, fire, and ghost modules

The InceptionBlock receives the output from the ConvBlock and comprises two consecutive inception modules. While structurally similar to the modules used in GoogLeNet, our implementation utilizes varying kernel sizes and filter quantities in the first inception module to enable feature learning across multiple scales. This approach enhances model accuracy by mitigating the vanishing gradient problem, a common issue in deep neural networks. Additionally, a 1×1 convolutional filter within the inception module allows the network to learn patterns across the entire image depth, reducing feature map dimensionality.

The FireBlock comprises three fire modules in series, receiving an output from the InceptionBlock as input. FireBlock expands the channel depth by 12 on input. Each of the initial two fire modules stretches the input channel depth by 4 of their input, while the third one by 3. The fire module primarily focuses on optimizing computation and extracting features in an efficient way. The fire module increases the number of channels of input feature maps while preserving height and width. The first fire module in our architecture transforms feature maps from (27, 27, 32) to (27, 27, 128). MaxPooling2D of ConvModule reduces feature map dimensions by downscaling and preserving essential features while reducing the computation and memory requirements. The Convolution2D operation in ConvModule generates multiple feature maps that capture different patterns without altering the spatial dimensions. As a result, the ConvModule reduces the dimensions of the input feature maps from (27, 27, 128) to (14, 14, 96). The input of the next fire module will be feature maps with dimensions (14, 14, 96). Thus, a ConvModule acts as a bridge between two sequentially connected fire modules. Our modified approach has allowed us to reduce the input dimension by 50%, leading to remarkable efficiency in learning high- and low-level input features. The GhostBlock, composed of two sequentially connected ghost modules [27], efficiently generates additional feature maps through linear operations. Each ghost module first employs standard convolutions followed by linear transformations to produce additional feature maps.

Attention module enables the network to focus on the most critical features and generates a feature map. Global average pooling is a process that reduces the spatial dimension of the feature map generated by the AttentionBlock and converts it into a fixed-length feature vector. Each element of the vector is assigned a channel wise singular value. A dense layer with the rectified linear unit (ReLU) activation function is added following the global average pooling operation. The ReLU activation function helps to speed up the training process by reducing the likelihood of vanishing gradients. The dropout layer follows the dense layer. The dropout layer randomly drops neurons after each iteration to prevent over-reliance on specific features.

2.4. Hyperparameters optimization

2.4.1. Control factors and level selection

This study examined the influence of six key hyperparameters on a CNN model's performance: epochs, learning rate, batch size, optimizer, number of neurons in the final dense layer, and dropout rate. We assigned a discrete set of levels to each hyperparameter to enable a systematic evaluation. All hyperparameters, except dropout rate, were assigned equal levels to ensure a balanced exploration of the hyperparameter space. Utilizing the data presented in Table 1 and the formulas in section 2.4.2, we constructed an effective OA, as shown in Table 2. This array allows us to assess every possible combination of hyperparameter values systematically.

Table 1. Level of hyperparameters and corresponding values

Level	Hyperparameters					
	Neurons at dense layer	Epoch	Optimizer	Learning rate	Dropout rate	Batch size
Level 1	90	10	RMSprop	0.0001	0.10	16
Level 2	120	25	AdamW	0.0002	0.20	32
Level 3		50	Nadam	0.0003	0.30	64
Level 4		75	Adam	0.0005	0.40	128

2.4.2. Design of orthogonal array

The minimum number of experiments for N number of control factors in the Taguchi method is defined as [18],

$$(DOE)_{min} = \sum_{j=1}^{N} (DOF)_j + 1 \tag{3}$$

and DOF of a control factor with L level is defined as,

$$(DOF)_L = L - 1 \tag{4}$$

In this study, we investigated six variables, comprising five control variables, each with four levels and one binary variable. This experimental setup resulted in a total of 16 degrees of freedom. An $L_{16}(4^5)$ Taguchi OA was employed to explore the design space for the initial five variables efficiently. Subsequently, to accommodate the binary variable and expand the experimental scope, we extended the array to 32 experimental runs. This expansion was achieved by incorporating the two levels of the sixth variable while ensuring a uniform distribution of levels across all parameters. This approach not only satisfied but exceeded the minimum requirements for experimental size, thereby enhancing the statistical robustness of the analysis.

