Optimizing CNN Architectures for In-field Grape Disease Diagnosis on Mobile Embedded Systems

Shital Karande [1,2,*] and Bindu Garg [1,*]

[1] Department of Computer Engineering, Bharati Vidyapeeth's College of Engineering, Pune, India
[2] Department of Computer Engineering, Bharati Vidyapeeth's College of Engineering for Women, Pune, India
Email: shital.jadhav@bharatividyapeeth.edu (S.K.); brgarg@bvucoep.edu.in (B.G.)
*Corresponding author

***Abstract*—Agriculture evolution involves practices to monitor leaf visual pattern to accurately diagnose plant diseases with computer vision solution. The diagnostics carried out in the field has complexity that is removed with proposed segmentation algorithm. Proposed approach is fusion of multi color threshold and Grab Cut suitable for light weight mobile devices. Mathematical principle in Convolutional Neural Network (CNN) has several variations of architectures. Because they have building blocks that process input image data in serial, parallel or sliced manner. This paper evaluates grape disease data-set with latest CNN models ranging from simple sequential to efficient parallel models. Different architecture design patterns explored in this research work to give birds eye view in deep learning utilization for image classification task. Eight different CNN architectures are based on different principles like parallel layers, skip connection, depth wise separable convolution, densely connected as well as sparsely connected layers evaluated in this research work. Training and testing carried out with commodity hardware and tested on portable devices. State of art architecture like Visual Geometry Group (VGG) 16 achieves accuracy of 0.98 for testing whereas Inception with parallel multi-layer filter model achieves comparatively less accuracy of 0.90. ResNet is the one of promising pre-trained model giving accuracy of 0.9916 by learning from previous layers with skip connection. DenseNet architecture is an improvement over ResNet architecture with reduced model size using novel connectivity patterns. Latest Vision Transformer (ViT) gives best result with accuracy of 0.9942. This paper proposed eleven-layer architecture for efficient classification of grape crop diseases. Variation of architecture with input images size 128 and 160 are evaluated proving size 160 more suitable in comparison with other deep architectures. Proposed model is best in accuracy, F1-Score and recall of 1.0 for majority classes. It is light weight, has 3.6 MB size and is suitable for embedded systems.**

***Keywords*—Convolutional Neural Network (CNN), Grab Cut, plant disease, embedded systems, parallel models, DenseNet**

I. Introduction

India is the second largest producer of grapes in the world with 162,000 hectares planted in 2021. Summer's hot and dry climate is most favorable to grape plants but uncertain rain or clouds turns harmful to grapes while ripening. Warm and moist weather develops diseases like Black Rot and Bacterial Leaf Spot. Early diagnosis of diseases with visual examination of leaves enables immediate action. Artificial Intelligence (AI) devices can be used in the farm field to provide initial stage diagnosis and timely action. Computer vision is part of AI that uses trained Convolutional Neural Networks (CNN) as core building blocks for disease classification. CNN is a deep learning technique designed to automatically learn features having spatial hierarchies from lines to complex patterns. The latest trend in CNN architecture building tends towards increased model in depth and size if enough labeled datasets are available for training. Computational cost and computational efficiency are highly sensitive for utilization in real world applications [1]. Mobile devices require low parameter count to work efficiently in contrast with big-data scenarios which are having complexity of data and requiring additional processing over data. Utilizing tools and techniques as per requirement of scenario and scaling up as per applicability of autonomous solutions in various image classification problems need to be used thoughtfully [2]. Image classification is a subpart of computer vision solutions to recognize object or categorize image for classic problems of disease diagnostics. Image is well recognized with image segmentation. Multiple categories of algorithm work on retrieving boundary and regional information. Complex images well segmented for further classification in Image recognition task. Till today there is no identified technique that proves effective for all kinds of classification tasks. Requirements of number of layers, type of optimization, input size vary as per problem specification [3].

Image segmentation carried out with algorithms i.e., threshold, multi color threshold, edge detector like Robert, Canny etc. K-means, fuzzy clustering used in segmentation along with other techniques such as graph-based algorithms like Min Cut, Grab Cut. These algorithms have a specific purpose to identify texture, color features of the images [4].

Grab Cut is an image segmentation algorithm preserving boundary and texture information of image

Manuscript received June 26, 2025; revised July 25, 2025; accepted September 28, 2025; published February 27, 2026.

obtaining good segmentation for complex images. Infected plants leaves are captured with camera, which require highlighting focus on the leaf under observation. Traditional techniques require user interaction to mark foreground and background pixels. Grab Cut algorithm with two variants which represent objects of interest either with a bounding box or mask. Segmentation is an iterative process where count of iteration varies in different applications as per image complexity [5, 6].

Plant disease dataset available for multiple crops. Farmers cultivate particular crops depending on geographic condition and environment. Proposed work is useful for grape cultivators which suggests solutions for grape disease diagnosis using mobile device. Mobile embedded systems restricted with limited device storage. Farms majority of the time do not have internet connectivity. Mobile as a crop diagnostics tool should operate without the internet giving capability to application. Research work carried out in three stages: First is preparation of data-set with automatic segmentation, second is evaluation of various models for classification and final stage is developing mobile embedded systems.

This paper presents improved automatic Grab Cut algorithm, combination of texture, color, shape features and graph-based image segmentation. Paper evaluates performance of various CNN for grape disease classification.

The research questions that will be answered in this paper are:

1) How to perform efficiently in-field image segmentation?
2) What are different important architecture practices in classification using CNN model?
3) How a parallel CNN model would be effective in crop disease classification?
4) How to design and improve lightweight CNN for crop disease identification and classification?
5) How to compare the performance of CNN using various factors?

Research work presents an application grape care with suitable segmentation and classification to resolve in field diagnostic tasks. Section II of the paper discusses research work in plant disease identification followed by mathematical modelling of proposed system in Section III. Architecture models selected for training explained in Section IV and Section V details about dataset as well as experimental setup. Section VI gives system architecture details and following Section VII provides experimental results with discussion. Section VIII presents' conclusions and the last section is about references.

II. Literature Review

Plants are susceptible to diseases with changing environmental conditions. Healthy plants have uniform green leaves whereas disease infection changes visual appearance like discoloration, spots or ring-like textures. Crop image segmentation is carried out to cluster together similar features for identification of disease types. Multiple features such as contrast, homogeneity, entropy, energy etc. calculated with techniques like K-mean, fuzzy clustering and genetic algorithms [7]. Techniques like Grey Level Co-occurrence Matrix (GLCM) and Scale Invariant Feature Transform (SIFT) are used to analyze features and the shape of leaves. Histogram analysis is used to identify threshold values required for image segmentation. Most of the research work uses gray level images. Masks are created from input images of plant leaves where every pixel analyzed. If pixel exceeds a certain threshold, it is labeled as disease spot. Petrellis [8] used pixels label 0 for background color, 1 for healthy leaf and 2 for disease marks this is called Background Normal Spot (BNS). Color images have multiple channels. Several types of color models used for representing images. It is difficult to determine optimal technique for multichannel color image segmentation. Illumination conditions while capturing images in-field affects the result. In the farm crop images have additional background information like soil, dried leaves and hand or any other body part of human while capturing. Accuracy reduces drastically due to this unwanted information. Present work has a limited number of publications addressing background removal from complex images. K-means clustering, genetic algorithms and fuzzy means clustering require prior information of possible number of clusters in image. It is difficult to know cluster numbers in prior about real field images [9–11].

Grab Cut is a semi-automatic, iterative and context preserving, image segmentation algorithm used for human face detection, video segmentation etc. This algorithm requires user interaction to identify the bounding box around the object. The Gaussian Mixture Model (GMM) principle is used for separating foreground from background information with energy minimization. Mobile or light weight devices are restricted in memory and processing powers, requiring rapid segmentation. Research papers have used sliding window protocol to identify this boundary [12]. Mostly with a large number of classes increases complexity and two classes can have similar features creating conflict. Hand crafting feature requires in-depth understanding of the images and success depends on expertise. Extracted features traditionally classified with classifiers like Decision Tree, Random Forest, Support Vector Machine, Clustering Algorithms and Artificial Neural Network [13, 14].

Plant-Village is a cropping image dataset. It is huge and publicly available, useful for deep learning models. Farm images are complex whereas Plant-Village images acquired in laboratory environment have no such complexity. Class imbalance get created with images from different sources. Pre-processing techniques like contrast enhancement technique, background removal and data augmentation methods are used to remove class imbalances [15]. CNN is a successful algorithm for classification task. This algorithm fully automates the process of feature extraction. LeNet is the first successful CNN model experimented by the research community for applications like character recognition, object classification, leaf diseases classification etc. [16]. CNN architectures are designed with different combinations of layers sequences, slicing as well as parallel processing.

Research carried out with three stage CNN architecture for real field images of maize crop. CNN is used in stages for generating heat feature maps. Higher stage CNN acts as a classifier combining different features at the previous stage [17].

Image-net is an image classification challenge that discovers multiple solutions. AlexNet is one of the initial deep learning models having eight layers obtained with high accuracy in plant disease identification tasks [18]. VGGNet is a sixteen layer or nineteen layers architecture with deep layers. Research work compares AlexNet initial deep neural network with improved architecture Visual Geometry Group (VGG) 16 for tomato crop's seven disease classes [19]. Results observe accuracy of 95.81% with AlexNet whereas VGG16 architecture achieves 96.19%. CaffeNet deep convolutional neural network architecture is used for grape disease identification with accuracy of 96.3%. Prominent use of the Inception module is in GoogleNet (Inception V1) architecture with 27 layers reducing the problem of overfitting. Factorized kernel improved speed in Inception as well as 1×1 convolution introduced in middle layers of Inception [20–22]. Plant-Village dataset trained with GoogleNet achieved highest accuracy of 0.9916% [23]. Regularization technique in which densely connected layer's neurons are switched off to avoid overfitting. Zhang *et al*. [24] carried out with varying dropout for nine types of maize disease categories achieves accuracy of 98.9% with GoogleNet. Image classification accuracy improves very deep neural networks with skip connection in 2015 is ResidualNet architecture. Variants of ResidualNet have different number of layers i.e. 18, 34, 50, 101 and 152 layers. Selvaraj *et al*. [25] for banana diseases and pest detection uses ResNet as backbone network for region-based CNN achieving highest accuracy of 99.9%. Region proposal networks perform selective search identifying objects of interest from complex images. Training network requires intensive labeling images for possible disease classes which is time consuming [26]. Research work proposed two stage networks, in first stage AlexNet used for feature extraction and second stage Yolo for classification. Yolo uses 32-layer classifier with average pooling. Proposed approach expanded Plant-Village dataset with augmentation using Generative Adversarial Network (GAN) achieving accuracy of 93.67% [27].

CNN classifier speeded up with depth-wise one-dimensional convolution which is introduced in MobileNet with 28 layers designed specifically for mobile devices [28]. Apple, corn, grape, potato, tobacco and tomato crops 29 disease classes identified using Inception, DenseNet, MobileNet model with accuracy 93.8%, 90.5%, 91.0% respectively [29, 30]. Input size 224×224 is used in majority successful architecture, as well as experimentation with different input sizes carried out. Some research work explored other input sizes such as 60×60, 256×256, 98×98 and 299×299 [31, 32].

Geetharamani and Pandian [33] proposed nine-layer lightweight model based on a deep CNN trained 39 different classes. Six types of data augmentation methods like image rotation, flipping, gamma correction, color augmentation, and scaling were done. Similar model is used for nine disease classes of tomato, gives 91.2% accuracy [34]. Apart from layer organization other factors determining accuracy and computational efficiency are Activation function. Logistic (Sigmoid) is used in earlier feed forward neural network. Rectified Linear Unit (ReLU) is more suitable for Image classification. SoftMax is probabilistic function, used in higher layer of CNN summing up to value one giving classification probability [35, 36]. Latest research work used sequential three stage model with 19 layers. The first stage consists of two convolutional layers with 32 filters, one max-pooling layer and one batch normalization layer. The second stage has two convolutional layers with 64 filters, one max-pooling layer, one batch normalization layer, and 0.3 dropout. The third stage consists of two convolutional layers with 128 filters, one max-pooling layer, one batch-normalization layer, and one 0.3 dropout. The model consists of one flatten layer and two dense layers. Proposed model achieves an accuracy of 89% [37]. Models trained with pretrained architectures later deployed on Raspberry Pi kit [38]. Low computing devices are having restricted amount of storage and processing resources [39]. Real time monitoring of crop disease enables timely action and prevent spread of disease. Appropriate tools which are suitable to capture crop images and diagnose disease accurately, are widely needed [40]. Research work on real field images with effective segmentation and classification are latest trend in plant monitoring. Present literature focuses more on existing pre-trained models for classification rather than developing new architecture for crop disease identification.

