Efficient Nightshade Crop Leaf Disease Identification: An Optimized CNN Approach with Comprehensive Time and Space Complexity Analysis

*Barkha Joshi, Dr. Hetal Bhavsar

Computer Engineering Dept.
Sardar Vallabhbhai Patel Institute of Technology.Vasad,Gujarat,Anand, India.
Computer Science and Engineering Dept.
The Maharaja Sayajirao University of Baroda.Vadodara,Gujarat,Vadodara, India

*barkhajoshi.comp@svitvasad.ac.in, barkha_ce@yahoo.co.in
hetal.bhavsar-cse@msubaroda.ac.in

Abstract— This study investigated the optimization of convolutional neural network (CNN) models for the efficient identification of leaf diseases in nightshade crops, focusing on tomatoes, potatoes, bell peppers, and eggplants. Through a comprehensive analysis of time and space complexity, various CNN architectures, including AlexNet, VGG, GoogleNet, and custom Nightshade-CNN https://doi.org/10.1007/978-981-99-1431-9_2, https://ijisae.org/index.php/IJISAE/article/view/2461 and Enhanced Nightshade-CNN https://doi.org/10.1007/s00371-023-03127-y models, were evaluated. This study used a large dataset of nightshade leaf images with bacterial, viral, and fungal diseases. Theoretical and empirical analyses consider factors such as the image size, filter size, layer count, and model parameters. Our findings reveal that deeper models, such as VGG and GoogleNet, exhibit higher computational requirements than leaner alternatives. By applying optimization techniques, the performance of CNNs can be enhanced, demonstrating the need to balance model complexity with efficiency. The Nightshade-CNN and Enhanced Nightshade-CNN models, with their reduced layer counts and filter sizes, show promise for achieving efficient and accurate disease identification. This study offers insights into the design of effective and interpretable CNN models for plant disease analysis, ultimately supporting improved crop management and agriculture.

Keywords—Space and Time complexity Parameters, Nightshade-CNN and Enhanced Nightshade-CNN space and time complexity

1. INTRODUCTION

In modern agriculture, early and accurate detection of plant diseases is important to obtain healthy crops and increase crop yield. Nightshade plants, such as tomatoes, potatoes, peppers, and eggplants, are particularly susceptible to many diseases, including blight, early blight, and powdery mildew[8]. Traditional disease identification methods use manual examinations, which are time-consuming, labor-intensive, and error-prone. This has encouraged the development of artificial intelligence technology that uses the power of convolutional neural networks (CNN) to improve the accuracy and efficiency of disease diagnosis. This material is well known for its rod-like characteristics and can be used extensively in plant pathology research. Because CNN can extract hierarchical features from images, they can easily differentiate between normal and diseased tissues. Nevertheless, the application of CNN-based models to real farms requires a good computational understanding. In particular, the temporal and spatial complexity of these models relates to practical requirements, such as computation time, amount of memory used, and efficiency of the model. Owing to the temporal and spatial characteristics of the CNN, diseases on the leaves of eggplants were detected. In addition to conventional models such as AlexNet, VGG, and GoogleNet, research has been conducted to develop specific models such as Nightshade-CNN[1-3] and Enhanced Nightshade-CNN[4]. Based on the theoretical view and actual experience with the given model, it is possible to outline the economic conditions for making a correct diagnosis. The need for disease analysis is to evaluate and compare their performance and productivity based on actual data in order to find ways to minimize the number of inclusions with at least the same level of efficiency.

This comprehensive review aims to provide researchers and practitioners with practical knowledge regarding the use of CNN-based models in agriculture, ultimately promoting sustainable and profitable agriculture. The remainder of this paper is organized as follows. Section 2 covers related work. Section 3 describes the methodology and results, and Section 5 concludes the study. Figure 1 displays the flow of the study in a graphical representation.

Abstract

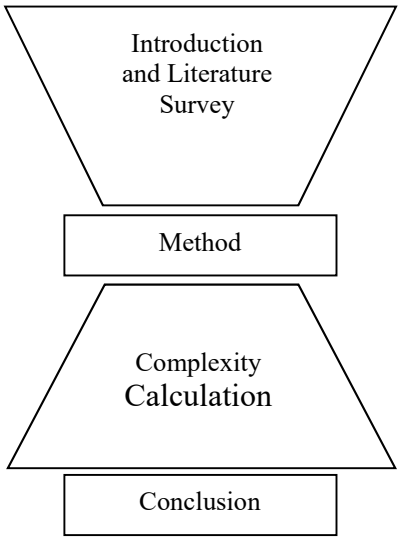

Figure 1 Paper Flow Diagram

2. LITERATURE SURVEY

The use of convolutional neural networks (CNNs) for plant disease identification has advanced significantly, particularly for nightshade crops such as tomatoes, potatoes, and peppers. CNNs, with their robust image recognition capabilities, are well suited for detecting and classifying plant diseases based on leaf images. However, the deployment of CNNs in real-world agricultural applications requires an understanding of their time and space complexities.