2.4.3. Taguchi method

The Taguchi method, a robust optimization framework developed by Genichi Taguchi, has significantly enhanced product and process quality across various industries [28]. Unlike the exhaustive full factorial approach, Taguchi's methodology reduces the experimental burden while effectively identifying optimal parameter combinations. Validation accuracy and loss are critical metrics for assessing DL model performance. Hyperparameters, such as epoch, learning rate, optimizer, batch size, and dropout, substantially impact these metrics. Traditionally, researchers have relied on time-consuming trial-and-error methods to optimize these hyperparameters. The Taguchi method offers a more efficient alternative using OA to explore the design space systematically. OAs enable the investigation of multiple factors and their interactions with a minimal number of experiments. Although the selection of OAs is influenced by the number of control parameters and their levels [29], researchers can customize the array size to meet specific experimental requirements [30]. The Taguchi method is notably more efficient than the full factorial approach, requiring significantly fewer experiments ($L \times (P - 1)$ compared to L^P), where L represents the number of levels, and P denotes the number of parameters. This efficiency is particularly advantageous when computational resources or time constraints are limiting factors. The Taguchi method uses the signal-to-noise ratio (S/N) as an optimization criterion, which is defined by (5).

$$\text{S/N} = \frac{\text{Desired signal strength}}{\text{Unwanted noise power}} \tag{5}$$

The S/N value is determined based on the problem type and evaluated using one of three performance criteria: larger-is-better, smaller-is-better, or nominal-is-better. For the larger-is-better criterion, the S/N ratio is given by:

$$\eta_l = (S/N)_l = -10 \log \left(\frac{1}{n} \sum_{i=1}^{n} \frac{1}{y_i^2} \right) \tag{6}$$

for the smaller-is-better criterion, the S/N ratio is:

$$\eta_s = (S/N)_s = -10 \log \left(\frac{1}{n} \sum_{i=1}^{n} y_i^2 \right) \tag{7}$$

for the nominal-is-better criterion, the S/N ratio is:

$$\eta_a = (S/N)_a = 10 \log \left(\frac{\overline{y}^2}{s_y^2} \right) \tag{8}$$

here, y_i represents the outcome of the i-th run of a collection of n observations, $\overline{y}^2$ denotes the mean squared response, and s_y^2 is the variance.

2.4.4. Pareto optimization

Pareto optimization, also known as Pareto front optimization, is a technique for multi-objective optimization in various fields, including engineering and mathematics. Pareto dominance is the key concept in Pareto front optimization. The domination between two solutions is defined as [31], [32]: A solution P1 is considered to dominate another solution P2 if and only if both of the following conditions are satisfied:

a) The solution P1 is superior or equal to P2 in all objectives.
b) The solution P1 is superior to P2 in at least one objective. The non-dominant points are represented as a non-domination front.

In Figure 4, the curve passing through P_3, P_5, and P_6 labeled as "Non-dominated front" of the graph with two conflicting objectives - f_1 and f_2 respectively.

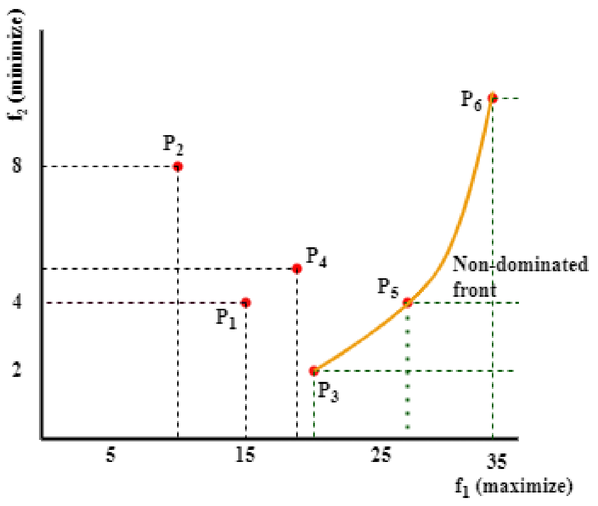

Figure 4. A set of points along with the first non-dominated front

2.4.5. Proposed Taguchi-based Pareto front optimization algorithm

The flowchart of our proposed algorithm is shown in the Figure 5. In our Algorithm 1, the size of the OA(R) depends solely on the number of control factors (P) and the levels for each control factor (levelFactor). The proposed Taguchi-based Pareto front optimization (TPFO) takes a parameter named totalTrials equal to R.

The value of R is derived using the formula and technique explained in section 2.4.2, levelFactor $= [\ell_{jk} \mid j \in \{1, \ldots, P\}, k \in \{1, \ldots, L_i\}]$ be a set of arrays where ℓ_{jk} refers to the k-th level of the j-th factor. The function CREATE_OA generates an OA of size R (L^R) with P number of control factors and levelFactor. The literature review established the function arguments. A vector with P components represents each input for P hyperparameters and is mathematically represented as,

$$H = [\ell_{h1}(\tau), \ell_{h2}(\tau), \ldots, \ell_{hP}(\tau)]$$

where $\ell_{hi}(\tau)$ represents the level of the i-th hyperparameter at the τ-th trial. Each SETFACTOR operation uses a unique combination of parameter levels from the Taguchi table. CONDUCT_EXP runs the experiment using that setting as input and records the outcomes in $\mathbb{T}$. $\mathbb{T}$ is a 2D table with R (total number of trials/runs) rows and two columns for storing validation accuracy and loss values. For instance, $\mathbb{T}[\tau][1]$ and $\mathbb{T}[\tau][2]$ contain the validation loss and accuracy for the τ-th trial. The SORT and FILTER functions arrange $\mathbb{T}$'s validation accuracy records in descending order and discard those records below a user-defined threshold.