III. Mathematical Modelling of Proposed System

Existing published work carried out on images with uniform background and lightening condition from plant village dataset. Real field crop images are used to expand data-set for new disease classes or images. Proposed work provides a new method to segment multiple real field images with an automatic approach. Green color component is used to identify potential leaf part foreground to separate from background.

Proposed technique has two stages to fully automate the Grab Cut algorithm for concisely segmenting the leaf image. Proposed algorithm extracts real field grape images for disease detection. This technique is fast and performs well on bulk image data-set. Fig. 1 presents flowchart of proposed system. In phase one of technique RGB color images transformed to HSV color format in order to work effectively for non-uniform lighting conditions. Mask is created from green pixels intensity which extract leaf from image eliminating background. Multiple clusters identified from foreground leaf image later sorted to find highlighted potential leaf under observation. Stage two applies Grab Cut algorithm with mask and later check extracted objects are leaf or non-leaf.

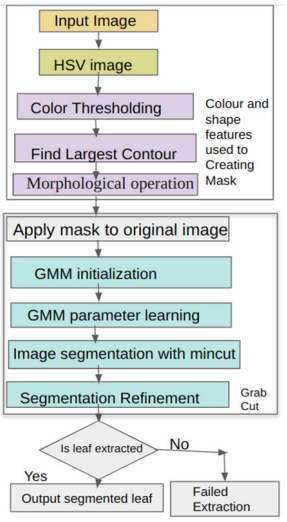


Fig. 1. Proposed segmentation flowchart.

Image represents the distribution of color intensities across two-dimension planes (x, y). An image function $z = f(x, y)$ represents z color intensities at coordinate (x, y), where x, y are dependent variables and z is an independent variable. Individual pixel p in image carries e edge information collectively forming object boundary.

Image F contains two parts: Foreground object and background, where multiple foreground objects are possible in one image. Foreground object R_o is a region which is a set of pixels with similar intensity values and a set of background pixels is represented with R_b. Boundary information is the collective energy of edges of pixels with same color intensities. Pixels lying on the object boundary region represented with R_u changes intensities across the border of the object. Color image F has individual pixel intensity, which is a combination of three components "Red", "Green" and "Blue" i.e., $F(R,G,B)$.

A. Convert Image from RGB to HSV Color Model

1) Stage one of segmentation

Step 1: Convert image from RGB to HSV color model. Fig. 2 shows image representation in form of histogram. Global color thresholding is applied for original image to separate object of interest from background.

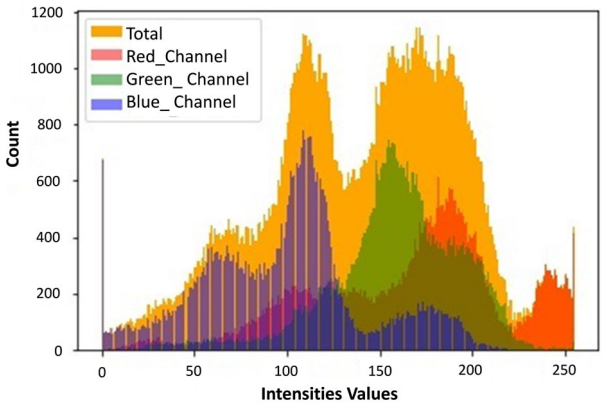


Fig. 2. Image color histogram.

$F(R,G,B)$ is the red, green and blue color model transformed to $I(H,S,V)$ is Hue, Saturation and Value (HSV) color model. HSV model derived from RGB model to describe color, used specifically for high quality computer graphics. HSV conical color model describing primary color where base is white and vertex is black. R, G, B color are normalized to calculate Hue, Saturation, Value. Hue is angle with vertical axis, saturation ratio of purity of hue and value is brightness of color.

$$F(R,G,B) = I(H,S,V) \tag{1}$$

where value V is calculated as,

$$V = \max(R,G,B), \quad 0 \le V \le 1 \tag{2}$$

If $V \neq 0$:

$$S = \left(V - \min(R,G,B)\right)/V \tag{3}$$

Else $S = 0$.
If $V = R$,

$$H = 60(G-B)/(V - \min(R,G,B)) \tag{4}$$

If $V = G$,

$$H = 120 + 60(B-R)/(V - \min(R,G,B)) \tag{5}$$

If $V = B$,

$$H = 240 + 60(B-R)/(V - \min(R,G,B)) \tag{6}$$

Step 2: Global color thresholding is applied to the entire original image to separate objects of interest from background. Image captured in visible light highlighting leaf under observation. It has two parts, foreground object of interest and background information.

Histogram represents distributions of intensities in image where objects and background information are represented with lobes and pixels that sufficiently distinct. Global threshold is used over the entire image to separate probable leaf green regions from other information.

$$S = T\left(I(H,S,V)\right) \tag{7}$$

where S is a segmented image after applying global threshold, an iterative algorithm over the I image with a sequence of steps as follows:

(a) Select minimum and maximum green intensity color range as global threshold T.
(b) Segment the image using value T assigned in step (a). This will produce two groups of pixels: O_1 representing a set of pixels with intensity greater than T, and B_1 representing a set of pixels with intensity lesser than T.
(c) Calculate the mean (average) intensity values m_1 and m_2 for pixels in O_1 and B_1.
(d) Calculate a new threshold value $T = (m_1 + m_2)$.
(e) Jump back to steps (b) through (d) until get difference between values of T with repetitive

iterations. Value should be less than a predefined parameter ΔT.

Algorithm performs well for in-field images with highlighted leaves. Histogram representation shows a clear valley between the parts of the objects and background pixels.

Step 3: S segmented image contains R number of contours, since leaf under observation is highlighted, it would be the largest contour for image under observation. Masks are created to identify leaf regions in the image with following steps:

(1) Read the segmented Image and convert it to Grayscale format using binary threshold $S \Rightarrow G$. Apply threshold to generate a binary image having only two values 0 and 1 from a given segmented image S by separating it into two parts object and background with a threshold value. Hence pixels having intensity values greater than the T threshold will be treated as white or 1 in the output image and the other pixels will be black or 0.

(2) Find all contours with chain approximation method: A curve connecting all the points across the border having identical color and intensity levels is called a contour. Contours are useful in shape analysis and leaf detection in complex real field images.

$$G = U_{i=1}^{n} \cdot R_i \tag{8}$$

(3) Sort and identify largest contour: Sort $G[R_i]$ to get R_k, where $R_1 > R_2 > R_3 > \ldots > R_n$.

(4) Save largest contours to create mask for leaf: Mask M is created with the most probable leaf region.

$$M = G_{Largest}(R_k) \tag{9}$$

Step 4: Morphological operation of dilation is used to create a clear mask.

The closing dilation applied for binary image M by a structuring element B, represented by:

$$M \times B = (M \oplus B) \tag{10}$$

where $\oplus$ denote the dilation.

2) Stage two of segmentation

Step 5: Image segmentation is performed with Grab Cut graph-based segmentation using a mask. Original image F and mask M used to identify precise leaf using Grab Cut algorithm. Image F there is a set of pixels representing foreground and background context called sources and sink respectively. Three sets of pixels are defined by $T = \{T_B, T_U, T_F\}$ help for segmentation. Background pixels are represented with T_B, foreground pixels are represented with T_F and some unspecified pixels are represented with T_U. Edge is the local property of pixel and its immediate neighboring pixel. It is a vector with magnitude and direction that represents variation in the image intensity in a 4 or 8 neighborhood of a pixel. Grab Cut graph uses edge weight for any two n_i, n_j pixels from the entire input image $z\{z_1, z_2, ..., z_j\}$. Segmentation based on opacity α at each pixel i.e. (α_1 ,..., α_n) and $0 \leq \alpha_n \leq 1$. Grab Cut label background pixels as 0 and foreground pixel as 1, $\alpha_n \in \{0,1\}$. Grab Cut algorithm allocates two arrays that separate foreground and background using Gaussian Mixture Model (GMM) used to cluster pixels K—the number of clusters. Each cluster has:

$$\theta\{\pi(\alpha,k), \mu(\alpha,k), \sigma(\alpha,k), \alpha = 0,1, k = 1,...K,\} \tag{11}$$

where: μ is mean RGB value, π is weighting coefficient, Σ is covariance matrix (3×3).

Step 6:

(1) Initially mask created in the previous step used around the foreground object. Outside this mask background pixels set represented with T_B. Unspecified pixels are set to be $T_U = T_B$ and $T_F = \theta$.

(2) Set $\alpha_n = 0$, when $n \varepsilon T_B$ and $\alpha_n = 1$ when $n \varepsilon T_U$.

(3) GMMs background and foreground are initialized from sets $\alpha_n = 0$ and $\alpha_n = 1$ respectively.

(4) Iterative minimization.

Opacity calculated with Gibb's energy function:

$$E(\alpha, k, \theta, z) = W(\alpha, k, \theta, z) + B(\alpha, z) \tag{12}$$

GMM determines cluster count k. W regional energy calculated with GMM, as:

$$W(\alpha, k, \theta, z) = \sigma D(\alpha_n, k_n, \theta, z_n), n \tag{13}$$

where:

$$D(\alpha_n, k_n, \theta, z_n) = \log p(z_n | \alpha_n, k_n, \theta) \log \pi(\alpha_n, k_n) \tag{14}$$

Gaussian probability distribution is represented with $p(\cdot)$, and mixture weighting coefficients are represented with $\pi(\cdot)$. B is boundary energy which is smoothness term constant for monochrome except that the contrast term calculated with Euclidean distance in color space:

$$B(\alpha, z) = \sum_{(m,n) \in C} [\alpha_n \neq \alpha_m] \beta \left\| e^{(z_m - z_n)^2} \right. \tag{15}$$

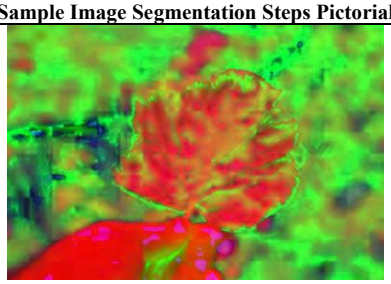
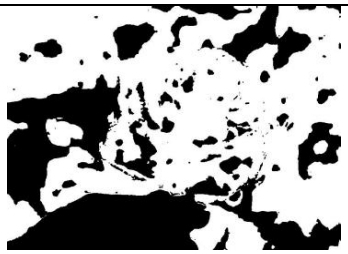
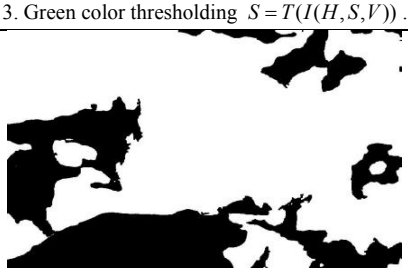
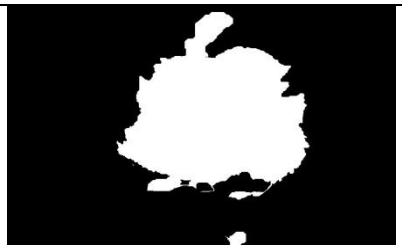
Step 7: After several iterations, a definite foreground mask was created and applied to the input image to extract a precise leaf.

Step 8: Save extracted leaf to prepare image database with total m number of images.

$$L = L_1, L_2, L_3, L_4, ..., L_m \tag{16}$$

Table I presents proposed segmentation results stepwise in pictorial format for further clarity.