Early studies[1] by Krizhevsky et al. introduced AlexNet, a deep CNN architecture that achieves groundbreaking results in image classification. The AlexNet model with five convolutional layers and three fully connected layers proved that deep learning could work, but at the same time, it showed how computationally intensive such models were for training and using them. In the same year, Simonyan and Zisserman [2] developed the VGGNet, which extended the CNN architectures in the use of 16-19 layers with 3 × 3 small filters. Although there is an increase in the accuracy of VGGNet, the same factor has been proven to increase the computational and memory requirements; hence, its applicability in agriculture is limited. The time complexity of a CNN is dependent on the number of layers, size of the filters, and size of the input image. For instance, the AlexNet architecture, although very effective, is computationally intensive owing to its large number of parameters and operations. Nevertheless, VGGNet aggravates this issue because of its deeper architecture, which results in a higher computational complexity.

GoogleNet[3] introduced initial modules that enabled the network to go deeper without a corresponding increase in the computational complexity. However, the time complexity of deeper models is still an issue with real-time applications, as such innovations are still in progress. The spatial complexity of a CNN is defined as the memory required to store the model parameters and temporary feature maps. AlexNet, for example, and VGGNet with millions of parameters consume large amounts of memory. For instance, VGG-16 has approximately 138 million parameters, which are quite large and consume a large amount of storage space and computational power.

Han et al.. proposed deep compression with methods of pruning, quantization, and Huffman coding to decrease the model size and enhance the efficiency at the same rate. These techniques are essential for the execution of CNNs within devices that have a small amount of memory and processing power [4].

Some of the models that have been established are specific to detecting diseases in eggplant crops, and in doing this, it is done in a way that takes less time while at the same time being effective. Deep learning researchers Mohanty et al. utilized a large set of data for training a CNN with improved classification success in different plant diseases, including diseases that affect eggplant crops. Their study made them realize the need to have large and diverse datasets but also pointed out the increased computational needs of large training [5].

The newly proposed Nightshade-CNN and the Improved Nightshade-CNN were designed to achieve better performance with less time and space consumption. These models often employ fewer layers, and the filter size is chosen such that it provides better performance with reduced computational complexity.

Several optimization techniques [6] have been reviewed to improve the ability of CNNs in plant disease diagnosis. Some of the main strategies include the use of the learning rate, mini-batch size, and epoch, which enhances the efficiency of the model. Transfer learning, in which models are fine-tuned for specific tasks, has also been used to save time and computational power.

Advances in hardware accelerators, such as GPUs and TPUs, have greatly improved the feasibility of deploying deep learning models in agricultural settings. These accelerators enable faster computations and efficient memory usage, thereby facilitating real-time disease detection and classification [7].

The proposed Nightshade-CNN[9] model achieves 93-95% accuracy in training and testing, accurately identifying healthy and unhealthy leaves of nightshade crops, classifying unhealthy diseases, and suggesting treatment. It generates fewer parameters and requires less computational power and resources than other standard models such as AlexNet, VGG, and GoogLeNet.

Crop diseases pose a significant threat to food safety; however, their rapid diagnosis remains challenging. Deep-learning models have shown better performance than traditional machine-learning techniques in various fields, including agronomy. This research proposes a model[10] to identify plant leaf diseases with greater accuracy and efficiency than existing approaches. The model, trained on night-shed plants, has nine categorical classes of diseases and healthy leaves. The model achieved a disease classification accuracy rate of 93–95%, indicating its potential to significantly improve the speed and accuracy of identification of disease-infected leaves.

Early detection of plant diseases is crucial for crop health and successful harvest. Advancements in computer vision technology have improved methods, but traditional deep learning algorithms have drawbacks. This research article raises awareness about the Enhance-Nightshade-CNN [11] model, which improves nightshade crop leaf disease detection, achieving 95-100% accuracy.

The paper[13] provides an extensive overview of deep-learning-based blind motion deblurring, focusing on the role of deep learning in this field. It introduces different types of motion blur and highlights the shortcomings of traditional nonblind deblurring algorithms. This study categorizes existing methods based on different backbone networks, including convolutional neural networks, generative adversarial networks, recurrent neural networks, and transformer networks. It also discusses the advantages and limitations of these methods, compares their performance on four widely used datasets, and analyzes current challenges and prospects to drive innovation in image deblurring research.

Time complexity represents [18] the amount of computational resources required for a model to handle input data and generate output predictions, which are typically quantified by the number of mathematical operations involved. In the case of CNNs, time complexity can differ depending on factors like the depth of the network, input data size, and the intricacy of operations within each layer. Deeper CNNs with more layers and larger filters generally have a higher time complexity because they require more operations for convolution, pooling, and activation processes.

