Applied Soft Computing Journal 86 (2020) 105933

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

# Attention embedded residual CNN for disease detection in tomato leaves

Karthik R. [a], Hariharan M. [b], Sundar Anand [a], Priyanka Mathikshara [a], Annie Johnson [a], Menaka R. [a,*]

[a] School of Electronics Engineering, Vellore Institute of Technology, Chennai, India
[b] School of Computing sciences and Engineering, Vellore Institute of Technology, Chennai, India

ARTICLE INFO

*Article history:*
Received 29 April 2019
Received in revised form 8 October 2019
Accepted 6 November 2019
Available online 12 November 2019

*Keywords:*
Attention
CNN
Residual connections
Tomato
Deep learning

ABSTRACT

Automation in plant disease detection and diagnosis is one of the challenging research areas that has gained significant attention in the agricultural sector. Traditional disease detection methods rely on extracting handcrafted features from the acquired images to identify the type of infection. Also, the performance of these works solely depends on the nature of the handcrafted features selected. This can be addressed by learning the features automatically with the help of Convolutional Neural Networks (CNN). This research presents two different deep architectures for detecting the type of infection in tomato leaves. The first architecture applies residual learning to learn significant features for classification. The second architecture applies attention mechanism on top of the residual deep network. Experiments were conducted using Plant Village Dataset comprising of three diseases namely early blight, late blight, and leaf mold. The proposed work exploited the features learned by the CNN at various processing hierarchy using the attention mechanism and achieved an overall accuracy of 98% on the validation sets in the 5-fold cross-validation.



## 1. Introduction

Tomato holds an inevitable place in the economy of Indian agriculture. India stands third in the production of tomatoes with a yield of 53,00,000 tons and it is harvested around 3,50,000 hectares of land. The harvest index of tomato in India is comparatively less than in other countries. One of the major reasons for the reduction in yield is due to diseases that occur frequently on the leaves of the plant. Tomato crops are highly affected by diseases like bacterial spot, early blight, late blight, and leaf mold. The blight is the most prevalent disease among others.

The tomato crop is highly susceptible to a wide range of disease at each stage of its growth. This is due to different factors based on climatic conditions and environmental parameters. By identifying these diseases, tremendous loss in the yield can be alleviated. Also, the final agricultural product obtained in terms of quality and quantity can be improved. It is relatively complex in real time to maintain a manual record of all the symptoms and signs caused by the diseases. Also, monitoring of plants in a large field requires extensive manual effort. Hence, different automation schemes for disease detection were presented in the last two decades [1–5].

As a part of sustainable agriculture, various measures can be taken by leveraging the technology towards automated inspection of diseases. Factors like pathogen development, modification of host resistance and wider global transfer of diseases have led to the development of many solutions [6]. Precision farming was aimed at limiting the employment of expensive methods of farming which uses harmful chemicals. In this type of farming, mobile robotics, remote sensor networks and drones are used to advocate controlled and measured amounts of medicine to the infected areas of the plants.

The main challenge of precision farming is that, it had numerous challenges in data collection, processing and make expert inferences. Therefore, precision farming incorporated image processing and computer vision techniques to process the information in the cultivation field. Image processing was quite successful in solving the problems of disease detection, weed detection, understanding the symptoms of a disease and even more recently, grading the yield output. As machine learning algorithms continued to advance, the accuracy in image processing techniques continued to grow consistently. However, these algorithms demand handpicked features to detect diseases which made deep learning techniques pertinent. This research places an attempt towards application of deep learning architectures for detection of diseases in tomato leaves.

* Corresponding author.
*E-mail address:* menaka.r@vit.ac.in (Menaka R.).

https://doi.org/10.1016/j.asoc.2019.105933
1568-4946/

## 2. Related works

Several research works have been presented in the last two decades towards detection of disease in different crops. Image processing techniques were applied to extract the features and given as input to machine learning algorithms for precise classification. In short, these approaches can be broadly classified into (1) Machine learning methods (2) Deep learning methods.