TABLE I. PROPOSED SEGMENTATION TECHNIQUE

Sample Image Segmentation Steps Pictorial	Sample Image Segmentation Steps Pictorial	Equations of Segmentation Steps
 1. Input image $F(R,G,B)$ original image.	 2. HSV representation transform to HSV format $I(H,S,V)$.	$F(R,G,B) \Rightarrow I(H,S,V)$
 3. Green color thresholding $S = T(I(H,S,V))$.	 4. Mask after thresholding.	$S = T(I(H,S,V))$
 5. Mask after morphological operation dilation $M \oplus B$.	 6. Original images apply mask *M*.	Mask *M* is created with the most probable leaf region. $M = G(R_k)$, $M \times B = (M \oplus B)$ where $\oplus$ denote the dilation.
 7. Grab-cut probable foreground *M* is created with the most probable leaf region.	 8.Output image.	$E(\alpha,k,\theta,z) = W(\alpha,k,\theta,z) + B(\alpha,z)$

B. *Mathematical Model of Proposed Classification Technique with Convolutional Neural Network (CNN)*

The CNN is a specialized neural network used specifically for image classification. Image is a two-dimensional matrix representing pixel intensities. Pixels 4 or 8 ways connected to neighboring pixels. This leaning architecture takes advantage of image property which has similar intensities in pixels belonging to the same region. Each layer is sparsely connected to the next layer. Many variations of CNN architecture proposed by researcher but basic component remains same, explained in following part of section:

1) *Convolution layer*

The training database is denoted:

$$L = \left[X(1), y(1)\right], \left[X(2), y(2)\right], ..., \left[X(m), y(m)\right] \quad (17)$$

where $X(i)$ image and $y(i)$ are labels associated with it.

The convolutional layer applies feature extracting kernel function to each input layer and passes learned feature maps to the next layer. The new feature map is obtained with convolution operation using a learned kernel function. Neuron apply an activation function to reduce weight of convoluted results element-wise non-linearly.

To learn different features in detail, several kernels can be applied at different layers depending on architecture design. Mathematical representation of the feature value is z at coordinate position (i, j) in the k^{th} feature map of l^{th} layer in Eq. (18) as follows:

$$z_{i,j,k}^{l} = w_k^{l^T} + b_k^l \quad (18)$$

where x input at $(i, j)^{th}$ location, k^{th} feature of l^{th} layer, b bias and z output feature.

2) Rectified Linear Unit (ReLU) activation function

The activation function in neurons is used after the convolution operation to normalize weight function and speed up network data processing. ReLU is mostly used for images since intensities have non-negative values. The ReLU activation function formulated has zero or positive value.

$$a = \max\left(a(z_{ijk}^{l}), 0\right) \tag{19}$$

3) Max-pooling layer

The pooling layer is applied after every convolution layer reduces the resolution of the feature maps. The categories of pooling operations are minimum, average and maximum pooling. Combination of convolutional and pooling layers extract features at different levels of CNN representations.

$$y_{j,k,l}^{i} = \max\ pool\left(a_{m,n,k}^{l}\right) \forall (m,n) \in R_{ij} \tag{20}$$

where R_{ij} is a group of connected pixels around location (i, j).

Max-Pooling layer selects maximum value from applied kernel window region. Where R_{ij} represents maximum pooling region j in feature map at i^{th} index of each element within it; s denotes the pooled feature maps.

$$s_j = \max_{i \in R_j} \alpha \cdot i \tag{21}$$

4) SoftMax regression

SoftMax regression is generally used at higher layer of classifiers in multi class classification. The hypothesis function is used to predict probability of class with following Eq. (22):

$$h_{\theta}(x) = \frac{1}{1 + \exp_{\theta_T x}} \tag{22}$$

The model parameters learned with several forward and backward iterations to minimize the cost function $J(\theta)$. The cost function $J(\theta)$ is represented with following Eq. (23):

$$J(\theta) = -\frac{1}{m}\left\{\sum_{i=1}^{m}\sum_{j=1}^{k} ly^{(j)} = i \log p\left(y^{(j)} = j \middle| x^{(i)}; \theta\right)\right\} \tag{23}$$

In SoftMax regression, the possibility of classifying input x into output class label j is:

$$\log p\left(y^{(j)} = j \middle| x^{(i)}\right) = \frac{e^{e_j^T x_{(i)}}}{\sum_{l=1}^{k} e^{e_j^T x_{(i)}}} \tag{24}$$

IV. Classification Technique with Transfer Learning and Designing Proposed Convolutional Neural Network (CNN) Model

Performance of the model varies with input image size, number of layers, kernel size, number of filters, type of optimizer as well as layer connection pattern. Six representative models with different fundamental layer organization and connection selected for experimentation. Table II provides details about criteria used for model selection. VGG16 is a deep network with a smaller filter size. Inception is based on the Hebbian principle which states that "neurons that fire together, wire together". This architecture is deep with increased width using parallel layers. ResNet's deep architecture has the disadvantage of vanishing gradient since each layer derives new value from previous layers which can be very small, near to zero. ResNet uses skip connection which connects layer output to next layers as well as by hoping next few layers output. MobileNet architecture designed for embedded and mobile devices, it reduces parameter size by depth-wise and point-wise convolution. DenseNet architecture has all previous layers connected to the next layer to avoid vanishing gradient and bottleneck point-wise convolution to reduce parameters.

TABLE II. Transfer Learning and Model Selection Criteria

Model name	Reason for selecting pretrained model	Key points for comparison with proposed model
VGG16	Simpler architectural model, extraction of more complex features with deeper convolutional layers, smaller filters 3×3.	Similar architectural design but image size used is 224 against 160 of the proposed models. To understand the impact of layer count and size reduction.
Inception	Parallel convolutions, all outputs are concatenated, dimension reduced due pointwise filters, various filters sizes in same layer.	Selected the fastest model for training to comparison against the proposed model.
ResNet	Deeper network to learn complex features, skip connection and residual connection eliminates vanishing gradient issue associated with deeper network, widely used in benchmarking.	To understand performance of proposed model against established deeper model selected for experimentation.
DenseNet	Architecture allows each layer features are passed to all subsequent layers. Feature reuse and dense connection allow gradient flow smoothly during backpropagation avoiding vanishing gradient.	Comparison of rich feature set and intricate patterns learning against proposed models design pattern performance.
MobileNet	Architecture for portable devices, depth-wise and pointwise convolution. Model parameters reduced by width multiplier and resolution wise multiplier.	Selected widely accepted mobile device model for comparison with proposed model.
Transformer (ViT)	Improved performance with local and global feature learning with self-attention mechanism, different parts of images processed simultaneously to learn contextual features effectively.	Self-attention-based ViT model capturing long range dependencies effective for resilience to distortions compared with proposed work with real field images.

A. Visual Geometry Group (VGG) 16

Alex-net is the first deep CNN proposed in 2012 to solve the ImageNet challenge. VGG16 is based on AlexNet proposed by Visual Geometry Group (VGG) at the University of Oxford. VGG16 architecture has 224×224 RGB image input, first and second block with two convolution and one max pooling. Whereas third, fourth and fifth block have three convolution and one max pooling each, totaling 13 convolution layers, five pooling layers and last block has three fully connected layers. Traditional CNN model used 7×7, 11×11 kernels but VGG16 used smaller sized kernel with 3×3 convolution with stride of two, 2×2 pooling, and ReLU activation function [40, 41]. Fig. 3 provides architecture details of VGG16 in pictorial format.

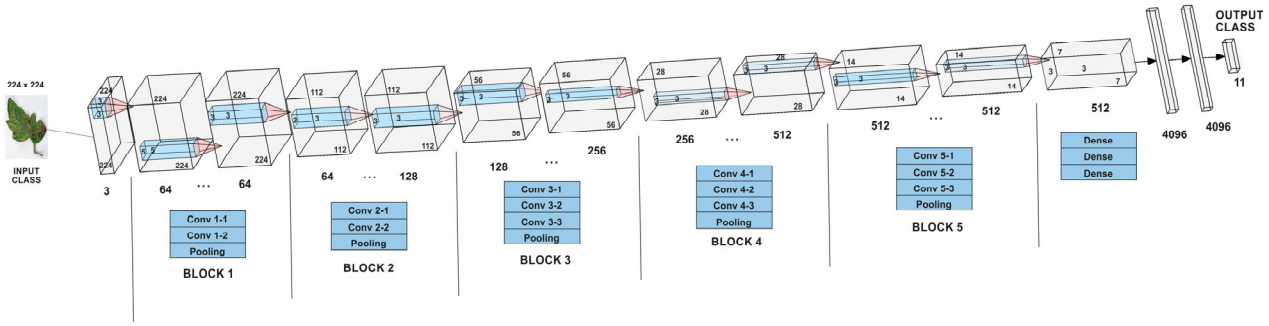


Fig. 3. VGG16 architecture.

B. Inception

VGG architecture stacks together a series of convolution, ReLU and pooling and dense layers. Inception v1 popularly known as GoogLeNet proposed in 2014. Inception architecture uses parallel convolution, pooling layer as well as filter of size 1×1 to reduce dimension. Computation cost of this architecture is much lower than VGGNet. The Inception model learns from parallel layers of different sizes and scales. Fig. 4 provides architecture details of model. Each image has a different location and size information. It is difficult to choose right size filters. Depending on information size, multiple filters with multiple sizes need to be used. Small information is retrieved by small filter size as well as bigger information is by bigger filter size. The idea of Inception is to capture information accordingly. Max pooling layer applied after convolution. Deep neural network's higher of layers number leads to over-fitting. Inception has parallel layers becoming a wider network. More parameters count requires more computation resources, reduce the number of parameters to reduce dimension. This is possible by the Inception module's liberal use of parallel structures and dimensional reduction. It lessens the effect of structural alterations on adjacent components. The number of input channels applied with 1×1 convolution before the 3×3 and 5×5 convolutions. Dimensions reduced by 1×1 convolutions are cheaper than 5×5 convolutions. Google-net architecture is the first version of Inception architecture that has 22 layers. All the channels in the Inception model are concatenated i.e. (48×48×(64+16+8+32)) = 48×48×120. Inception V2 and V3 use of RMSprop optimizer, batch normalization in the dense layer and label smoothing regularization with 0.2% dropout led to reduction in the error rate.

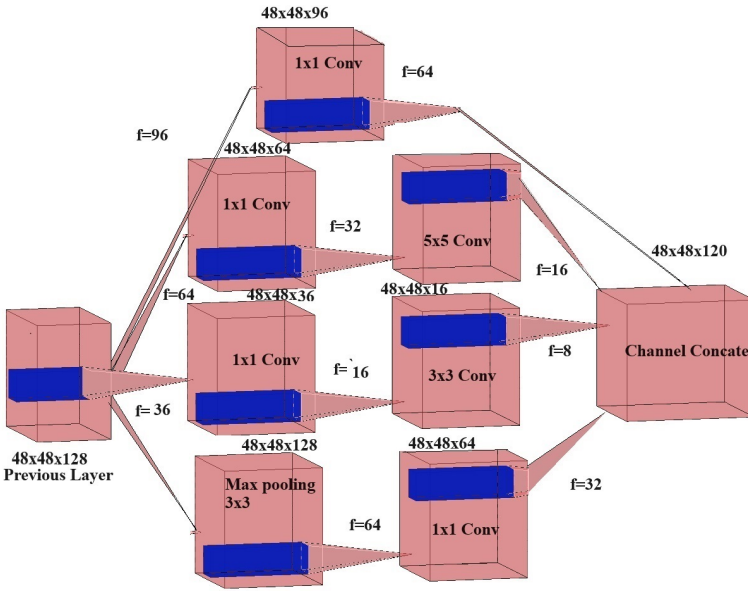


Fig. 4. Inception architecture.

C. ResNet

ResNet is a CNN that won the Image-net challenge in 2015. It contains 150+ layers and overcomes the problem of vanishing gradients that tend to negatively affect network performance. Vanishing gradient problem occurs when the gradient is propagated backwards to previous layers which result in very small gradient value. Back-propagation is the derivatives chain rule of calculus repeated multiplication makes weight very small till reaching back to earlier layers. Fig. 5 provides architecture details of ResNet in pictorial format.