It is essential to carefully manage the model complexity in deep learning to ensure effectiveness. If a model is too complex, it may fit the training data exactly, but struggle to perform well on new, unseen data, a problem known as overfitting. Conversely, if a model is too simple, it may fail to capture the key patterns in the data, leading to underfitting. To build models that generalize well and perform effectively in real-world applications, a balance must be established through thoughtful design, experimentation, and the use of regularization techniques[19].

Neural network architectures[20] typically consist of key elements such as convolutional, fully connected, and pooling layers. A critical aspect of evaluating a model's complexity and computational demands is understanding the number of "learnable" parameters in each layer.

The study [21] explored methods to reduce the computation requirements of neural networks and make them suitable for mobile devices. It reviews various techniques, including deep compression, which involves network pruning, quantization, and encoding of network weights. The technique reduces the training time by pruning irrelevant connections, quantizing weights, and using the Huffman encoding algorithm to address storage issues. This approach can potentially improve the performance of neural networks in mobile devices.

The authors of [22] examined the time complexity of two algorithms, KNN and CNN, for character recognition. KNN uses the Euclidean distance between the input images, whereas CNN uses Convolutional Neural Networks (CNN) on Keras and TensorFlow. The first layer of the neural network used 784 neurons, with each neuron generating a 0-9 output. The KNN classifier presented the results, whereas the CNN outperformed the KNN in terms of accuracy.

In [23] a novel graph-based frequency channel selection method was proposed to improve the speed of Convolutional Neural Networks (CNNs) in the compressed domain. This method reduces the computational complexity by retaining important frequency components and eliminating unnecessary layers. The experimental results show that the modified ResNet-50 is 70% faster than the traditional ResNet-50 with a similar classification accuracy. The proposed preprocessing step with partial encoding improves the image distortion resilience.

The Author[24] proposed an algorithm that combines the Winograd minimal filtering and Strassen algorithms to reduce the computational complexity in convolutional neural networks. The algorithm saves 75% of the runtime compared with conventional algorithms, demonstrating its potential for optimal performance.

The model-based comparison chart from the literature survey is presented in Table 1.

Table 1 Model Comparison Chart

Model	**Number Of Convolution Layers**	**Number of Filters**	**Pooling Size**	**Filter Size**	**FC**
AlexNet	5	[96,256]	[3x3]	[11x11,3x3]	3
VGG	16	[64,128,256,512]	[2x2]	[3x3]	3
GoogleNet	22	Varies	[3x3,5x5]	[5x5, 3x3,1x1]	1

Findings from extensive research into the performance of CNN models revealed the following insights.

Research finding:

- Deeper architectures such as VGG and GoogleNet require higher computational times and more memory space than other shallow models such as AlexNet.
- If the model generates fewer parameters, it requires less computation time and storage space.
- If increasing size of datasets and models,

scalability becomes a critical factor.

- Decreasing the number of neurons in the Fully Connected layer can help reduce both time and space complexity.
- Less of an FC layer can help to reduce the time and space complexity.

3. DATASET

This study introduces an economical model designed to identify leaf diseases in preprocessed nightshade crop leaves, including tomatoes, potatoes, bell peppers, and eggplants. The model demonstrated proficiency in detecting diseases caused by fungi, bacteria, and viruses in both plain and complex backgrounds. Nightshade crops, which belong to the Solanaceae family, are vital for their economic, medicinal, and culinary benefits. Potatoes provide significant caloric intake, whereas tomatoes, eggplants, and peppers enhance culinary diversity and nutrition, reinforcing global food security. This study focuses on nightshade crop leaves using RGB image datasets from the Plant Village repository, which offers extensive images of diseased and healthy plants for machine learning model development. Specifically, the research focused on tomato, potato, bell pepper, and eggplant leaves, leveraging a dataset [12] of 25,605 images to aid in early disease detection and classification, ultimately supporting effective disease management and crop yield improvement. These images are used for the diagnostic testing of leaf diseases, particularly focusing on abnormalities. All photographs were taken in sunlight before 5:00 p.m., and the data included 18 groups of diseased and healthy leaves. All images collected had a plain background.

Figure 2 displays the images of healthy and infected nightshade crop leaves on a plain background.

.