### 2.1. Machine learning based methods

Akhtar et al. presented an automated approach for plant disease detection using Gray Level Co-occurrence Matrix (GLCM) and Wavelet-based features [7]. The features were trained with different machine learning algorithms namely K- Nearest Neighbor (KNN), Naïve Bayes Classifier, Support Vector Machine (SVM), Decision Tree and Recurrent Neural Networks. An automated approach for tomato grading system was presented by Semary et al. [8]. This approach utilized color and texture features and classified using SVM. Prasad et al. developed an automated approach for leaf disease diagnosis using Gabor Wavelet Features (GWF) and GLCM features. These multi-resolution features were trained using weighted KNN [9].

Ashourloo et al. presented a method to detect leaf disease using hyperspectral measurement [10]. An approach to detect the severity of the disease in leaves was proposed in [11]. Statistical features in the RGB and HSV color space were utilized for determining the severity level. H. Sabrol et al. presented an approach for leaf disease detection in tomatoes by combining Otsu's segmentation with decision trees for classification. This approach considered color, shape and texture features for learning the characteristics of the leaf diseases [12].

Padol et al. presented an approach to detect leaf diseases using color and texture features. The infected region was initially segmented using K-means clustering. Then, features were extracted from the required region of interest and trained using SVM for classification [13]. Another approach using K-means algorithm was proposed for leaf disease detection and classification [14]. T. Mehra et al. employed K-means clustering to identify the presence of fungal infections on leaves [15]. One of the major challenge in applying the above clustering algorithms is the determination of precise number of clusters and fixing of parameters to differentiate each cluster.

In the past few years, Scale invariant feature transforms were explored for many image processing problems [16–18]. An approach using Scale Invariant Feature Transform (SIFT) for detection of leaf disease was presented by Dandawate et al. [19]. In this work, SIFT features were trained using SVM for detecting the presence of disease. SIFT based features were combined with Johnson SB distribution for effective classification of diseases in tomatoes [20].

All the above methods for disease detection were based on hand engineered features extracted from the leaf portion of the image. The accuracy of these works solely depends on the nature of the handcrafted features selected. Also, it is to be noted that the performance of these works needs to be validated against a wide range of datasets. These drawbacks can be addressed by using deep learning techniques.

### 2.2. Deep learning based methods

Unlike machine learning algorithms, deep learning algorithms can be applied directly over the input data and does not require any handcrafted features. In today's world, the computing power delivered by High-Performance Computing (HPC) and Graphics Processing Unit (GPU) allows for efficient training of deep models while simultaneously implementing parallelism in computing. A number of deep learning models have been proposed in order to train leaf images to perform disease detection.

Most of the researches were based on applying existing deep learning architectures like VGG16, AlexNet, ResNet, GoogleNet etc. for detection of infection in tomato leaves. Jia Shijie et al. presented an approach to detect diseases in tomato leaves using VGG16 architecture [21]. Suryawati et al. presented another deep CNN using VGG16 architecture to detect infection in tomato leaves [22]. Aravind et al. compared the performance of VGG16 with AlexNet architecture for disease detection in tomato leaves. It was inferred that the model trained with AlexNet architecture was accurate than VGG16 by a small margin [23]. Jayme Garcia et al. utilized a pre-trained GoogleNet CNN architecture for disease detection in leaves [24]. Zhang et al. presented a transfer learning approach using AlexNet, GoogleNet, and ResNet architectures for disease detection in tomato leaves [25]. Liang et al. presented an approach involving the use of Resnet50, Wideresnet50, DPN92 neural networks for classification of plant diseases [26]. A deep architecture based on LeNet was proposed to detect the type of disease in tomato leaves in [27].