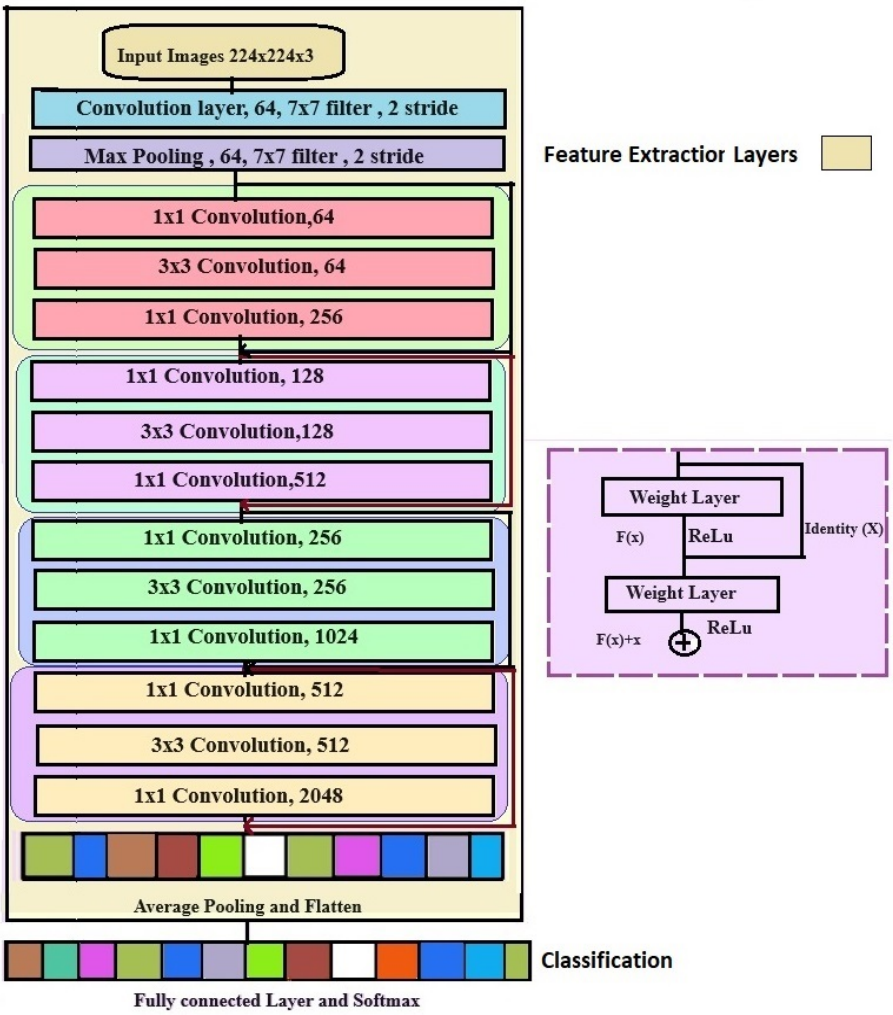


Fig. 5. ResNet architecture.

ResNet includes a skip connection concept which enables training of multiple deep layers without vanishing gradient problem. Addition of the original input with the output convolution block carried out. Missing some layers will not reach a very small value of gradient. Other network losses calculated with the input layer but in ResNet are adding actual input with output.

X input Y is output $Y = F(x)$.

If we make $F(x) = O$ then it is easy for us to make input equal to output.

$$Y = \begin{cases} X + F(x) \\ X + O \\ X \end{cases} \tag{25}$$

Regular networks learn from Y whereas ResNet learns from $F(X)$ and the target is to make $F(X) = 0$ then only to make input equal to output. ResNet has variants from ResNet-18 to ResNet-200.Some examples of ResNet models are ResNet-50, ResNet-101 and ResNet-152 with 50, 101, 152 layers respectively. Research work in paper evaluate ResNet 50 with total 24,691,595 parameters out of which 24,638,475 are trainable whereas 53, 120 are non-trainable.

D. DenseNet

DenseNet each layer takes input from all previous layers and passes to the next layer. Network efficiently improves information flow and is represented with $L(L+1)/2$ where L layers are directly connected. Each dense block has a layer which receives the combined output of all previous layers as its input. Dense block has n layers, and each dense block layer generates k feature maps called as the growth rate, the l-th layer has $K\times(l+l_o)$ input feature maps (where l_o is the count of input channels to the dense block).

Dense blocks connected with transition layers. The main use is reduction of feature maps and the spatial dimensions

down-sampling. Reduction helps in obtaining computational efficiency and the network compactness. A transition layer composed of batch normalization to normalize the feature maps, 1×1 convolution to decrease the count of feature maps and average pooling to down samples the spatial dimensions. DenseNet has versions with a number of layers 121, 169, 201 and 264. Research work evaluated DenseNet 121with total 7,238,212 parameters out of which 200,708 are non-trainable whereas 7,037,504 are trainable. Fig. 6 provides architecture details of DenseNet model.

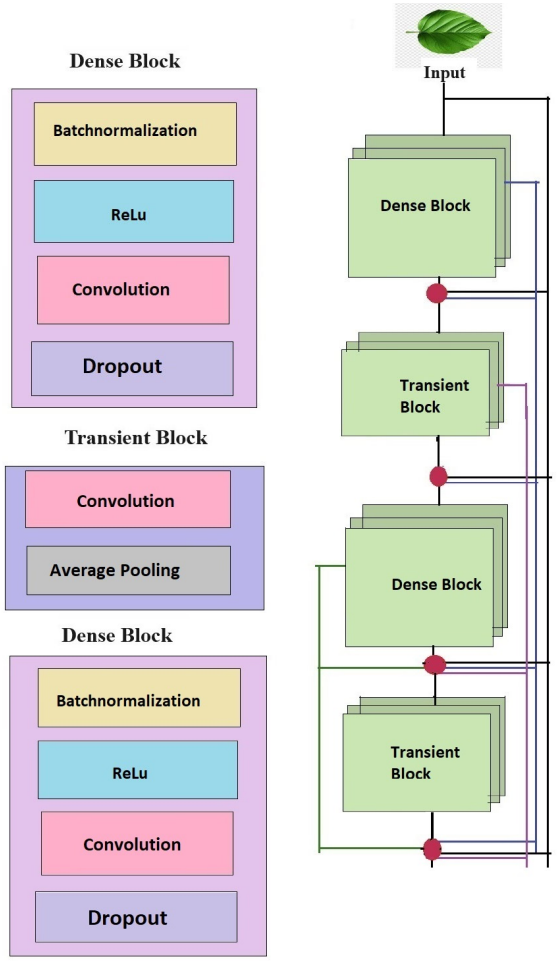


Fig. 6. DenseNet architecture.

E. MobileNet

Mobile net is proposed in 2017 specifically lightweight convolutional neural network architecture for computer vision applications. Point wise convolution 1×1 along with depth wise separable convolution reduced computation cost. Each convolution of kernel size 3×3 immediately follows with depth wise and point wise convolutions. Fig. 7 provides pictorial details of MobileNet model.

Computation of convolution cost calculated with following Eq. (26).

$$F_K \cdot F_K \cdot I \cdot O \cdot F_d \cdot F_d \tag{26}$$

where $F_K \cdot F_K$ are kernel; I is input image array; O is output label class; $F_d \cdot F_d$ are feature map.

Given input and the kernel convolution produce the effect of filtering and joining.

Input channel depth-wise convolution is applied with a single filter, afterwards this output is joined by a point wise filter.

If K is the depth of kernel size of output feature map. Modified convolution is represented by:

$$F_K \cdot F_K \cdot I \cdot F_d \cdot F_d \tag{27}$$

MobileNet support any input size greater than 32×32, with increased image sizes better performance is achieved. This model provides promising results in comparison to other pre-trained models like Inception and ResNet. Convergence time is 8 to 9 times lesser than standard CNN.

F. Vision Transformer (ViT)

Latest industry trend using ViT in computer vision task to perform better in terms of accuracy and computational efficiency. CNNs with comparison of ViT models need to be evaluated. ViT models approach uses self-attention mechanism which allows the model to learn relations between different parts of the image. Model calculate score based on importance features in patches and capturing the most relevant feature in recognizing image. This helps the model make better sense of the image and perform various classification tasks related to computer vision. ViT architecture based on splitting image in patches. Fig. 8 provides architecture details of ViT.

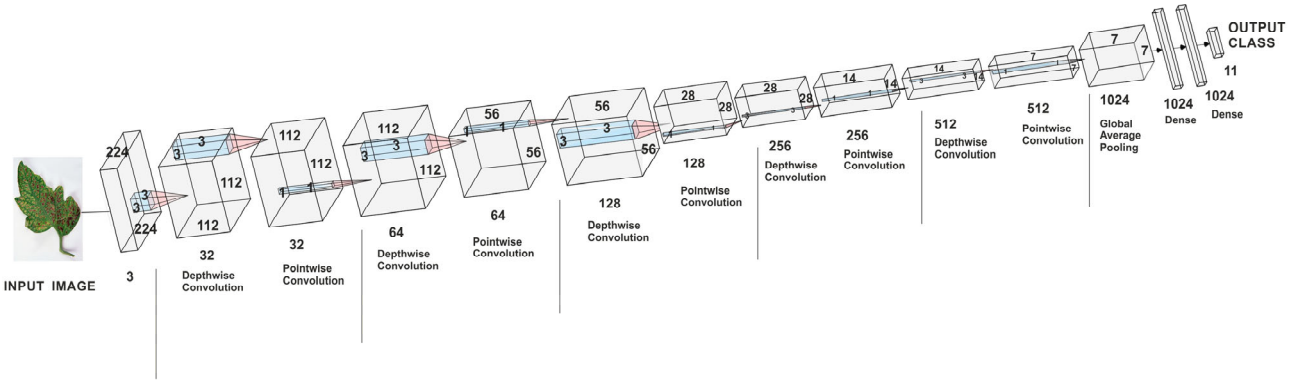


Fig. 7. MobileNet architecture.

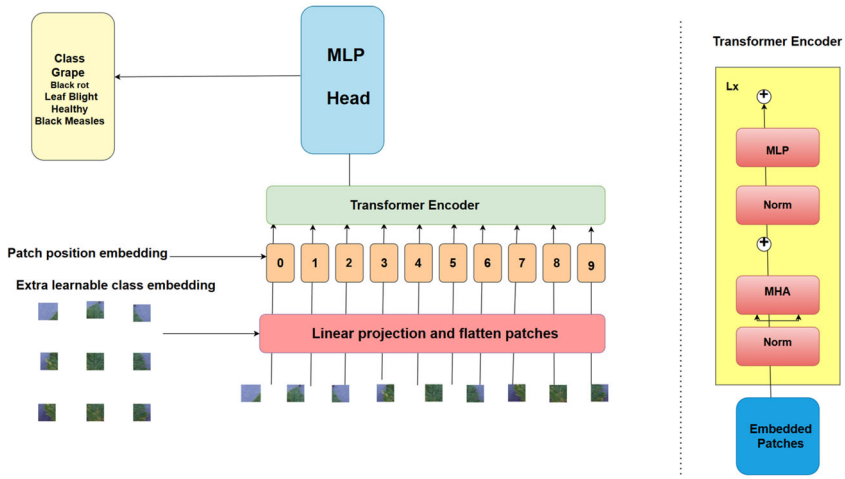


Fig. 8. Vision Transformer (ViT) architecture.

Every patch consists of smaller information size; tiny image patches sequence converted into token flatten patch. Later tokenized patches provided as input to transformer. Each patch processed by linear projection and position encoding. ViT linear projection will reduce information in lower dimension vector. This operation is very similar to weight matrix multiplication. Training process along with reducing dimension learns parameters and it extracts essential features. ViT transformer encoder captures dependencies within tokenized patch image acquiring both local and global dependency. ViT model later feed output to feed forward network to map image to information grape disease class.

G. *Designing Proposed Convolutional Neural Network (CNN) Model*

1) *Customised sequential model with input size 128*

CNN designing is crucial. Various hyperparameters, number of layers, input size, optimizer, batch normalization and optimizer are analyzed then used in proposed design. Initial experiment carried out with input size 128×128. This design has three blocks in which the first block has convolutional layers with ReLU activation function, followed by max pooling layer. Second and third blocks have consecutive two convolutional layers to amplify the dominant feature and maximum pooling. Convolution layer uses filter size 3×3 and stride of one and maximum pooling uses window size 2×2. Number of kernels in three blocks are decreasing in number from 32 to 16, and 8 for the last block. After these three blocks the model contains flatten and two dense layers in the end. Hyper-parameter optimizer is Adaptive Moment Estimation (Adam), dropout is 0.2 and the batch size is 64. Sequential layered architecture evaluated for grape augmented data-set. Proposed design has a total eleven layers and trainable parameters are 158,587. Fig. 9 provides architecture details of customized model with input image size 128.