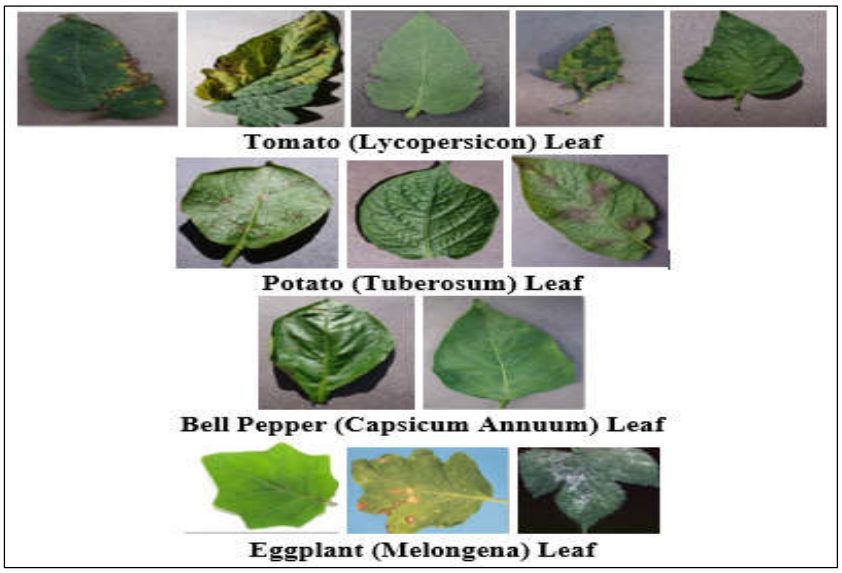

Figure 2 healthy and infected nightshade crop leaf images

Preprocessing is essential as it ensures that the data are in an appropriate form for examination, reduces noise and inconsistencies, and helps machine learning models perform effectively and accurately[8]. Proper preprocessing can improve the quality of the input data, leading to more efficient and effective image classification. While CNNs are powerful in automatically learning features, preprocessing can help optimize the input data for better model performance. Pre-processing techniques, such as rescaling, image resizing, data augmentation, padding, normalization, noise reduction, and handling imbalanced data, have been applied to nightshade crop leaves to improve their quality.

4. METHOD AND RESULT

Factors such as layers, layer operations, filter size, input size, model architecture, hardware acceleration, optimization techniques, and batch size [3] are important for determining the complexity of CNN models. This study examined several parameters that influence the time and space complexity of convolutional neural network (CNN) models, particularly for nightshade crop leaf disease identification.

Parameters for Complexity:

Key factors include the number of parameters, number of layers, interactions within layers, and activation functions: control techniques, architectural choices, input layer, pooling layer, convolutional layer, and fully connected layer. Together, these parameters determine the computational complexity and effectiveness of the CNN models used in agricultural technology.

- **Number of Parameters:** The total number of weights or parameters in the network affects the model complexity. Although more parameters increase the flexibility of the model, if they are not properly regulated, overfitting may occur.
- **Number of Layers:** An important component of complexity is the depth of the neural network, which is based on the number of hidden layers. Although they must be carefully trained and regularized, deeper networks can learn more complex representations.
- **Interaction among Layers:** The quantity and interplay of neurons or units in every layer are other factors that

contribute to complexity. Complex relationships in the data can be captured using a larger number of neurons.

- **Activation Function:** Model complexity may be affected by the selection of activation functions. By introducing nonlinearity, functions such as ReLU enable the model to learn intricate mapping.
- **Regularization Techniques:** Model complexity can be controlled, and overfitting can be avoided by utilizing techniques such as batch normalization, L1/L2 regularization, and dropout.
- **Architectural Choices:** The use of convolutional layers in CNNs for image processing or recurrent layers in recurrent neural networks (RNNs) for sequential data are examples of architectural decisions that affect model complexity.
- **Input Layer:** There learnable parameters in the input layer. Its purpose is to receive and transfer the input data to the next tier in the required format.
- **Pooling Layer:** The utilization of pooling layers lowers the total number of parameters in the model as well as its computational complexity. They also aid in preventing overfitting by reducing the sample size of input data.
- **Convolution Layer:** Convolution layers process input data using learnable filters called kernels. The number of filters, filter size, and other criteria determine the number of parameters in the convolutional layer. The convolution layers shrink the size of the image without sacrificing a significant pixel-to-pixel correspondence.
- **Fully Connected Layer:** All parameters of the fully connected layer are linked to each other. This layer is crucial for determining intricate connections and effects between variables, especially when dealing with classification-type tasks. In the fully linked layer, the number of parameters is directly proportional to the total number of neurons.

Time Complexity:

In a CNN, the time complexity is the number of operations used to analyze the input data to produce output predictions. Thus, the time complexity depends on the number of layers, size of inputs, and complexity of operations in layers. More layers in a CNN and larger filter sizes in the network show that the time complexity increases with deeper CNNs owing to the increased operations in the convolution pooling and activation functions. This is where improvements in hardware, including GPUs and different optimization techniques, are of assistance. The degree of complexity has to be just right to prevent over- or under-fitting, and to allow the model to generalize to actual tasks.

With regard to the above, the time complexity of the CNN model can be determined using the following factors; height of the input image (I_height), width of the input image (I_width), filter height (F_height), filter width (F_width) and the number of input channel (ch_in), number of output channel (ch_out), step operation, and offset. parameters.

The time complexity of a convolutional operation in a neural network is given by eq. 1

$$\text{Time complexity} = (I\_width \times I\_height \times F\_width \times F\_height \times C\_in) \times C\_out) \quad 1$$

The time complexity of the CNN model depends on the number of parameters in the layers, such as the input, pooling, and fully connected layers.