Create a scatter plot of filtered records where each point P_τ represents a feasible solution in the objective space defined by objective 1 and objective 2 for the τ-th entry of the OA. FIND_SOLUTIONS compares each solution with every other solution to determine whether any other solution dominates. Add the solution to the Best_Response set only if no other solutions dominate it. H^* is a set of combinations of hyperparameters considered the best possible settings for achieving optimal results, i.e., $H^* \subseteq H$. Highlight the Pareto front by sketching non-dominated solutions.

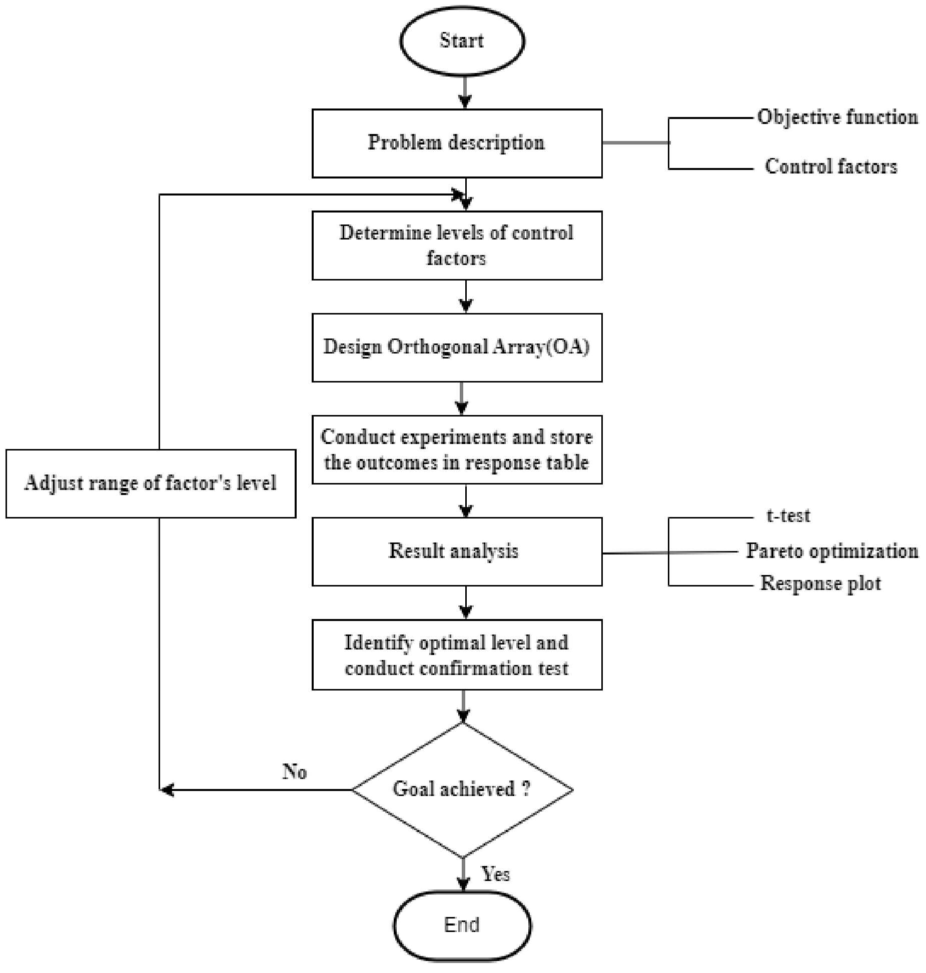
Start
Problem description
Objective function
Control factors
Determine levels of control factors
Design Orthogonal Array(OA)
Conduct experiments and store the outcomes in response table
Adjust range of factor's level
Result analysis
t-test
Pareto optimization
Response plot
Identify optimal level and conduct confirmation test
Goal achieved ?
No
Yes
End