Q. H. Cap et al. used two super resolution (SR) models which are based on super resolution convolutional neural networks (SRCNN) and enhanced super resolution generative adversarial networks (ESRGAN). The SRCNN model is used to identify prominent disease features, whereas, the ESRGAN model focuses on high frequency details to obtain a more accurate prediction [28]. Another deep learning architecture named 'PD2SENET' was proposed to detect and indicate the severity of the disease [29]. In this architecture, the shallow layers considered raw pixel values of plant images as input and the progressive feature maps are generated with the help of residual learning. Srdjan Sladojevic et al. presented a CaffeNet based architecture for detection of leaf diseases. This architecture had eight layers for learning the characteristics of the disease patterns and utilized around 30 K samples for training the model [30]. Alvaro Fuentes et al. presented deep learning meta architectures for disease detection by combining Faster Region-based Convolutional Neural Network (Faster R-CNN), Region-based Fully Convolutional Network (R-FCN), and Single Shot Multibox Detector (SSD) with ResNet and VGG architectures. It was inferred that, R-FCN with ResNet combination outperformed the other two methods [31].

In addition to the application of existing CNN architectures, several custom architectures were proposed for disease detection in tomato leaves. Ferdouse et al. presented one such CNN to identify diseases in tomato leaves [32]. This architecture consists of 15 layers to extract a wide range of features for classification. Ruedeeniraman et al. presented a VegeCare tool that made use of Deep Neural Network (DNN) to classify six tomato diseases [33]. Fuentes et al. presented another deep architecture to identify diseases in tomatoes [34]. Melike Sardogan et al. presented a CNN model to identify the type of disease in tomato plant [35]. This method considered only 400 images for training, which is relatively less for a deep learning model. Pardede et al. presented an unsupervised convolutional auto-encoder for automatic detection of plant diseases [36].

In contrast to the above standalone deep learning applications, few CNN models were also presented as mobile applications focusing on disease detection in tomato leaves. A. Elhassouny et al. presented a MobileNet CNN model that involves depth-wise separable convolution operations to address computational burden of the traditional CNN for real time applications [37]. Another mobile application was developed by H. Durmus et al. which used SqueezeNet for classification of tomato leaf diseases [38].

### 2.3. Research gaps and motivation

Though several approaches were presented for detection of diseases in tomato leaves, there exist some significant challenges in it.

1. It is quite complex to identify and extract significant features in tomato leaves to differentiate the properties of different diseases using traditional image processing techniques. As the characteristics of these diseases exhibit huge variation, the properties of the disease patterns have to be studied exhaustively with a wide range of datasets in an automated way.
2. The performance of the machine learning based models solely depends on the nature of the manually selected handcrafted features. Hence, feature extraction has to be made automatic to select and learn an optimal set of features for classification purpose.
3. Most of the deep learning models give equal weightage to all features derived across different levels. But to make the model more sensitive for classification, feature weighting has to be done at each stage. By doing so, significant features can be learnt and passed to deeper levels of the network for precise classification.
4. Some of the deep learning models utilize generic and proven architectures like VGG16, GoogleNet etc. Hence, it utilizes millions of parameters for classification. For real-time deployment of such models, a trade-off has to be achieved between the computational burden and accuracy.
5. Also, the deep learning network has to be trained with a large collection of samples to ensure better generalization of features.

### 2.4. Research contributions

To address the above research gaps, the proposed research employed two different deep architectures for disease detection in tomato leaves. The following are the major contributions of the proposed work.

1. Two different deep learning architectures were proposed in this research. The first architecture employed residual learning to learn a hierarchy of features for better classification. The second architecture employed the attention mechanism to specifically learn distinctive feature maps and improve the performance of the residual CNN.
2. To the best of our knowledge, this is the first attempt to develop attention based residual deep network for disease detection in tomato leaves. Attention mechanism is employed to learn and weight significant features across different levels. Hence, the significant features were given more weightage with the help of attention coefficients learnt and passed to deeper levels for precise classification.
3. The proposed architecture was trained with a large collection of samples. 95999 images were used for training the model and 24001 images were used for validation purpose.