The input image is fed into the convolution layer, giving rise to the general eq. for the complexity of the convolution layer, denoted as:

$$O\left(N^2 * S^2 * C_{in} * C_{out}\right) \quad 2$$

where N represents the image size (N × N), Si is the filter size, Cin is the input channel, and Cout is the output channel.

The complexity of a pooling layer in a neural network depends primarily on the size of the pooling window (kernel) and the stride used for pooling. The pooling operation samples input feature maps by reducing their spatial dimensions. where Nin = height or width of the input feature map, S = width or height of the pooling layer, and Cin= number of input channels. The complexity of a pooling layer can be expressed in terms of the number of operations required to perform the pooling operations. For each pooling window, a single operation was typically performed to compute the maximum value within the window. Therefore, the overall complexity of the pooling layer can be denoted by eq.3

$$O\left(\frac{N_{in}}{S} * \frac{N_{in}}{S} * C_{in}\right) \quad 3$$

The complexity of a fully connected layer in a neural network depends on the number of neurons in the layer and size of the input feature vector. Nin = dimensionality of the input feature vector and Nout = number of neurons in the fully connected layer. The complexity of a fully connected layer can be expressed as the number of operations required to compute the output of that layer. For each neuron in the fully connected layer, a dot product operation was performed between the input feature vector and the weights of the neuron, followed by the addition of the bias term and an activation function. Therefore, the overall complexity of a fully connected layer can be expressed as

$$O\left(N_{in} * N_{out}\right) \quad 4$$

The time complexity of the proposed model was calculated using eq. 2,3,4 and other parameters, such as filter size, no filters, stride, and padding. For the proposed model, the filter sizes were 11 × 11, 3 × 3, and 1 × 1. The numbers of filters is 32,64,128,256,512. The same padding and different stride values 4x4,2x2 and 1x1 have were taken[13]. Considering all these parameters of the proposed model, the total summary of eq 2, 3, and 4 is given by eq..5

$$O\left(N^2 * S^2 * C_{in} * C_{out}\right) + O\left(\frac{N_{in}}{S} * \frac{N_{in}}{S} * C_{in}\right) + O\left(N_{in} * N_{out}\right) \quad 5$$

To prove this practically, experiments were performed with different CNN models to examine 2 and 4 and the overall time required for the model according to eq 5.

The analysis focuses on the time complexity of CNN models, exploring how it varies with parameters such as the number of convolutions and fully connected layers, pooling size, filter size, number of filters, stride, and padding. A standardized image size of 227 × 227 pixels was utilized across the CNN models. Leaky ReLU activation functions are employed in all layers except the output layer, where Softmax activation is used. A batch size of 32, learning rate of 0.0001 with the Adam optimizer, and the same padding were employed[14]. Different CNN models were selected based on variations in these parameters. The model setup parameters, detailed in Table 2, are defined by eq. 2 and 4:

Table 2 CNN model setup Parameters

Model	**Number Of Convolution Layers**	**Number of Filters**	**Pooling Size**	**Filter Size**	**FC**
CNN-1	4	32,64	[2x2, 1x1]	[5x5, 5x5]	1
CNN-2	4	[32, 64, 128]	[2x2, 1x1]	[11x11]	1
CNN-3	5	[32, 64, 128, 256]	[2x2, 1x1]	[11x11, 3x3]	2
CNN-4	5	[32, 64, 128, 256, 512]	[2x2, 1x1]	[11x11, 3x3, 1x1]	3
AlexNet	5	[96,256]	[3x3]	[11x11,3x3]	3
VGG	16	[64,128,256,512]	[2x2]	[3x3]	3
GoogleNet	22	Varies	[3x3,5x5]	[5x5,3x3,1x1]	1
Nightshade-CNN	**5**	**[32, 64, 128, 256, 512]**	**[2x2, 1x1]**	**[11x11, 3x3, 1x1]**	**1**
Enhanced Nightshade-CNN					

The time complexity of the CNN models was evaluated using Google Colab with NVIDIA GPUs (TESLA T4 and A100), by analyzing various CNN parameters and execution epochs for nightshade crop leaf datasets, as shown in Table 3.

Table 3 CNN parameters and execution epochs for nightshade crop leaf datasets

Model	**Number Of Convolution Layers**	**FC**	**Filter Size**	**Epochs (Times in Hrs.)**
CNN-1	4	1	[5x5, 5x5]	2.5
CNN-2	4	1	[11x11]	4.5
CNN-3	5	2	[11x11, 3x3]	5.5
CNN-4	5	3	[11x11, 3x3, 1x1]	9
AlexNet	5	3	[11x11,3x3]	8
VGG	16	3	[3x3]	26
GoogleNet	22	1	[5x5,3x3,1x1]	48
Nightshade-CNN[9-10]	**5**	**1**	**[11x11, 3x3, 1x1]**	**4**
Enhanced Nightshade-CNN[11]	**5**	**1**	**[11x11, 3x3, 1x1]**	**6**

Table 3 compares various neural network models based on the number of convolution layers, fully connected (FC) layers, filter sizes, and training time (epochs in hours). The following is a brief explanation.