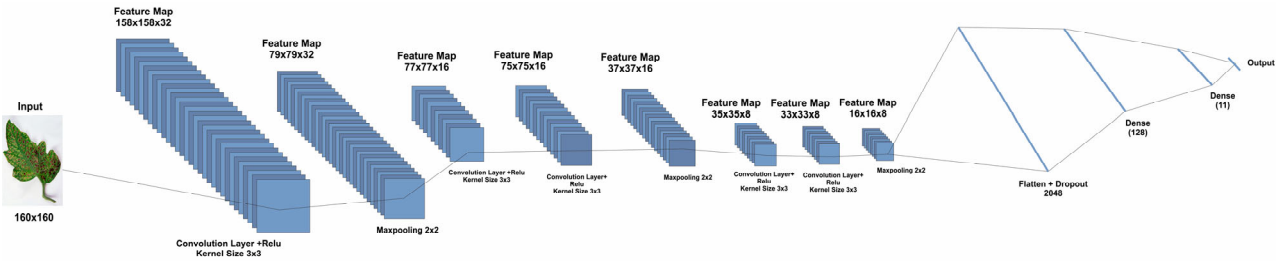


Fig. 9. Customized sequential with input size 128 architecture.

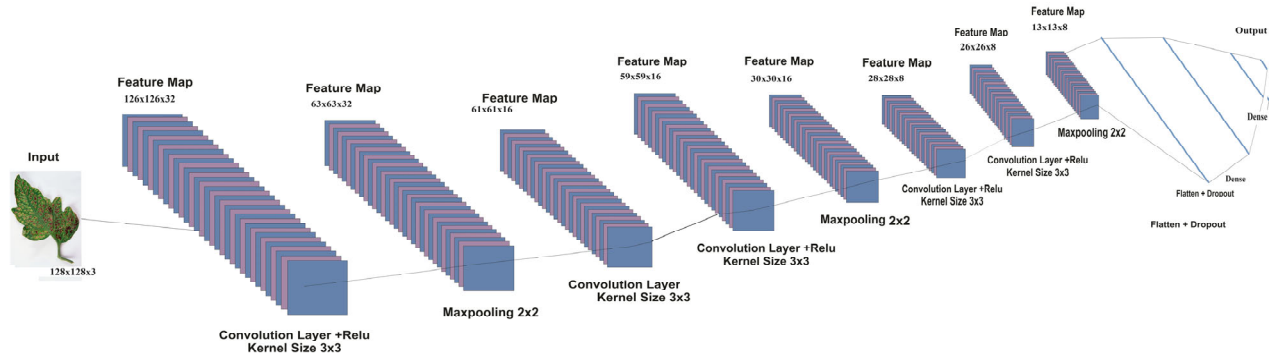


Fig. 10. Customized sequential with input size 160 architecture.

2) Customized sequential with input size 160

Increasing image size increases the amount of information carried in the image. The proposed design with an input size of 160 uses sequential custom-designed models. Sequence of layers is the same as the first proposed model with input size 128. During forward pass input convoluted with kernel function generating activation features that need to be normalized with function and passes to the next layer. During back propagation weight adjustment can reduce weight generating problems of vanishing gradient. ReLU is a non-linear activation function having nonnegative value resolving problems of vanishing gradient. CNN use supervised learning where features are extracted by lower layers and higher layers map these features with output. Backward pass of the network adjusts weight to reduce error. Error or loss is the difference between predicted and actual values. Hyperparameter optimizer used in proposed design is Adam since stochastic gradient descents capabilities are extended in this optimizer based on learning rates and their second moments. Regularization carried out with dropout is 0.2 and the batch size is 64. Dense layer maps learned features to output using SoftMax activation. Total trainable parameters are 273,275. Experiment trains models up to 192 epochs. Fig. 10 provides architecture details of proposed customized model with input image size 160.

V. Data Set and Experimental Setup

Performance of the model depends on architecture of model and dataset. Particular models can be suitable for certain datasets and performance varies with selection of hyper-parameters. CNN training classes are scalable.

A. Details of the Dataset Used for Training

The original Plant-Village dataset for four grape disease classes is 4748. Plant-Village image dataset is expanded with image augmentation operations like rotation, flipping and scaling. Images acquired in the field are segmented with seven steps proposed segmentation algorithm. Grape dataset contains Plant-Village and real field images have total count 8822. Later split to 7222 for training and validation. Image count for testing models is 1600. Plant-Village image dataset is expanded with image augmentation operation like rotation, flipping and scaling. Images acquired infield are segmented with seven steps proposed segmentation algorithm. Dataset used for experimentation has three categories: training, validation and test data set. Paper adopted 80% training and 20% validation split since analysis showed good results with this split. Number of images used for testing each class is 400. Fig. 11 provides class wise distribution of image dataset. First class Black Rot image count is 1888. Second class Esca (Black Measles) uses 1920. Third class Healthy has 1692 images. Final class category Leaf Blight (Isariopsis Leaf Spots) uses 1722 images. Table III provides details about hyperparameter selection in model optimization.

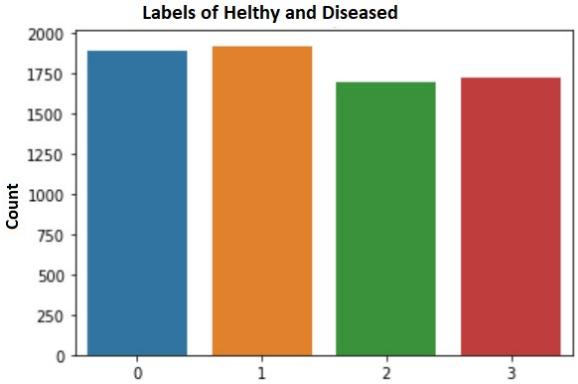


Fig. 11. Dataset distribution: 0 is Black Rot; 1 is Esca (Black Measles); 2 is healthy; 3 is Leaf Blight (Isariopsis Leaf Spots).

TABLE III. Hyperparameter Selection for Model

Name of hyperparameter	Hyper-parameter selection for model
Epochs	Empirical observations shows that learning starts converging well after 30 epochs. If training and validation accuracy not improving, early stopping is selected. If model is improving, epochs are extended up to 192 keeping RAM limitation in consideration.
Learning rate	Rate at which function adjusted weight during backpropagation process, precise result with smaller value, value 0.001 proven in successful result across research literature.
Batch size	Large batch size fastens model training and requires more RAM. Balancing between memory and performance 64 batch size selected.
Optimizer	Selected based on task, dataset and Adam is a popular choice in image classification tasks.

B. Experimental Setup

Machine with Intel Core i5 7,400 CPU 3.00 GHz, RAM 16 GB and Ubuntu 20.04 is the execution platform. Coding and development carried out with Jupiter notebook. Real field true-color RGB image collected with Samsung 12-megapixel Camera.

The proposed research carries out real field images of grape along with Plant-Village images for grape disease identification and classification.

(1) Real field images are pre-processed to enhance features with threshold and masks identified to locate the area of interest.
(2) Color, shape and texture-based features designed to identify non leaf and leaf.
(3) Later Segmentation carried out with contour-based Grab Cut algorithm and multi color threshold to separate out complex background from foreground.
(4) Accuracy of segmentation is evaluated and improved.

Segmented Images are added to the Plant-Village dataset for expanding dataset and meet the requirement for a large number of images.

TensorFlow Application Programming Interface (API) library used for data-flow programming which converts image multidimensional array and makes use of computational graph concept for computation multiple layer functioning, automatic differentiation used in backward pass of neural network architecture to reduce loss and improve accuracy, and the adaptability of the TensorFlow python API structure makes experimentation flexible with different hyperparameter.

The Keras (API) is built on top of TensorFlow to construct CNN architecture layers and models. The model learns to associate images and labels. TensorFlow converts set of images in dataset object with data generator with NumPy arrays and Panda's data frames irrespective underline processor. Functionality fit is used to train model with data generated for training and validation. Later save trained model which used with functionality predict to get result. Table IV provides class wise distribution of augmented image dataset. Augmented images dataset re-sized to fit architecture and dataset split carried out to train and validation sets. The performance of architecture is evaluated based on network shape, network parameter count, target model and convergence time.

TABLE IV. DISTRIBUTION OF IMAGES FROM AUGMENTED DATASET

Name of disease	Plant-Village without images augmentation	Plant-Village images	Total number of infield images	Total number of images	Total number of images for training	Total number of images for validation	Total number of images for testing
Black_Rot	1180	1863	25	1888	1510	378	400
Esca_(Black_Measles)	1383	1920	0	1920	1536	384	400
Healthy	1145	1617	75	1692	1354	338	400
Leaf_blight_(Isariopsis_Leaf_Spot)	1076	1712	10	1722	1378	344	400
Total	4784	7112	110	7222	5777	1445	1600

A total of eight experiments is conducted using different models and data prepared with augmentation, resize and rescale. TensorFlow keras library utilizes n-dimensional array products to extract features from multidimensional images. Max pooling extracts dominant features by sliding windows on extracted features and reduces dimension of image weight. Flatten layer converts multi-channel data into a single vector and passes to hidden dense layers. The SoftMax layer is the output layer to make sure outputs are mapped to categorical data type output class labels for classification. Majority hyper-parameters are kept common in all experiments to compare performance of models. Batch size of 64, Adam optimizer, ReLU activation and SoftMax classifier used for all experiments.

Finally trained and validated models can be evaluated with a test image data-set. Predict functionality of TensorFlow Keras API to find probability of output disease class. Maximum probability is selected for diagnosed disease class. Acceptable model with satisfactory results is deployed on Android Mobile applications for in the field diagnostics.

VI. SYSTEM ARCHITECTURE

Training on eight different models is carried out to select the optimal model for deployment. The first experiment with the VGG16 model having trainable parameters count is 14,714,688 whereas non-trainable parameters count is 100,356 while the total sum of parameters is 14,815,044. This deep model with series of layer requires 22 h for training four disease classes. The second experiment uses Inception version 3 with factorized convolutional executes in parallel. It uses trainable parameters 21,802,784 and non-trainable parameters 563,211 in total parameters 22,365,995. Third experiment using ResNet with 50 layers has trainable parameters 24,638,475 and non-trainable parameters 53,120 in total parameters are 24,691,595. Fourth experiment using DenseNet has 121 layers trainable parameters 6,953,856 and non-trainable parameters 83,648 in total parameters are 7,037,504. Fifth experiment using MobileNet version 2 with additional inverted residual blocks with bottle-necking features than version 1. It is very similar to the original MobileNet. It has a drastically lower parameter count than the original MobileNet which has total parameters 3,549,995 out of which trainable parameters 3,538,984 and non-trainable parameters 11,011.

Sixth experiment with Vision Transformer (ViT) uses input image size 96×96 to reduce number of model parameters. Time required for training and model size are reduced. Hyper parameters are set for transformer which includes learning rate of 0.001, patch size 8, number of layers 6 and batch size 128.Total parameters count of this architecture model is 21,499,595.

Seventh experiment proposed design with 11 layers and 128 input image size, used total parameters 158,587 out of which trainable parameters are 158,587 and zero non-trainable parameters. Eighth final experiment increases input image size from 128 to 160 and has 11 layers. This model's total parameters count is 273,275 which are all trainable parameters. Fig. 12 provides proposed system details during training phase. All experiment saves model after training in format .h5 and Keras.

Mobile vision-based system is used to diagnose grape disease. Android is an operating system for mobile devices that creates interactive and engaging experiences for users. This is an embedded operating system for hand held consumer devices having on chip all resources to run applications. Open-source Linux platform is the base and Java programming language is used to build it. The deployment scenarios for mobile applications using the inference are performed locally at the device terminal. Farmers, researchers, and home gardeners can use grape care plant disease diagnostic applications using graphical

user interface. The proposed lightweight model is deployed on Android mobile phones. Fig. 13 provides proposed system details during deployment phase. Android project imports required OpenCV library and lite model. Trained lightweight .h5 and Keras format model converted in TensorFlow lite format. TensorFlow converter LiteRT converts .h5 model in tflite format without reducing accuracy. Model weights precision reduced to 16 bit floats and 8 bit integers for CPU and hardware acceleration. This requires fake quantization during training with range information using a calibrated dataset. During hypothesizing weight and activation in depth computation required for integer than float values.