## 3. Proposed work

This research proposes a novel CNN framework that specializes in the task of infestation detection in the tomato plant. The objective of this work is to design a computationally inexpensive and accurate learning model for disease detection. Two different deep architectures were proposed in this work, to detect disease infestation in tomato leaf. The first architecture integrates residual learning on top of a feed-forward CNN. The second architecture integrates the strengths of Attention mechanism and Residual Learning on CNN.

### 3.1. Residual learning based CNN

The learning pattern of a CNN is generally based on aggregation of feature maps derived at multiple levels. As a consequence of this aggregation occurring in the deep layers of the CNN, it tends to lose the significance of the fine granular details learnt by the initial layers. The traditional CNN based methods for tomato leaf infestation detection focusses on learning the features in an orderly fashion starting from basic image level features like edges and move towards complex texture based differences. By doing so, few significant details are not passed to the deeper layers of the network. Hence in this method, residual connections are employed to pass those significant features extracted in the initial layers to the deeper layers of the network. This supports effective aggregation of feature maps for precise classification.

The architecture of the proposed residual connection based on CNN is presented in Fig. 1. It consists of a sequence of three Residual Progressive Feature Extraction (RPFE) blocks, each set to learn progressive features. The number of channels increases from 32 to 128 along the depth of the network. The first RPFE block has a convolution receptive area of size $7 \times 7$, trailed by a $5 \times 5$ kernel for the second block and finally a $3 \times 3$ filter for the third block. Then, it applies the average pooling over the feature map. This enables the classifier to model a reduced set of features without much loss of context and also avoids the risk of overfitting. The entire model is followed by a sequence of $1 \times 1$ convolutional layers after the last RPFE block.

#### 3.1.1. Residual Progressive Feature Extraction (RPFE) block

The Residual Progressive Feature Extraction (RPFE) block consists of a 2D convolutional layer, a max pooling layer, and a batch norm layer. In the 'Conv' layer, a set of variable filters distinctly convolve across the face of the feature map (padded to the same size), one for each channel. The filter sizes for the 'Conv' layer are receptive to a smaller region of interest along the line of the blocks ($7 \times 7$ for the first block, $5 \times 5$ for the second, $3 \times 3$ for the third and so on). This is followed by a Rectified Linear Unit (ReLU) activation layer that rectifies the convolved image, zeroing out negative values. ReLU was used because it sustains a steady gradient even for larger activations, thus stabilizing the learning.

The output from the convolutional layers, '$x$' in the first and second RPFE blocks is directed in two functional paths, $F(x)$ and $G(x)$. $F(x)$ denotes the set of operations (max pooling and batch normalization) that were applied to take '$x$', in a simple feed-forward manner to the next block. G(x) indicates the set of operations that skip '$x$' to the next block using convolutional and max-pooling layers. Finally, the response from the RPFE blocks, $Y(x)$ are generated by summing the individual responses of $F(x)$ and $G(x)$, as given by Eq. (1).

$$Y(x) = F(x) + G(x) \tag{1}$$

Fig. 2 presents the visual representation of the skip functions, generically to an RPFE block in the described Residual CNN.

The proposed network was designed end-to-end only with 2D convolution, pooling, batch norm layers, with no dense layers. The observed bottleneck in the case of the last layers being fully connected is that the model fails to exploit the run entirely in the GPU. With full convolution, the system can now generate a spatial map whose correspondences can be tracked to different parts of the input image. This essentially translates to sliding a classifier over the input image, making predictions at each window, regardless of the input size. This approach towards identifying the infection makes it possible to,

(a) share parameters (significantly at the initial few layers)
(b) exploit spatial locality (when used with a stride less than the filter size)