Figure 5. Flowchart of Taguchi-based pareto optimized CNN

Algorithm 1. TPFO for hyperparameters tuning

```
 1: Procedure TPFO(totalTrials)
 2: Declare $\tau$, $\mathbb{T}[1..R, 1..2]$, $P$, $LP$, Threshold
 3: Input: $H = \{[\ell_{h1}(\tau), \ell_{h2}(\tau), \ldots, \ell_{hP}(\tau)], 1 \leq \ell_{hi}(\tau) \leq L_i$ and $1 \leq \tau \leq R$, following Taguchi orthogonal array$\}$
 4: Initialize $totalTrials \leftarrow R$, $\tau \leftarrow 0$, $H^* \leftarrow \varnothing$, $Best\_Response \leftarrow \{[-\infty, +\infty]\}$
 5: Set $numFactors \leftarrow P$, $levelFactor \leftarrow [L_1, L_2, L_3, \ldots, L_P]$
 6: // Create a Taguchi orthogonal array with $R$ number of rows //
 7: $LR \leftarrow CREATE\_OA(P, levelFactor)$
 8: // Perform the experiments for each trial //
 9: while $\tau < R$ do
10:     SET_FACTORS($H_\tau$)
11:     $[ValAccuracy, ValLoss] \leftarrow CONDUCT\_EXP(\tau)$
12:     $\mathbb{T}[\tau, 1] \leftarrow ValAccuracy$
13:     $\mathbb{T}[\tau, 2] \leftarrow ValLoss$
14:     $\tau \leftarrow \tau + 1$
15: end while
16: SORT_DESCEND($\mathbb{T}.ValAccuracy$) {Sort the table by ValAccuracy in descending order}
17: FILTER($\mathbb{T}$) based on ($\mathbb{T}.ValAccuracy \geq Threshold$)
18: PLOT((Filtered($\mathbb{T}$)))
19: $Best\_Response \leftarrow Best\_Response \cup \{FIND\_SOL(Filtered(\mathbb{T}))\}$
20: $H^* \leftarrow H^* \cup \{H$ associated with $Best\_Response\}$
21: Draw Pareto front.
```

3. RESULTS AND DISCUSSION

We performed our investigations on a laptop equipped with an AMD Ryzen 5 5600H processor, an NVIDIA GeForce GTX 1650 GPU, and a 64-bit Windows 11 operating system. TensorFlow with Keras in Python 3.9.12 was the DL framework, utilizing CUDA 11.6 for GPU acceleration. Additionally, we benefited from the high-performance GPU P100 setup offered by Kaggle accelerators for computationally intensive tasks.

An experimental design based on a Taguchi OA was employed to investigate the influence of hyperparameters on model performance. The results, presented in Table 2, stem from multiple trials using two distinct models with varying hyperparameters. Each trial involved different settings for the number of neurons in the dense layer, epochs, optimizer, learning rate, dropout rate, and batch size. The primary objective of these trials was to identify the most effective combinations of these hyperparameters. We conducted paired and unpaired t-tests to assess the impact of the number of neurons on model performance. The results, summarized in Table 3, indicate no statistically significant differences between model 1 and model 2 regarding validation accuracy and loss. We have used the Pearson correlation coefficient to analyze the linear relationship between numeric hyperparameters and model performance. The optimizer, being a categorical variable, was excluded from this analysis.

The results, visualized in Figure 6, revealed a strong positive correlation between epochs and learning rate with validation accuracy (r = 0.7476 and r = 0.7370, respectively). These r values indicate that increases in these hyperparameters are associated with improved model performance. Conversely, epochs and learning rate exhibited the strongest negative correlation with validation loss (r = -0.7334 and r = -0.7289, respectively), suggesting that increasing these hyperparameters leads to a decline in model error. Both hyperparameters exhibited statistically significant (p-value<0.05) positive correlations with validation accuracy and negative correlations with validation loss. Dropout showed negligible correlations with validation accuracy (r = -0.2518, p = 0.1795) and validation loss (r = 0.2748, p = 0.1416), suggesting that dropout may not have been a critical factor in improving model performance within the explored parameter space. Similarly, batch size had minimal impact on validation accuracy (r = -0.2161, p = 0.2514) and validation loss (r = 0.2118, p = 0.2612), indicating that the range of batch sizes tested had a negligible effect on model generalization. The results indicate that epochs and learning rate are the most critical hyperparameters influencing model performance. Careful tuning of these parameters can lead to significant improvements in validation accuracy and reductions in validation loss. Dropout and batch size, within the ranges explored, have minimal impact on the model's performance.

We accomplished a comparative analysis of four optimizers (Adam, Nadam, AdamW, and RMSprop) to evaluate their impact on model performance. Adam demonstrated superior performance across all metrics, achieving the highest validation accuracy (99.794%) with the lowest validation loss (0.00653). Nevertheless, it displayed a broader range of performance, indicating a possible sensitivity to hyperparameter settings or dataset characteristics. Nadam exhibited exceptional performance, achieving the highest accuracy of 99.846% and the lowest loss of 0.00703. While its performance was consistent, it did not consistently surpass Adam. AdamW and RMSprop generally underperformed compared to Adam and Nadam. The performance of RMSprop was characterized by the broadest range of outcomes, suggesting potential instability. Table 2 illustrates that over 93% of the trials achieved an accuracy exceeding 90%, with 30% of cases surpassing 99% accuracy. We have selectively presented data points with validation accuracy greater than 99% to visualize top-performing models. Trials 15 and 32 demonstrated exceptional performance, achieving validation accuracies of 99.846% and 99.794%, respectively, with corresponding losses of 0.00703 and 0.00653. Both trials belong to the first non-dominated Pareto front as shown in Figure 7, providing options for selecting optimal models. The choice of optimizer significantly impacts model performance, with Adam demonstrating superior overall results. However, its sensitivity suggests that it may only be universally optimal for some scenarios. Nadam emerged as a reliable alternative, balancing performance and stability. RMSprop and AdamW generally underperformed compared to Adam and Nadam. Future research should focus on expanding the dataset and exploring a broader range of hyperparameter values. Additionally, investigating adaptive optimization techniques that integrate the strengths of various optimizers could be a promising direction for future work.