- **CNN-1**:4 convolution layers, 1 FC layer, filter sizes of 5x5, with training times increasing from 0.5 to 2.5 hours over 100 to 500 epochs.
- **CNN-2**: Similar to CNN-1, but with an 11 × 11 filter, and its training time increases from 0.5 to 4.5 hours.
- **CNN-3**:5 convolution layers, 2 FC layers, using filters of 11 × 11 and 3 × 3, with longer training times (up to 5.5 hours).
- **CNN-4**:5 convolution layers, 3 FC layers, with filters 11 × 11, 3 × 3, and 1 × 1, the training time increased significantly to 9 h at 500 epochs.
- **AlexNet**: five convolution layers, three FC layers, and filters similar to CNN-4, requiring up to 8 h.
- **VGG**: 16 convolution layers, three FC layers, with 3 × 3 filters, requiring much more training time (up to 26 h).
- **GoogleNet**: 22 convolution layers, one FC layer, using various filters, and required the longest training time (up to 48 h).

- **Nightshade-CNN**: five convolution layers, one FC layer, with filters similar to CNN-4, and training for up to 4 h.
- **Enhanced Nightshade-CNN**: Same architecture as Nightshade-CNN, but with a slightly longer training time of up to 6 h.

Space Complexity

Parameters, intermediate activations, and other data models during training and inference. In CNNs, location complexity depends on conditions such as the number of parameters in the model, its size, and the size of the average maps created by each layer[15-17]. or the densely connected layers are more complex. Additionally, large batch sizes can increase the memory requirements during training. Managing the complexity of a site is especially important in areas where resources are limited, such as edge devices or mobile applications. This can help protect deep learning models while making them efficient. In the context of CNNs, the required resources, including time and memory (space) complexity, are issues of concern. To solve these problems, the Nightshade-CNN [9,10] and Enhance-Nightshade-CNN[11] models use special filters (11 × 11, 3 × 3, and 1 × 1) with a total of six layers. This method helps reduce the number of inconsistencies and, hence, the complexity of the site.

The space complexity of the model depends on the weight, which is determined by the number of input channels (c), weight (w), height (h), and the number of output channels (k), with 'k' representing the bias.

Each filter in an instance of a convolutional layer has a set of parameters, which are trainable parameters that arrive at by taking the product of the number of input channels (c), width (w), and height (h) of the filter and the number of filters (k), thus cwhk. These parameters are learned during training to obtain relevant features from the input data. Bias terms are extra variables incorporated into a neural network layer to enhance the model's ability to fit the training data. Every filter has a bias term that provides k bias parameters in total.

By adding the weights and bias terms, the total number of parameters |W| in the convolutional layer was calculated using eq. 6

$$|W| = cwhk + k$$
$$=(cwh+1)k \quad 6$$

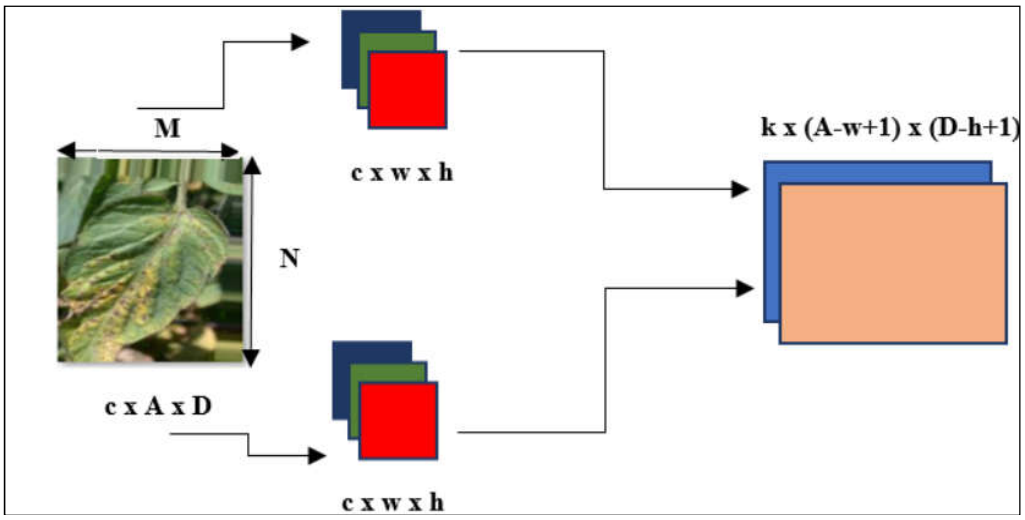

Figure 3 Space Complexity of Nightshade Crop Leaf

Figure 3 shows that the nightshade crop leaf image size was c × A × D 7.