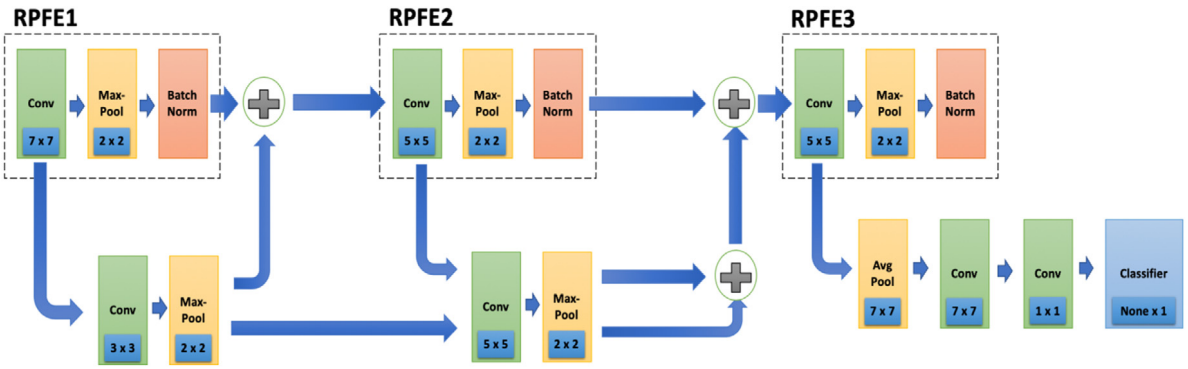



**Fig. 1.** The architecture of the proposed residual CNN.

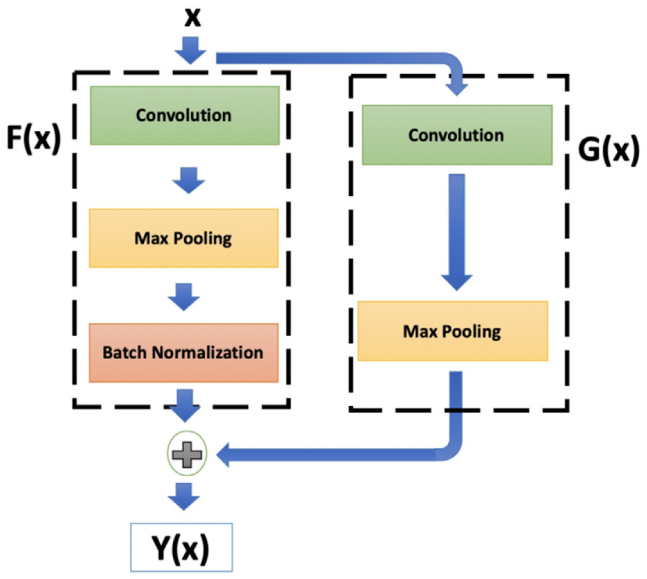



**Fig. 2.** A representation of the functions F(x) and G(x) that are skipped through the RPFE block to generate the output Y(x) from the residual block.

(c) feature the different parts of the image (matched with the receptive fields of the input layers' filters down to the deep layers) all in one-shot.

### *Convolutional Layer*

The convolutional layer defines a set of filters that perform the convolution operation over the entire image. involve a series of convolution operation among an input volume 'I' and a set of 'n' convolutional filters '$F^E$' followed by a non-linear activation. This finally yields an output volume 'O' as presented in Eq. (2).

$$O_m(i,j) = a\left(\left(\sum_{d=1}^{D}\sum_{u=-2k-1}^{2k+1}\sum_{v=-2k-1}^{2k+1} F^E_{md}(u,v)\, I_d(i-u, j-v)\right) + b_m\right) \quad (2)$$

where,

'2k+1' is the side of a square with odd convolutional filter

'a' refers to the activation function

'$b_m$' refers to the bias for the $m^{th}$ feature map

The activation maps produced with the help of above relation are the encoding of the input 'I' in a low dimensional space i.e. it refers to the parameters used to build every feature map '$O_m$'. After '$O_m$' is calculated, it is subjected to a max-pooling operation to down-sample it. Intuitively, each convolutional layer in this architecture learns the various attributes that capture discriminatory patterns to differentiate the type of infection in the tomato leaf.