Table 2. Performance evaluation of hyperparameter combinations using OA with multiple objectives

Trials	Model	Hyperparameters						Objectives	
		Neurons at dense layer	Epoch	Optimizer	Learning rate	Dropout	Batch size	Validation accuracy	Validation loss
1	Model 1	1	1	1	1	1	1	94.425%	0.18000
2		1	1	2	2	2	2	93.016%	0.19360
3		1	1	3	3	3	3	93.690%	0.19640
4		1	1	4	4	4	4	88.730%	0.32100
5		1	2	1	2	3	4	96.202%	0.11780
6		1	2	2	1	4	3	95.160%	0.13580
7		1	2	3	4	1	2	96.018%	0.11360
8		1	2	4	3	2	1	97.672%	0.07540
9		1	3	1	3	4	2	97.978%	0.06000
10		1	3	2	4	3	1	93.720%	0.18100
11		1	3	3	1	2	4	98.890%	0.03845
12		1	3	4	2	3	1	99.264%	0.02160
13		1	4	1	4	2	3	98.500%	0.04890
14		1	4	2	3	1	4	99.660%	0.01350
15		1	4	3	2	4	1	99.846%	0.00703
16		1	4	4	1	3	2	99.755%	0.00700
17	Model 2	2	1	1	1	1	1	93.660%	0.17460
18		2	1	2	2	2	2	95.069%	0.14500
19		2	1	3	3	3	3	95.038%	0.13590
20		2	1	4	4	4	4	87.351%	0.34750
21		2	2	1	2	3	4	92.890%	0.19380
22		2	2	2	1	4	3	97.703%	0.06658
23		2	2	3	4	1	2	96.477%	0.09570
24		2	2	4	3	2	1	97.152%	0.07570
25		2	3	1	3	4	2	97.700%	0.07362
26		2	3	2	4	3	1	95.957%	0.17560
27		2	3	3	1	2	4	99.173%	0.02460
28		2	3	4	2	1	3	99.387%	0.01880
29		2	4	1	4	2	3	98.652%	0.04483
30		2	4	2	3	1	4	99.690%	0.01010
31		2	4	3	2	4	1	99.720%	0.01160
32		2	4	4	1	3	2	99.794%	0.00653

Table 4 presents a comparative analysis of various CNN models for tomato leaf disease classification based on the number of trainable parameters and achieved accuracy. The proposed TPFO CNN outperforms all other models with a validation accuracy of 99.84% while maintaining a reduced number of trainable parameters (<3 M). Integrating an attention mechanism effectively captures relevant features, contributing to enhanced performance. In contrast, LMBRNet [9] achieves a high accuracy of 99.70% but with a larger parameter count.

Models like DenseNet121 [5] and SECNN [7] demonstrate strong performance but require significantly more parameters. The trade-off between accuracy and efficiency is evident, with models like LeNet offering minimal parameters but compromising accuracy. Future research should explore strategies to reduce the parameter count of high-performing models further while preserving accuracy. This compact CNN model will improve the practical applicability of CNN models in resource-constrained environments.

Table 3. T-test results: performance comparison of model 1 and model 2

	Validation accuracy		Validation loss	
	Model 1	Model 2	Model 1	Model 2
	96.41% ± 3.13	96.59 % ± 3.31	0.107 ± 0.089	0.100 ± 0.092
SEM	0.783	0.827	0.022	0.023
Sample size (N)	16	16	16	16
Statistical tests	Unpaired t-test	Paired t-test	Unpaired t-test	Paired t-test
t-statistic (t)	0.1585	0.5010	0.2153	0.8134
Degrees of freedom (df)	30	15	30	15
Standard error of difference	1.139	0.360	0.032	0.009
95% confidence interval	[- 2.506, 2.145]	[- 0.948, 0.587]	[- 0.058, 0.0725]	[- 0.011, 0.025]
Two-tailed p value	0.8751	0.6237	0.8310	0.4287