Where A= width
D = height
c = channel or feature map.

The weights were downloaded and stored in memory, and the required space was considered. The model has an input image with dimensions AxD and convolves it with a kernel of size c × w × h.

The number of parameters associated with each neuron in the fully connected (FC) layer is determined by the spatial dimensions (width and height) of the input image or feature map and the dimensions of the hidden layer. The effective width of the receptive field of the hidden layer was (A-w+1), and the effective height was (D-h+1). By multiplying these two terms, we obtained the total number of parameters per neuron. Each neuron in the FC layer has a weight parameter for each connection from the input features along with its associated bias parameter. Therefore, the total number of parameters for the weights and biases was 2k.

Multiplying the number of parameters per neuron by the total number of neurons (k) yields the total number of parameters in the fully connected layer:.

Hence

$$\text{resultant kernel} = 2k(A-w+1)(D-h+1) \quad 8$$

where A image width,
D=image height,
w=hidden layer width,
h= hidden layer height, and
k=bias in no of kernels.

Therefore, the total required space for the model is equal to the addition of eq. 6,7 and 8.

Hence,

Total Space requirement = cAD + k(cwh+1) + 2k(A-w+1) (D-h+1) 9

The spatial complexity of a convolutional neural network (CNN) model depends on various factors, including the size of the input image, the total number of weights, and the dimensions of the hidden layer feature maps. Therefore, space complexity, denoted as O(cAD + k(cwh+1) + 2k(A-w+1) (D-h+1), provides an estimate of the amount of memory required for these models.

The spatial complexity analysis of CNN models includes various parameters, such as convolution and fully connected layers, batch size, filter size, and regularization techniques. Based on these variations, models trained on 227 × 227 images with a dropout of 0.5 and a batch size of 32 were chosen. Memory requirements were calculated using eq. 9, with detailed results shown in Table 3, whereas the implementation used NVIDIA Tesla T4 and A100 Tensor GPUs.

Table 4 highlights the correlation between model size and storage space.

Model	**Number Of Convolution Layers**	**FC**	**GPU Type**	**Memory Requirement**	**Processor**
AlexNet	5	3	Tesla T4	16GB	NVIDIA Tesla T4
VGG	16	3	A100 Tensor	40 GB	NVIDIA A100 Tensor Core
GoogleNet	22	1	A100 Tensor	40 GB	NVIDIA A100 Tensor Core
Nightshade-CNN[8-10]	**5**	**1**	**Tesla T4**	**16GB**	**NVIDIA Tesla T4**
Enhanced Nightshade-CNN[11]	**5**	**1**	**Tesla T4**		

Table 4 highlights the correlation between the model size and storage space, with larger models such as VGG and GoogleNet requiring more storage space because of their extensive layering. By contrast, Nightshade-CNN and Enhanced Nightshade-CNN use Google infrastructure with Tesla T4 GPUs in an NVIDIA environment. The storage needs are affected by the size of the model and the dimensions of the input image, including the height and width.

From Table 3, the following points have been concluded:

- The classic AlexNet model is known for its pioneering role in deep learning with moderate computational requirements.
- The deeper VGG model requires more computational resources, particularly for handling larger datasets.
- GoogleNet focuses on reducing the number of parameters through its inception modules, making it more efficient, despite having more layers.
- NightShade-CNN appears to be a more lightweight model with fewer layers and lower memory requirements.
- The Enhanced Nightshade-CNN requires an additional preprocessing step; however, the final storage requirement remains the same as that of the Nightshade-CNN.

5. CONCLUSION

The time complexity of a model is greatly influenced by factors such as the number of layers, filters, and filter size, whereas the space complexity is largely determined by the size of the model. Models characterized by fewer layers and smaller batch sizes generally exhibit lower space complexity. Thus, meticulous design considerations are vital for enhancing both the time complexity O $(N^2 * S^2 * C_{in} * C_{out})$ + O $(\frac{N_{in}}{S} * \frac{N_{in}}{S} * C_{in})$ + O (Nin * Nout) and the space complexity O(cAD + k(cwh+1) + 2k(A-w+1) (D-h+1)) in the Nightshade-CNN and Enhanced Nightshade-CNN models.

Funding Statement No funding received.

Data availability Plant Village repository was used for the experiment. The actual dataset is available at https://www.kaggle.com/datasets/emmarex/plantdisease.

Code availability Proposed CNN models has been implemented in python with the OpenCV environment and executed on Google colab environment.

DECLARATIONS

Conflicts of interest

The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this study.

Ethics approval Not Applicable.

Consent to participate Not Applicable.

Author Contribution Statement:

Barkha Joshi: Conceived the research idea, designed the methodology, conducted data analysis, and performed the experiments. Wrote the original draft of the manuscript. Also served as the corresponding author, handling communication with the journal, submitting the manuscript, and responding to reviewer comments. First author.