### *ReLU Activation*

The Rectified Linear Unit (ReLU) is an activation function adopted in the design of most neural networks, particularly CNN's. It is the identity function, $f(x) = x$, for all positive values and zeros out for negative values of input '$x$'. ReLU is sparsely activated, which helps to mimic the inactivity of the biological neuron to certain impulses.

### *Max Pooling Layer*

This pooling layer maximally activates only a bunch of neurons from the feature map. It is used with a stride factor of '*2*' on a '*2-by-2*' window, across all the RPFE blocks. This effectively reduces the width and height of the feature maps while preserving the number of channels.

### *Batch Normalization Layer*

In Deep Neural Networks each layer sees different feature information from the previous layer after every single gradient update on a batch of data. And the data distribution of this input feature map largely varies, as the parameter of the previous layers is updated during the training phase. This significantly affects the training pace and also calls for various heuristics to decide upon the parameter initialization. Batch Normalization is a popular trick used to curtail this problem of Internal Covariate Shift and the outputs of the BN layer for a batch '$x$' is given by Eq. (3).

$$y = \frac{x - m}{\sqrt{s^2 + \varepsilon}}\beta + \varphi \quad (3)$$

where 'm' and 's' are respectively the mean and standard deviation of the batch 'x'. '$\beta$', '$\varphi$' are trainable parameters, that are updated at each iteration. '$\varepsilon$' is set to a small constant, introduced to increase the variance, as well as prevent the denominator from zeroing out. Batch Normalization overcomes the vanishing/exploding gradient problem by normalizing the values to a range between −3 to 3, fitting a maximum likelihood estimate (along with the line of channel activations, across a batch) for normal distribution.

The details of the tensor at each layer of this architecture are tabulated in Table 1.

## *3.2. Attention-based residual CNN*

The attention model works on top of the RPFE CNN by retaining the context relevant features. The previous RPFE based model

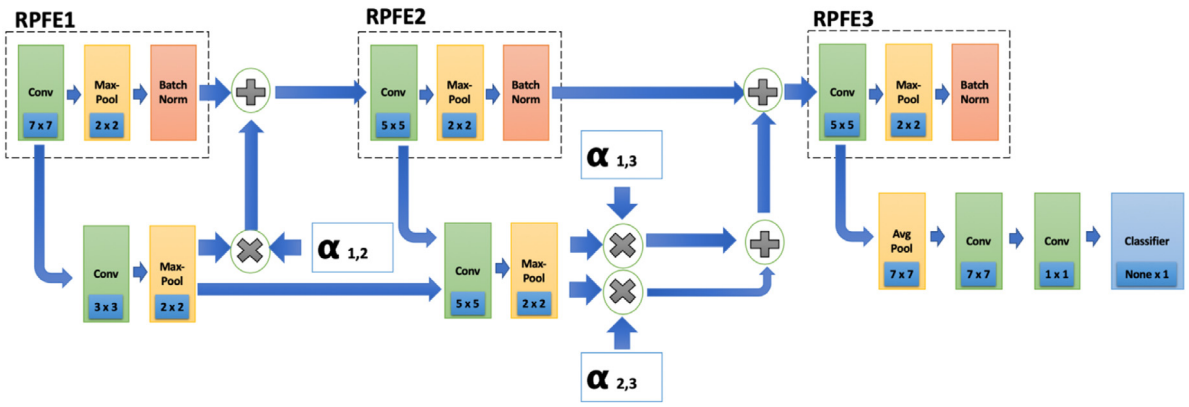



**Fig. 3.** An overview of the architecture employed to integrate attention within the residual net framework.

**Table 1**
A tabulation of the connections between the layers and the dimensions of the output tensors at each layer, for the entire Residual CNN.