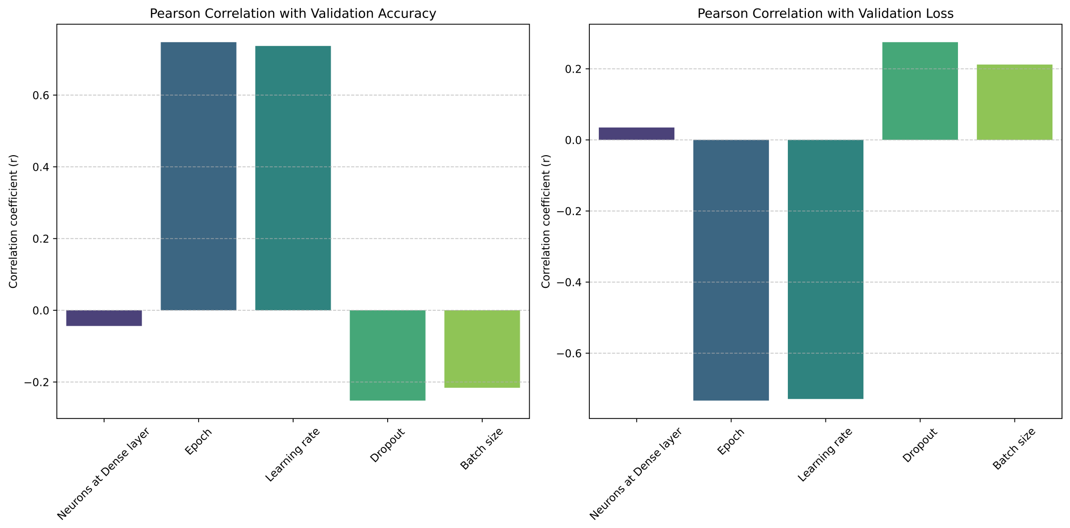

Figure 6. Impact of hyperparameters on model performance

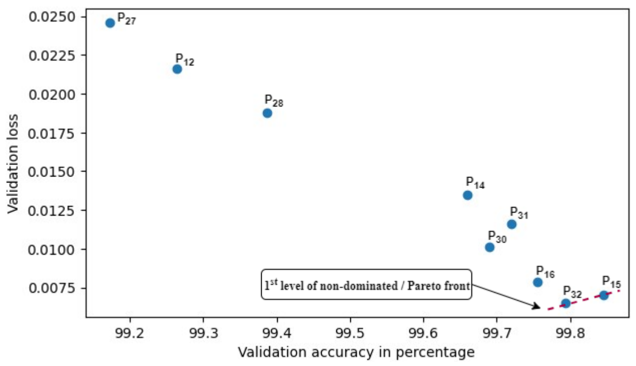

Figure 7. Validation accuracy distribution with threshold (99%)

Table 4. Comparison of trainable parameters and accuracy with existing models

Reference	Technique	Trainable parameters	Accuracy (%)
[5]	DenseNet121	29 M	97.11
[6]	DLMC-Net	6.4 M	96.56
[7]	SECNN	5.4 M	97.90
	AlexNet	60 M	95.65
	ResNet50 + SeNet	27.7 M	96.81
[8]	LeNet	0.06 M	94.85
	DenseNet121_Xception	29.2 M	97.10
	TomConv	3 M	98.19
[9]	LMBRNet	4.1 M	99.70
Proposed model	TPFO CNN	<3 M	99.84

4. CONCLUSION

This research introduces TPFO, a hyperparameter optimization technique, in conjunction with a lightweight CNN architecture to address the challenge of tomato leaf disease classification. The proposed method effectively optimizes model performance by employing a dual-metric approach that considers both validation accuracy and loss, surpassing the limitations of single-metric optimization strategies. Integrating Taguchi's OAs significantly accelerates the hyperparameter search process, enabling efficient exploration of the hyperparameter space. The resulting TPFO CNN model exhibits exceptional performance, achieving a remarkable validation accuracy of 99.84% while maintaining a compact architecture with fewer than 3 million trainable parameters. Comparative analyses demonstrate the superiority of TPFO CNN over unoptimized CNNs and traditional optimization methods in terms of accuracy and computational efficiency. These findings highlight the potential of TPFO CNN as a promising solution for practical applications demanding high performance and resource constraints. Future research will focus on expanding the TPFO framework to accommodate multiple performance metrics and exploring its applicability to diverse DL architectures to broaden its impact on various domains.

REFERENCES

[1] P. Schreinemachers, E. B. Simmons, and M. C. S. Wopereis, "Tapping the economic and nutritional power of vegetables," *Global Food Security*, vol. 16, pp. 36–45, 2018, doi: 10.1016/j.gfs.2017.09.005.

[2] M. Stilwell, "The global tomato online news processing in 2018." 2021.

[3] R. Wang, M. Lammers, Y. Tikunov, A. G. Bovy, G. C. Angenent, and R. A. de Maagd, "The rin, nor and Cnr spontaneous mutations inhibit tomato fruit ripening in additive and epistatic manners," *Plant Science*, vol. 294, p. 110436, 2020, doi: 10.1016/j.plantsci.2020.110436.