Dr. Hetal Bhavsar: Provided guidance throughout the study, reviewed and critically revised the manuscript, and suggested appropriate journals for submission.

All the authors have accepted responsibility for the entire content of this submitted manuscript and approved submission.

REFERENCES

1. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems.
2. Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.
3. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going Deeper with Convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.
4. Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both Weights and Connections for Efficient Neural Networks. Advances in Neural Information Processing Systems.
5. Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using Deep Learning for Image-Based Plant Disease Detection. Frontiers in Plant Science.
6. Pan, S. J., & Yang, Q. (2009). A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering.
7. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.
8. Joshi, B. M., & Bhavsar, H. (2020). Plant leaf disease detection and control: A survey. *Journal of Information and Optimization Sciences*, *41*(2), 475–487. https://doi.org/10.1080/02522667.2020.1734295.
9. Joshi, B.M., Bhavsar, H. (2023). Lycopersicon Crop Leaf Disease Identification Using Deep Learning. In: Pandit, M., Gaur, M.K., Kumar, S. (eds) Artificial Intelligence and Sustainable Computing. ICSISCET 2022. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-99-1431-9_2.
10. M. Joshi, B. ., & Bhavsar, H. . (2023). Deep Learning Technology based Night-CNN for Nightshade Crop Leaf Disease Detection. International Journal of Intelligent Systems and Applications in Engineering, 11(1), 215–227. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2461.
11. Joshi, B.M., Bhavsar, H. A nightshade crop leaf disease detection using enhance-nightshade-CNN for ground truth data. Vis Comput (2023). https://doi.org/10.1007/s00371-023-03127-y.
12. https://www.kaggle.com/datasets/emmarex/plantdisease.
13. Xiang, Y., Zhou, H., Li, C. *et al.* Deep learning in motion deblurring: current status, benchmarks and future prospects. *Vis Comput* (2024). https://doi.org/10.1007/s00371-024-03632-8.
14. Xu, C., Han, K. & Xu, W. Image-aware layout generation with user constraints for poster design. *Vis Comput* (2024). https://doi.org/10.1007/s00371-024-03657-z.
15. Salar, A., Ahmadi, A. Enhancing high-vocabulary image annotation with a novel attention-based pooling. *Vis Comput* (2024). https://doi.org/10.1007/s00371-024-03618-6.
16. Islam, M., Dinh, A., Wahid, K., and Bhowmik, P.: Detection of potato diseases using image segmentation and multiclass support vector machine. In: 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE) (pp. 1–4). IEEE (2)
17. Guadagna, P., Fernandes, M., Chen, F., et al.: Using deep learning for pruning region detection and plant organ segmentation in dormant spur-pruned grapevines. Precision Agric. 24, 1547–1569 (2023). https://doi.org/10.1007/s11119-023-10006-y.
18. Cheng Y., F. Yu, R. Feris, S. Kumar, A. Choudhary, S. Chang (2015) “An Exploration of Parameter Redundancy in Deep Networks with circulant projections” https://doi.org/10.48550/arXiv.1502.03436.
19. Giusti, A.; Dan, C.C.; Masci, J.; Gambardella, L.M.; Schmidhuber, J. Fast image scanning with deep max-pooling convolutional neural networks. In Proceedings of the 20th IEEE International Conference on Image Processing, Melbourne, Australia, 15–18 September 2013; pp. 4034–4038.
20. T. Li, M. Xu and X. Deng, "A deep convolutional neural network approach for complexity reduction on intra-mode HEVC," 2017 IEEE International Conference on Multimedia and Expo (ICME), Hong Kong, China, 2017, pp. 1255-1260, doi: 10.1109/ICME.2017.8019316.
21. Saad Naeem, Noreen Jamil, Habib Ullah Khan, Shah Nazir “Complexity of Deep Convolutional Neural Networks in Mobile Computing”. https://doi.org/10.1155/2020/3853780.
22. T. Makkar, Y. Kumar, A. K. Dubey, Á. Rocha and A. Goyal, "Analogizing time complexity of KNN and CNN in recognizing handwritten digits," *2017 Fourth International Conference on Image Information Processing (ICIIP)*, Shimla, India, 2017, pp. 1-6, doi: 10.1109/ICIIP.2017.8313707.

23. Abdellatef H, Karam LJ. Reduced-complexity Convolutional Neural Network in the compressed domain. Neural Netw. 2024 Jan;169:555-571. doi: 10.1016/j.neunet.2023.10.020. Epub 2023 Oct 24. PMID: 37952391.
24. Zhao, Yulin, Donghui Wang, Leiou Wang, and Peng Liu. 2018. "A Faster Algorithm for Reducing the Computational Complexity of Convolutional Neural Networks" *Algorithms* 11, no. 10: 159. https://doi.org/10.3390/a11100159.