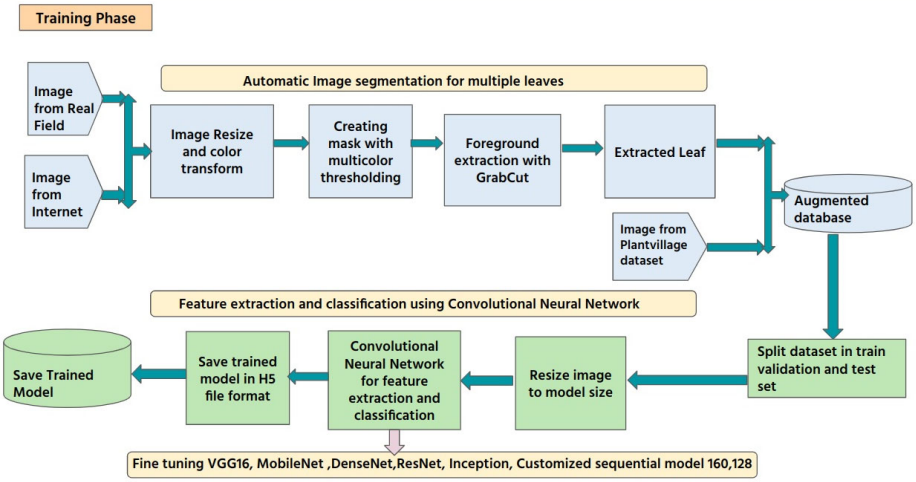


Fig. 12. Proposed system training phase diagram.

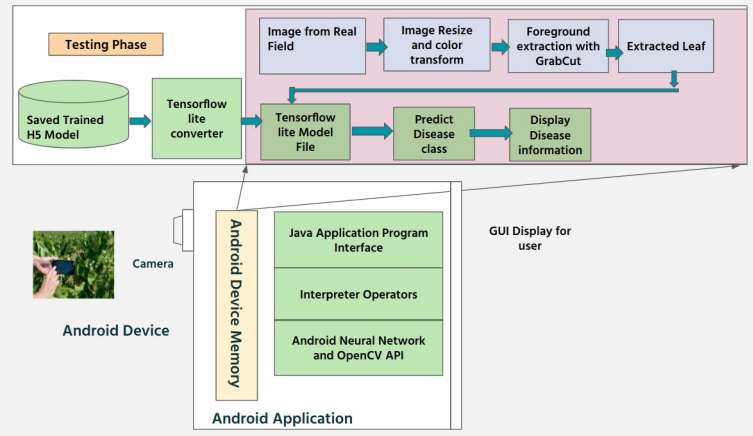


Fig. 13. Proposed system deployment architecture diagram.

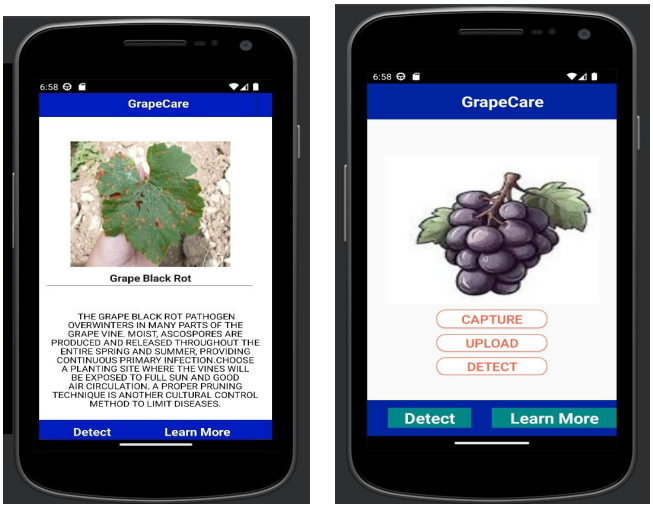


Fig. 14. Graphical user interface of proposed system.

Two scenarios used in paper for deployment of system. The mobile application runs the proposed lightweight model, and then results shown on device display. Android application provides a graphical user interface with three buttons: first button to capture images in the field with device camera, second button to upload from device memory and third button to diagnose disease and display results with highest confidence. Disease information gathered with the help of agriculture experts made available for users. Fig. 14 presents mobile embedded system. Other scenarios for embedded system are Raspberry Pi 5 with 3.5 inch touch display screen setup. Crop disease diagnosis Graphical user interface prepared with tkinter library, programs and tflite model stored on device. System provides option to make python function call for operation such as load image and detect disease. Fig. 15 presents Raspberry Pi based embedded system.

Fig. 15. Graphical user interface of proposed system on Raspberry Pi kit.

VII. Results and Discussion

A. Comparative Analysis

Eight architecture models used for experimentation carried out by keeping the majority of hyper parameters constant. This will make comparisons of models' performance justifiable due to the same dataset and conditions used for training and testing. Since disease diagnostic is crucial for accuracy and embedded system have limited device space and processing power, accuracy and model size used as prime parameter.

First experiment with VGG16 is a very deep, simple network, it uses kernel size 5×5 and 3×3 and is interesting to evaluate performance of pipe-lined networks. It requires 22 h for training models for 192 epochs. Model reaches training accuracy 1.0 for some batches and 0.97 validation accuracy after 70 epochs and maximum validation accuracy 0.9924. The VGG16 model achieves a good average training accuracy of 0.98 while validation accuracy 0.99. Trained model is saved in .h5 and Keras format with size 114.7 MB. The model tested with 4000 images from a different set than used for training achieves a good accuracy of 0.989. Leaf blight and healthy classified with 100% accuracy and Black rot disease class achieves accuracy 96.5%.

Second experiment with Inception is parallel balanced between the depth and width and kernel split along width in size of 1×1 and 3×3. Optimal performance with a constant amount of computation can be achieved. Inception V3 architecture's performance and efficiency used to reduce the computational cost of convolutions. Training and validation accuracy curve plots values 0.92 and above. Training for 192 epochs reduces time drastically to 5.76 h. Saved model size is 99.6 MB. Model achieves training accuracy of 0.9676 and validation accuracy 0.9659 but testing accuracy is less comparative to VGG16, it is 0.9075.

ResNet used for third experiment has 34-layer plain network architecture with skip connection residual blocks used to evaluate grape disease identification. Model trained for 192 epochs, training and validation accuracy curve plots values drastically drops for validation accuracy at multiple epochs like 28, 51, 60, 79, 101, 110 but after 175 epochs improves. Training and validation loss curve maintained constant near to 0 after 30th epoch. Plots architecture requires 11.84 h for training and saves model for later testing with size 300 MB. Model saved after 192 epochs has training accuracy 0.9989 and 0.9879 validation accuracy improves testing accuracy to 0.99.

Fourth experiment using DenseNet with each layer connected to the previous layer, dense and transient blocks produce high-level features in later layers of the network. Parameter count of DenseNet is four times less than ResNet but training time required is 13.7 h. This architecture saves the model to reduce size 60.6 MB which is less compared to previous three architectures. Model trained for 192 epochs and training and validation loss curve plots surprising shows validation loss lesser than training loss. Training accuracy is highest 1 and validation accuracy is 0.99 and testing accuracy is 0.98.

MobileNet is the fifth architecture used for experimentation that reduces kernel size to 1×1 and depth wise convolution. This architecture drastically reduces training time to 3.23 hours and saves a model with size 17.5 MB. MobileNet Model trained for 192 epochs, training and validation accuracy and loss curve plots are smooth incremental and incremental respectively with less fluctuations. This architecture achieves training accuracy is 0.84 and validation accuracy 0.82. When saved model tested gives accuracy of 0.84.

The performance is evaluated in five previous experiments with input image size 224, sixth experiment with ViT uses reduced image size of 96×96. Image is divided into patches and hyper-parameters selected to reduce training time. Model trained for 192 epochs achieving training accuracy of 0.9937 and training loss of 0.0274. Training time required is 13,890.75 seconds. Validation accuracy is 0.9948 and loss is 0.0213. Top 5 accuracy of ViT is 1.00.

Seventh experiment is based on proposed design which uses reduced image size. Proposed first sequential lightweight network with input size 128×128 is interesting to analyze due reduced image size. This architecture uses 3×3 convolutional kernels and consecutive convolution to enhance extracted features. Reduced parameter count requires 1 h for 192 epochs. Training and validation accuracy curve plots are smooth incremental whereas training and validation loss lesser fluctuations after 30 epochs. Training accuracy is 0.9964 and validation accuracy is 0.97. Model saved after training is 2.2 MB when tested for a different set of images gives 0.96.

Second proposed network increases input image size to 160×160 increasing the amount of information. Eleven-layer proposed architecture requires 1.4 h for training up to 192 epochs. After training model saved with 3.6 MB and training accuracy is 0.9953 and validation accuracy 0.9986. Testing accuracy of proposed model is 0.989.

B. Discussion about Experiment Results

Experimentation results of eight models are presented in Table V and all results in confusion matrix format are shown in Table VI. Training and validation accuracy and loss plots are presented on Table VII. Eight models evaluated for four classes of grape with the same dataset for training and testing. There is not any specific uniformity in wrong classification for any particular class. ResNet is heaviest model with size 300 MB followed by VGG16 114.7 MB. Parallel designed model Inception is the four-time faster trained model in comparison with VGG16. MobileNet is the fastest pretrained model which merely required hours for training. MobileNet saved model size 17.5 MB is smallest among all pretrained model. Training and validation accuracy is lowest for MobileNet in comparison with other seven architectures. Accuracy is very crucial in disease diagnostic tools. Proposed first model with input image size 128 fastest trained model among all eight models as well as lowest in model size. Proposed second model with eleven layer and increased input image size 160 improves amount of information for training. Training, validation and testing accuracy of this second proposed model is best among all eight model. Only 6 out of 1600 test cases output class diagnosed wrongly by proposed model in comparison with next best ResNet and VGG16 which has 14 and 17 out of 1600 test cases respectively diagnosed wrongly. The Inception model performs well in training time but disease class Leaf_blight_(Isariopsis_Leaf_Spot) accuracy is 0.75 whereas MobileNet model reduces for disease class Esca_(Black_Measles) to 0.74 which is lowest in all models. DenseNet architecture takes higher training time than ResNet but reduces model size to 60.6 MB which is five time lesser than ResNet model size. Research work reduces training time with new design architecture by reducing parameter count. Proposed lightweight model is best among eight studied models with accuracy of 0.9962 and model size of 3.6 MB. Fig. 16 provides eight models result comparison based on model size, training time and testing accuracy.

TABLE V. RESULT OF EXPERIMENTS WITH EIGHT MODELS

Name of model	No. of Epochs	Training Time	Model Size	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Testing Accuracy
VGG16	192	79,296	114.7 MB	0.9844	0.0182	0.9924	0.0253	0.9893
Inception	192	20,736	99.6 MB	0.9676	0.7561	0.9659	0.6446	0.9075
ResNet	192	42,624	300 MB	0.9986	0.0053	0.9879	0.0320	0.9912
DenseNet	192	49,344	60.6 MB	1.0000	0.0040	0.9903	0.1379	0.9843
MobileNet	192	11,654	17.5 MB	0.8416	0.4938	0.8288	0.5045	0.8431
Transformer (ViT)	192	13,891	82 MB	0.9937	0.027	0.9948	0.0213	0.9942
Cust. Seq. 128	192	3,232	2.2 MB	0.9964	0.0110	0.9702	0.1144	0.9631
Cust. Seq. 160	192	5,012	3.6 MB	0.9953	0.0145	0.9986	0.0048	0.9962

TABLE VI. MODEL CONFUSION MATRIX FOR RESULTS AND PERFORMANCE MEASUREMENT OF MODEL AFTER 192 EPOCHS

Model Name	Confusion matrix representing results				
		Black_rot	Esca_(Black_Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
VGG16	Black_rot	386	3	0	0
	Esca_(Black_Measles)	14	397	0	0
	Healthy	0	0	400	0
	Leaf_blight_(Isariopsis_Leaf_Spot)	0	0	0	400

Performance measurement of model				
	Black_rot	Esca_(Black_Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Precision	0.99	0.97	1.00	1.00
Accuracy	0.97	0.99	1.00	1.00
Recall	0.97	0.99	1.00	1.00
F1 Score	0.98	0.98	1.00	1.00

Inception

	Black_rot	Esca_(Black Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Black_rot	399	45	0	2
Esca_(Black Measles)	1	355	0	0
Healthy	0	0	400	0
Leaf_blight_(Isariopsis_Leaf_Spot)	0	0	0	298

	Black_rot	Esca_(Black_Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Precision	0.89	1.00	1.00	1.00
Accuracy	1.00	0.89	1.00	0.75
Recall	1.00	0.89	1.00	0.99
F1 Score	0.94	0.94	1.00	1.00