[4] B. Tugrul, E. Elfatimi, and R. Eryigit, "Convolutional neural networks in detection of plant leaf diseases: a review," *Agriculture (Switzerland)*, vol. 12, no. 8, p. 1192, 2022, doi: 10.3390/agriculture12081192.

[5] A. Abbas, S. Jain, M. Gour, and S. Vankudothu, "Tomato plant disease detection using transfer learning with C-GAN synthetic images," *Computers and Electronics in Agriculture*, vol. 187, p. 106279, 2021, doi: 10.1016/j.compag.2021.106279.

[6] V. Sharma, A. K. Tripathi, and H. Mittal, "DLMC-Net: deeper lightweight multi-class classification model for plant leaf disease detection," *Ecological Informatics*, vol. 75, p. 102025, 2023, doi: 10.1016/j.ecoinf.2023.102025.

[7] B. N. Naik, R. Malmathanraj, and P. Palanisamy, "Detection and classification of chilli leaf disease using a squeeze-and-excitation-based CNN model," *Ecological Informatics*, vol. 69, p. 101663, 2022, doi: 10.1016/j.ecoinf.2022.101663.

[8] P. Baser, J. R. Saini, and K. Kotecha, "TomConv: an improved CNN model for diagnosis of diseases in tomato plant leaves," *Procedia Computer Science*, vol. 218, pp. 1825–1833, 2023, doi: 10.1016/j.procs.2023.01.160.

[9] M. Li, G. Zhou, A. Chen, L. Li, and Y. Hu, "Identification of tomato leaf diseases based on LMBRNet," *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106195, 2023, doi: 10.1016/j.engappai.2023.106195.

[10] W. Shafik, A. Tufail, A. Namoun, L. C. De Silva, and R. A. A. H. M. Apong, "A systematic literature review on plant disease detection: motivations, classification techniques, datasets, challenges, and future trends," *IEEE Access*, vol. 11, pp. 59174–59203, 2023, doi: 10.1109/ACCESS.2023.3284760.

[11] C. Zhou, S. Zhou, J. Xing, and J. Song, "Tomato leaf disease identification by restructured deep residual dense network," *IEEE Access*, vol. 9, pp. 28822–28831, 2021, doi: 10.1109/ACCESS.2021.3058947.

[12] Y. Zhang, C. Song, and D. Zhang, "Deep learning-based object detection improvement for tomato disease," *IEEE Access*, vol. 8, pp. 56607–56614, 2020, doi: 10.1109/ACCESS.2020.2982456.

[13] M. Agarwal, A. Singh, S. Arjaria, A. Sinha, and S. Gupta, "ToLeD: tomato leaf disease detection using convolution neural network," *Procedia Computer Science*, vol. 167, pp. 293–301, 2020, doi: 10.1016/j.procs.2020.03.225.

[14] N. K. Trivedi *et al.*, "Early detection and classification of tomato leaf disease using high-performance deep neural network," *Sensors*, vol. 21, no. 23, p. 7987, 2021, doi: 10.3390/s21237987.

[15] P. Jiang, Y. Chen, B. Liu, D. He, and C. Liang, "Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks," *IEEE Access*, vol. 7, pp. 59069–59080, 2019, doi: 10.1109/ACCESS.2019.2914929.

[16] A. Bhujel, N. E. Kim, E. Arulmozhi, J. K. Basak, and H. T. Kim, "A Lightweight attention-based convolutional neural networks for tomato leaf disease classification," *Agriculture (Switzerland)*, vol. 12, no. 2, p. 228, 2022, doi: 10.3390/agriculture12020228.
[17] D. Gerdan, K. Caner, and M. Vatandaş, "Diagnosis of tomato plant diseases using pre-trained architectures and a proposed convolutional neural network model," *Journal of Agricultural Sciences*, vol. 29, no. 2, pp. 618–629, 2023, doi: 10.15832/ankutbd.957265.
[18] G. Kalyani, K. S. Sudheer, B. Janakiramaiah, and B. N. K. Rao, "Hyperparameter optimization for transfer learning-based disease detection in Cassava Plants," *Journal of Scientific and Industrial Research*, vol. 82, no. 5, pp. 536–545, 2023, doi: 10.56042/jsir.v82i05.1089.
[19] C. Sunil, C. Jaidhar, and N. Patil, "Tomato plant disease classification using multilevel feature fusion with adaptive channel spatial and pixel attention mechanism," *Expert Systems with Applications*, vol. 228, p. 120381, 2023, doi: 10.1016/j.eswa.2023.120381.
[20] M. V. Shewale and R. D. Daruwala, "High performance deep learning architecture for early detection and classification of plant leaf disease," *Journal of Agriculture and Food Research*, vol. 14, no. 6, p. 100675, 2023, doi: 10.1016/j.jafr.2023.100675.
[21] M. M. Islam *et al.*, "DeepCrop: deep learning-based crop disease prediction with web application," *Journal of Agriculture and Food Research*, vol. 14, p. 100764, 2023, doi: 10.1016/j.jafr.2023.100764.
[22] P. Borugadda, R. Lakshmi, and S. Sahoo, "Transfer learning VGG16 model for classification of tomato plant leaf diseases: a novel approach for multi-level dimensional reduction," *Pertanika Journal of Science and Technology*, vol. 31, no. 2, pp. 813–841, 2023, doi: 10.47836/pjst.31.2.09.
[23] T. Sanida, A. Sideris, M. V. Sanida, and M. Dasygenis, "Tomato leaf disease identification via two–stage transfer learning approach," *Smart Agricultural Technology*, vol. 5, p. 100275, 2023, doi: 10.1016/j.atech.2023.100275.
[24] C. J. Lin, S. Y. Jeng, and C. L. Lee, "Hyperparameter optimization of deep learning networks for classification of breast histopathology images," *Sensors and Materials*, vol. 33, no. 1, pp. 315–325, 2021, doi: 10.18494/SAM.2021.3015.
[25] C. J. Lin, X. Y. Lin, and J. Y. Jhang, "Malware classification using a Taguchi-based deep learning network," *Sensors and Materials*, vol. 34, no. 9, pp. 3569–3580, 2022, doi: 10.18494/SAM4044.
[26] S.-G. Wang and S. Jiang, "Optimal hyperparameters and structure setting of multi-objective robust CNN systems via generalized Taguchi method and objective vector norm," *arXiv preprint arXiv:2202.04567*, 2022, doi: 10.48550/arXiv.2202.04567.
[27] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: more features from cheap operations," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 1577–1586, doi: 10.1109/CVPR42600.2020.00165.
[28] S. R. Rao and G. Padmanabhan, "Application of Taguchi methods and ANOVA in optimization of process parameters for metal removal rate in electrochemical machining of Al/5% SiC composites," *International Journal of Engineering Research and Applications (IJERA)*, vol. 2, no. 3, pp. 192–197, 2012.
[29] J. M. Cimbala, "Taguchi orthogonal arrays," *Pennsylvania State University*, pp. 1–3, 2014.
[30] R. N. Kacker, E. S. Lagergren, and J. J. Filliben, "Taguchi's orthogonal arrays are classical designs of experiments," *Journal of Research of the National Institute of Standards and Technology*, vol. 96, no. 5, pp. 577–591, 1991, doi: 10.6028/jres.096.034.
[31] K. Deb, "Multi-objective optimisation using evolutionary algorithms: an introduction," in *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, Springer, 2011, pp. 3–34.
[32] K. Miettinen, *Nonlinear multiobjective optimization*, vol. 12. Springer Science & Business Media, 1999.

BIOGRAPHIES OF AUTHORS

Bappaditya Das is an assistant professor in the Computer Science and Engineering Department at Dr. B.C. Roy Engineering College, Durgapur, West Bengal, India. He is pursuing his Ph.D. in Computer Science and Engineering at Rama University, Kanpur, Uttar Pradesh, India. He received his B.Tech. and M.Tech. degrees in Computer Science and Engineering from the University of Calcutta, West Bengal, India, in 2001 and 2007, respectively. He completed his B.Sc. in Physics (Hons) from the University of Calcutta, West Bengal, India 1998. He received the National Scholarship Scheme award from MHRD, Government of India, for his exceptional academic performance. He has authored and co-authored more than 15 research articles in international journals and conferences with more than 150 citations. His research interests include machine learning, computer vision, deep learning, image processing, WSN, and applications. He can be contacted at email: mail2bappadityadas@gmail.com.

C. S. Raghuvanshi is a Professor and Head of the Computer Science and Engineering Department at Rama University, Uttar Pradesh, Kanpur, India. He achieved remarkable academic accomplishments, including a Ph.D. in Computer Science in 2014 and an M. Tech in Computer Science in 2010. Recognized for his innovative mindset, he has been honored with the title of Innovation Ambassador by the MoE's Innovation Cell. Furthermore, he is a lifelong member of IETE, Bhopal, and holds membership in IEEE. He has developed a concise course focusing on artificial intelligence, machine learning, deep learning, and their practical applications. Furthermore, he has contributed significantly to research by publishing 15 patents (national and international) and copyrights, primarily in Computer Science and Engineering, intending to benefit the nation. He has authored and co-authored over 45 papers in esteemed international journals recognized by SCI, SCOPUS, and UGC Care and has presented his work at various international and national conferences. He can be contacted at email: drcsraghuvanshi.fet@ramauniversity.ac.in.