ResNet

	Black_rot	Esca_(Black Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Black_rot	387	0	0	0
Esca_(Black Measles)	8	399	0	0
Healthy	0	0	400	0
Leaf_blight_(Isariopsis_Leaf_Spot)	5	1	0	400

	Black_rot	Esca_(Black_Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Precision	1.00	0.98	1.00	0.99
Accuracy	0.97	1.00	1.00	1.00
Recall	0.97	1.00	1.00	1.00
F1 Score	0.98	0.99	1.00	0.99

DenseNet

	Black_rot	Esca_(Black Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Black_rot	391	16	0	0
Esca_(Black Measles)	9	384	0	0
Healthy	0	0	400	0
Leaf_blight_(Isariopsis_Leaf_Spot)	0	0	0	400

	Black_rot	Esca_(Black_Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Precision	0.96	0.98	1.00	1.00
Accuracy	0.98	0.96	1.00	1.00
Recall	0.98	0.96	1.00	1.00
F1 Score	0.97	0.97	1.00	1.00

MobileNet

	Black_rot	Esca_(Black Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Black_rot	331	92	3	29
Esca_(Black Measles)	57	297	16	4
Healthy	10	6	371	17
Leaf_blight_(Isariopsis_Leaf_Spot)	2	5	11	350

	Black_rot	Esca_(Black_Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Precision	0.73	0.79	0.92	0.95
Accuracy	0.83	0.74	0.93	0.88
Recall	0.83	0.74	0.93	0.88
F1 Score	0.77	0.77	0.92	0.91

Transformer (ViT)

	Black_rot	Esca_(Black Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Black_rot	399	1	0	0
Esca_(Black Measles)	1	398	0	0
Healthy	1	0	400	0
Leaf_blight_(Isariopsis_Leaf_Spot)	2	1	0	400

	Black_rot	Esca_(Black_Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Precision	1.00	1.00	1.00	0.99
Accuracy	1.00	1.00	1.00	1.00
Recall	0.99	1.00	1.00	1.00
F1 Score	0.99	1.00	1.00	1.00

Custom Seq. 128

	Black_rot	Esca_(Black Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Black_rot	391	1	3	0
Esca_(Black Measles)	3	399	0	0
Healthy	0	0	351	0
Leaf_blight_(Isariopsis_Leaf_Spot)	6	0	46	400

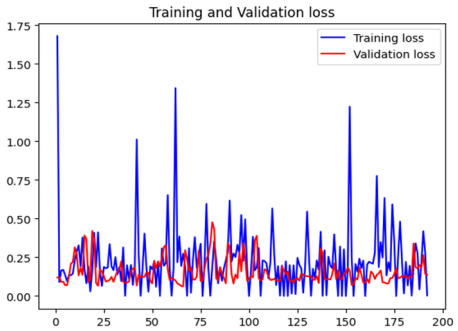
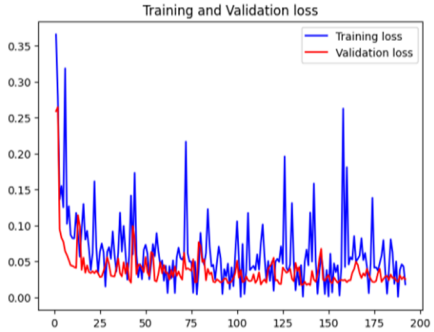
	Black_rot	Esca_(Black_Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Precision	0.99	0.99	1.00	0.88
Accuracy	0.98	1.00	0.88	1.00
Recall	0.98	1.00	0.88	1.00
F1 score	0.98	1.00	0.93	0.94

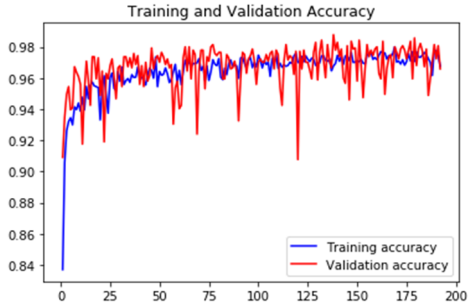
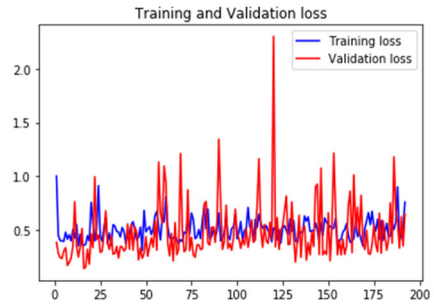
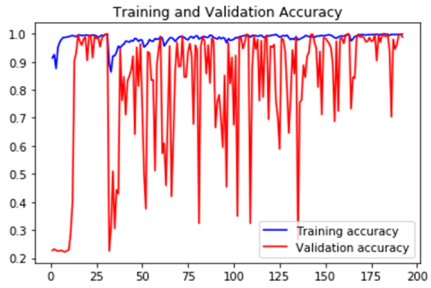
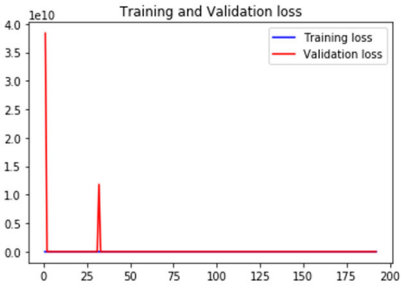
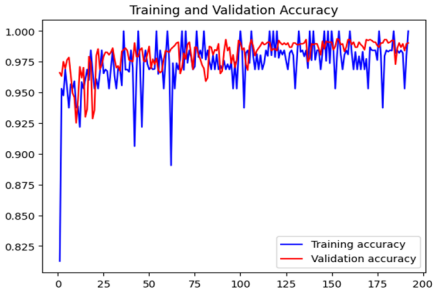
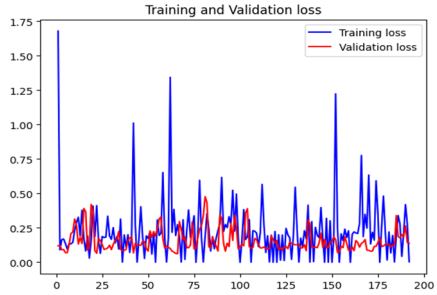
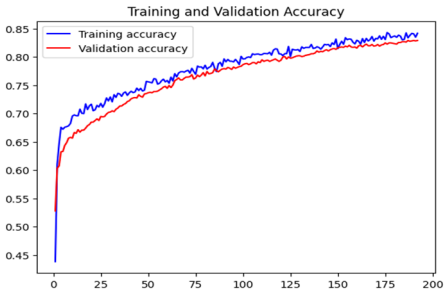
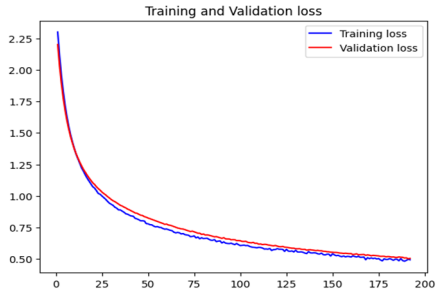
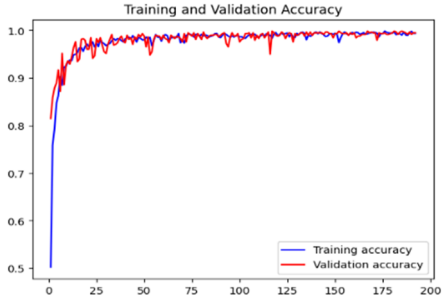
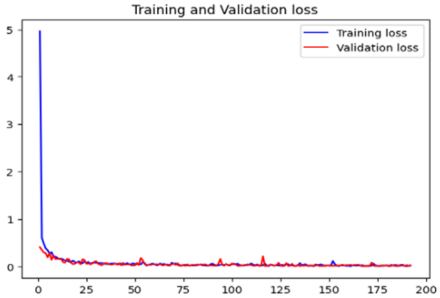
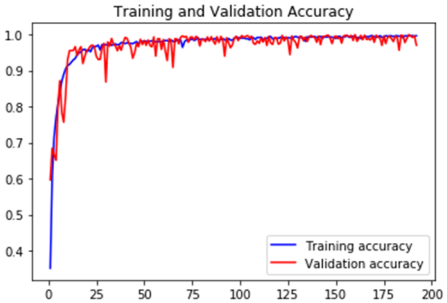
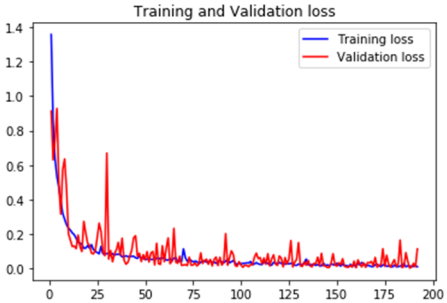
Custom Seq. 160

	Black_rot	Esca_(Black Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Black_rot	396	1	0	0
Esca_(Black Measles)	1	398	0	0
Healthy	1	0	400	0
Leaf_blight_(Isariopsis_	2	1	0	400

	Black_rot	Esca_(Black_Measles)	Healthy	Leaf_blight_(Isariopsis_Leaf_Spot)
Precision	1.00	1.00	1.00	0.99
Accuracy	0.99	1.00	1.00	1.00
Recall	0.99	1.00	1.00	1.00
F1 Score	0.99	1.00	1.00	1.00

TABLE VII. MODEL ACCURACY AND LOSS PLOT AFTER 192 EPOCHS

Model Name	Model after Epochs 192 Accuracy plot	Model after Epochs 192 Loss plot
VGG16		

Inception		
ResNet		
DenseNet		
MobileNet		
Transformer (ViT)		
Custom Seq.128		

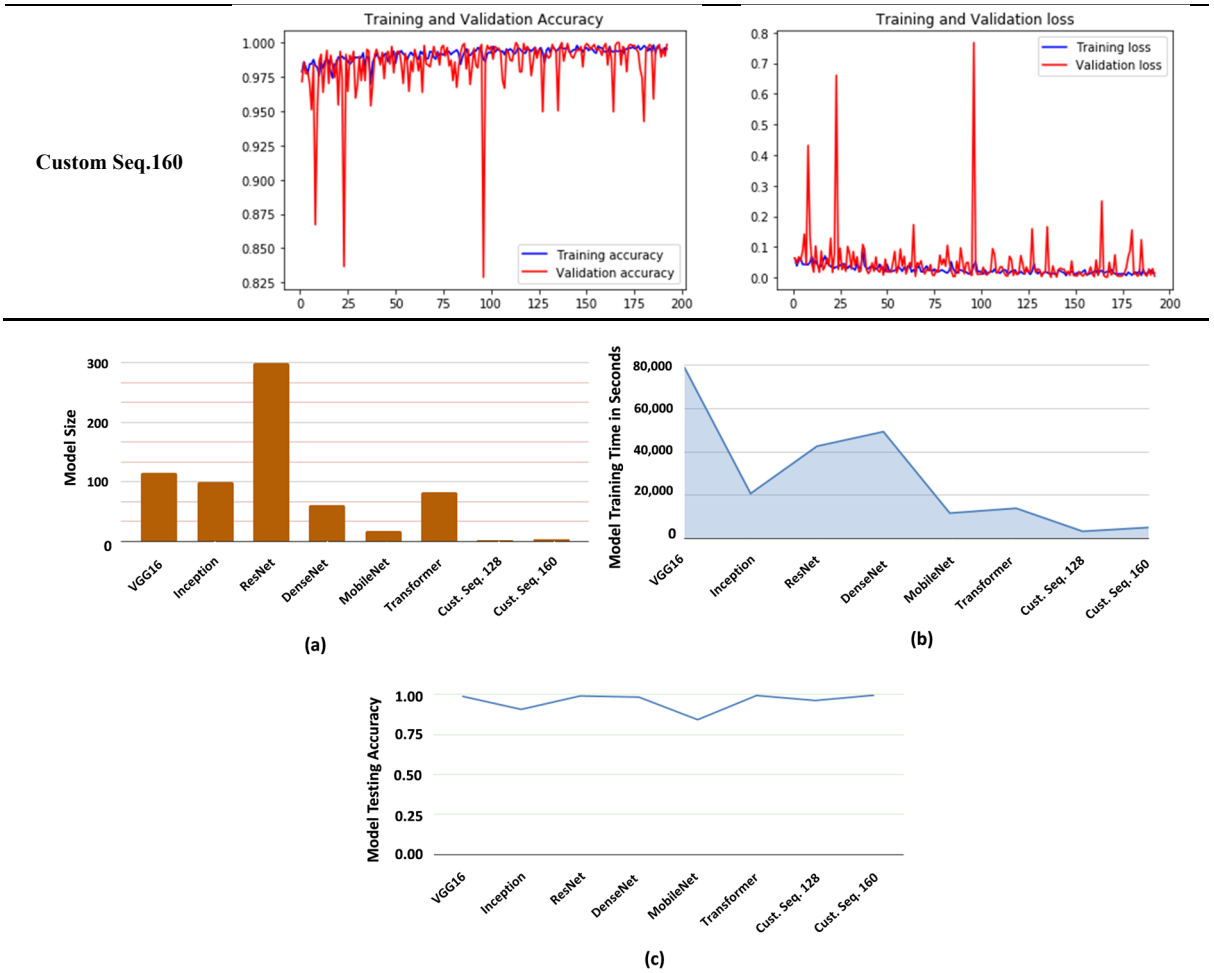


Fig. 16. Eight models result comparison. (a) CNN architecture model size. (b) Model training time in seconds vs. name of model. (c) Model testing accuracy vs. name of model.

VIII. Conclusion

Proposed segmentation method is useful for scaling crop disease dataset for more disease classes. Selection of hue, saturation and value color model make segmentation algorithm efficient in non-uniform lightening conditions. Preprocessing technique is useful during dataset preparation with capability of expanding datasets. Research work achieves goal of optimized model development through analysis of different architecture design patterns to give birds eye view in grape crop disease image classification task. Eight architectural model intensively explored with constant conditional parameters making justified comparison. Research work built and optimized to provide light weight 11-layer model with 160×160 input image size. Proposed model provides result equivalent to latest vision transformer as well as state-of-art models. Future scope is a large number of crop disease types that could be trained and tested with proposed model. This requires high end processor like graphical processing unit. Paper presented two possible scenarios of deployment of application. Development of grape care embedded system is useful as a diagnostic tool for farmers and gardeners.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Shital Karande did data collection, analysis, interpretation, software development, experimentation and wrote the paper. Bindu Garg guided the research direction and revised paper. All authors had approved the final version.

References

[1] C. Szegedy, W. Liu, Y. Jia *et al*., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 1–9. doi: 10.1109/CVPR.2015.7298594

[2] S. Jadhav and B. Garg, "A review of various agriculture systems based on IoT, data mining and cloud," *Test Eng. Manag.*, vol. 82, no. 1–2, pp. 16972–16979, 2020.

[3] S. Jadhav and B. Garg, "Comprehensive review on machine learning for plant disease identification and classification with image processing," in *Proc. Machine Learning and Data Analytics for Predicting, Managing, and Monitoring Disease*, 2022, pp. 317–334. doi: 10.1007/978-981-16-7136-4_20

[4] S. Jadhav and B. Garg, "Comparative analysis of image segmentation techniques for real field crop images," in *Proc. Int. Conf. Innov. Comput. Commun.*, 2023, pp. 1–13. doi: 10.1007/978-981-19-2535-1_1

[5] D. Khattab, H. Ebied, A. Hussein, and M. Tolba, "Clustering-based image segmentation using automatic GrabCut," in *Proc. 10th Int. Conf.*

Inform. Syst. (INFOS), 2016, pp. 108–113.
[6] S. Karande and B. Garg, "Performance evaluation and optimization of convolutional neural network architectures for tomato plant disease eleven classes based on augmented leaf images dataset," *Neural Comput. & Applic.*, vol. 36, no. 20, pp. 11919–11943, 2024. doi: 10.1007/s00521-024-09670-6
[7] V. Singh and A. K. Misra, "Detection of plant leaf diseases using image segmentation and soft computing techniques," *Inf. Process. Agric.*, vol. 4, no. 1, pp. 41–49, 2017. doi: 10.1016/j.inpa.2016.10.005
[8] N. Petrellis, "Mobile application for plant disease classification based on symptom signatures," in *Proc. 2nd Int. Conf. Comput. Sci. Appl. (CSA)*, 2017, pp. 1–6. doi: 10.1145/3372938.3372983
[9] E. J. A. V. Pascual, J. M. J. Plaza, J. L. L. Tesorero, and J. C. De Goma, "Disease detection of Asian rice (Oryza Sativa) in the Philippines using image processing," in *Proc. 3rd Int. Conf. Comput. Sci. Appl. (ICCSA)*, 2019, pp. 1–5. doi: 10.1145/3366650.3366676
[10] A. Picon, A. Alvarez-Gila, M. Seitz, A. Ortiz-Barredo, J. Echazarra, and A. Johannes, "Deep convolutional neural networks for mobile capture device-based crop disease classification in the wild," *Comput. Electron. Agric.*, vol. 151, pp. 18–26, 2018. doi: 10.1016/j.compag.2018.05.017
[11] M. A. Khan, M. I. Lali, and M. Sharif, "An optimized method for segmentation and classification of apple diseases based on strong correlation and genetic algorithm based feature selection," *IEEE Access*, vol. 7, pp. 46261–46277, 2019. doi: 10.1109/ACCESS.2019.2909320
[12] Y. Xiong, L. Liang, L. Wang, J. She, and M. Wu, "Identification of cash crop diseases using automatic image segmentation algorithm and deep learning with expanded dataset," *Comput. Electron. Agric.*, vol. 177, 105691, 2020. doi: 10.1016/j.compag.2020.105691
[13] H. Sabrol and K. Satish, "Tomato plant disease classification in digital images using classification tree," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, 2016, pp. 1242–1246. doi: 10.1109/ICCSP.2016.7754366
[14] M. Saleem, J. Potgieter, and K. M. Arif, "Plant disease detection and classification by deep learning," *Plants*, vol. 8, no. 11, p. 468, 2019. doi: 10.3390/plants8110468
[15] G. Sambasivam and G. Opiyo, "Predictive machine learning model for cassava disease detection and classification with imbalanced dataset using convolutional neural networks," *Egypt. Inform. J.*, vol. 22, no. 2, pp. 105–116, 2021. doi: 10.1016/j.eij.2020.08.005
[16] J. Amara, B. Bouazizi, and A. Algergawy, "A deep learning-based approach for banana leaf diseases classification," in *Proc. Lecture Notes Inform. (LNI)*, 2017, pp. 79–88.
[17] C. DeChant, T. Wiesner-Hanks, S. Chen, E. L. Stewart, J. Yosinski, and M. A. Gore, "Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning," *Phytopathology*, vol. 107, no. 11, pp. 1426–1432, 2017. doi: 10.1094/PHYTO-11-16-0417-R
[18] M. Brahimi, K. Boukhalfa, and A. Moussaoui, "Deep learning for tomato diseases: Classification and symptoms visualization," *Appl. Artif. Intell.*, vol. 31, no. 4, pp. 299–315, 2017. doi: 10.1080/08839514.2017.1315516
[19] A. K. Rangarajan, R. Purushothaman, and A. Ramesh, "Tomato crop disease classification using pre-trained deep learning algorithm," *Procedia Comput. Sci.*, vol. 133, pp. 1040–1047, 2018. doi: 10.1016/j.procs.2018.07.070
[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 2818–2826. doi: 10.1109/CVPR.2016.308
[21] A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, and D. P. Hughes, "Deep learning for image-based cassava disease detection," *Front. Plant Sci.*, vol. 8, p. 1852, 2017. doi: 10.3389/fpls.2017.01852
[22] J. Hang, D. Zhang, P. Chen, J. Zhang, and B. Wang, "Classification of plant leaf diseases based on improved convolutional neural network," *Sensors*, vol. 19, no. 19, p. 4161, 2019. doi: 10.3390/s19194161
[23] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Front. Plant Sci.*, vol. 7, 215232, 2016. doi: 10.3389/fpls.2016.01419
[24] X. Zhang, Y. Qiao, F. Meng, C. Fan, and M. Zhang, "Identification of maize leaf diseases using improved deep convolutional neural networks," *IEEE Access*, vol. 6, pp. 30370–30377, 2018. doi: 10.1109/ACCESS.2018.2844405
[25] M. G. Selvaraj, A. Vergara, H. Ruiz *et al*., "AI-powered banana diseases and pest detection," *Plant Methods*, vol. 15, no. 1, p. 92, 2019. doi: 10.1186/s13007-019-0475-z
[26] A. Fuentes, S. Yoon, S. C. Kim, and D. S. Park, "A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition," *Sensors*, vol. 17, no. 9, p. 2022, 2017. doi: 10.3390/s17092022
[27] M. Arsenovic, M. Karanovic, S. Sladojevic, A. Anderla, and D. Stefanovic, "Solving current limitations of deep learning based approaches for plant disease detection," *Symmetry*, vol. 11, no. 7, p. 939, 2019. doi: 10.3390/sym11070939
[28] A. G. Howard, M. Zhu, B. Chen *et al*., "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv Print, arXiv:1704.04861, 2017. doi.org/10.48550/arXiv.1704.04861
[29] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 2261–2269. doi: 10.1109/CVPR.2017.243
[30] Y. Kurmi and S. Gangwar, "A leaf image localization-based algorithm for different crops disease classification," *Inf. Process. Agric.*, vol. 9, no. 3, pp. 456–474, 2022. doi: 10.1016/j.inpa.2021.12.001
[31] J. G. A. Barbedo, "Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification," *Comput. Electron. Agric.*, vol. 153, pp. 46–53, 2018. doi: 10.1016/j.compag.2018.08.013
[32] S. Ahmed, B. Hasan, T. Ahmed, R. Karim, and H. A. Kabir, "Less is more: Lighter and faster deep neural architecture for tomato leaf disease classification," *IEEE Access*, vol. 10, pp. 68868–68884, 2022. doi: 10.1109/ACCESS.2022.3187203
[33] G. Geetharamani and A. Pandian, "Identification of plant leaf diseases using a nine-layer deep convolutional neural network," *Comput. Electr. Eng.*, vol. 76, pp. 323–338, 2019. doi: 10.1016/j.compeleceng.2019.04.011
[34] M. Agarwal, A. Singh, S. Arjaria, A. Sinha, and S. Gupta, "ToLeD: Tomato leaf disease detection using convolution neural network," *Procedia Comput. Sci.*, vol. 167, pp. 293–301, 2020. doi: 10.1016/j.procs.2020.03.225
[35] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, "Deep neural networks based recognition of plant diseases by leaf image classification," *Comput. Intell. Neurosci.*, vol. 2016, 3289801, 2016. doi: 10.1155/2016/3289801
[36] A. Sembiring, Y. Away, F. Arnia, and R. Muharar, "Development of concise convolutional neural network for tomato plant disease classification based on leaf images," *J. Phys., Conf. Ser.*, vol. 1845, no. 1, 012009, 2021. doi: 10.1088/1742-6596/1845/1/012009
[37] M. Nagaraju and P. Chawla, "Maize crop disease detection using NPNet-19 convolutional neural network," *Neural Comput. & Applic.*, vol. 35, no. 20, pp. 15109–15133, 2023. doi: 10.1007/s00521-022-07722-3
[38] F. Jakjoud, A. Hatim, and A. Bouhaddi, "Deep learning application for plant diseases detection," in *Proc. 4th Int. Conf. Comput. Sci. Appl. (ICCSA)*, 2019, pp. 1–6. doi: 10.1145/3372938.3372983
[39] Y. Liu, J. Liu, W. Cheng, Z. Chen, J. Zhou, H. Cheng, and C. Lv, "A high-precision plant disease detection method based on a dynamic pruning gate friendly to low-computing platforms," *Plants*, vol. 12, no. 11, p. 2073, 2023. doi: 10.3390/plants12112073
[40] J. N. Canlas, C. M. M. Cortez, R. A. M. Dela Cruz, and J. R. G. Padua, "Camera radial distance-based accuracy of a bacterial blight, brown spot, and rice blast plant disease identification system for remote communications deployment," *J. Image Graphics*, vol. 10, no. 4, pp. 170–175, 2022. doi: 10.18178/joig.10.4.170-175
[41] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Comput. Electron. Agric.*, vol. 145, pp. 311–318, 2018. doi: 10.1016/j.compag.2018.01.009