| Layer (type) | Output shape | No. of parameters | Connected to the previous layer |
|---|---|---|---|
| input_1 (InputLayer) | (None, 256, 256, 3) | 0 | |
| conv2d_1 (Conv2D) | (None, 256, 256, 32) | 4736 | input_1 (0,0) |
| conv2d_2 (Conv2D) | (None, 128, 128, 32) | 9248 | conv2d_1(0,0) |
| max_pooling2d_1 (MaxPooling2D) | (None, 128, 128, 32) | 0 | conv2d_1(0,0) |
| max_pooling2d_2 (MaxPooling2D) | (None, 128, 128, 32) | 0 | conv2d_39(0,0) |
| batch_normalization_1 (BatchNorm) | (None, 128, 128, 32) | 128 | max_pooling2d_1(0,0) |
| add_1 (Add) | (None, 128, 128, 32) | 0 | max_pooling2d_2(0,0), batch_normalization_1(0,0) |
| conv2d_3 (Conv2D) | (None, 64, 64, 64) | 51264 | add_1(0,0) |
| conv2d_4 (Conv2D) | (None, 64, 64, 64) | 36928 | conv2d_1(0,0) |
| conv2d_5 (Conv2D) | (None, 64, 64, 64) | 18496 | max_pooling2d_2(0,0) |
| max_pooling2d_3 (MaxPooling2D) | (None, 32, 32, 64) | 0 | conv2d_3(0,0) |
| max_pooling2d_4 (MaxPooling2D) | (None, 32, 32, 64) | 0 | conv2d_4(0,0) |
| max_pooling2d_5 (MaxPooling2D) | (None, 32, 32, 64) | 0 | conv2d_5(0,0) |
| batch_normalization_2 (BatchNorm) | (None, 32, 32, 64) | 256 | max_pooling2d_3(0,0) |
| add_2 (Add) | (None, 32, 32, 64) | 0 | max_pooling2d_4(0,0), max_pooling2d_5(0,0), batch_normalization_2(0,0) |
| conv2d_6 (Conv2D) | (None, 32, 32, 128) | 204928 | add_2(0,0) |
| max_pooling2d_6 (MaxPooling2D) | (None, 16, 16, 128) | 0 | conv2d_6(0,0) |
| batch_normalization_3 (BatchNorm) | (None, 16, 16, 128) | 512 | max_pooling2d_6(0,0) |
| average_pooling2d_1(AveragePooling2D) | (None, 8, 8, 128) | 0 | batch_normalization_3(0,0) |
| conv2d_7 (Conv2D) | (None, 1, 1, 64) | 401472 | average_pooling2d_1(0,0) |
| lambda_1 (Lambda) | (None, 1, 1, 64) | 0 | conv2d_7(0,0) |
| conv2d_8 (Conv2D) | (None, 1, 1, 4) | 260 | lambda_1(0,0) |
| reshape_1 (Reshape) | (None, 4) | 0 | conv2d_8(0,0) |

combines the features extracted in each block with the features derived from its preceding layer. In this way, equal importance is given to all features collected from the earlier RPFE blocks. For precise feature learning, significant features from the previous blocks need to be weighted high relative to other features. Hence, an attention mechanism was introduced on top of the RPFE architecture to learn and select prominent features from the previous RPFE blocks. This model learns an attention mask that weighs the relative importance of spatial features at that feature map. This way it learns attention coefficients for each pixel in the feature map to understand the properties of the infestation in an effective manner. The architecture of the proposed attention based on CNN, built on top of the described residual architecture in Section 2.1. is presented in Fig. 3.

*3.2.1. Attention embedded Residual Progressive Feature Extraction (ARPFE) block*

This architecture uses the attention mechanism across blocks to learn a weighted function for modeling the activations from the preceding blocks. The skip connections from the previous blocks are now weighted across the depth axis for each pixel in the spatial expanse of that layer.

The output from the convolutional layers, '*x*' in the first and second ARPFE blocks is directed in two functional paths, *F(x)* and *G'(x)*. *F(x)* denotes the set of operations (max pooling and batch normalization) that were applied to take '*x*', in a simple feed-forward manner to the next block. G'(x) indicates the attention-aided weighted set of operations that skip '*x*' through convolutional and max-pooling layers. As discussed in 3.1.1 the weighted summation is used to generate the output Y(x) from an

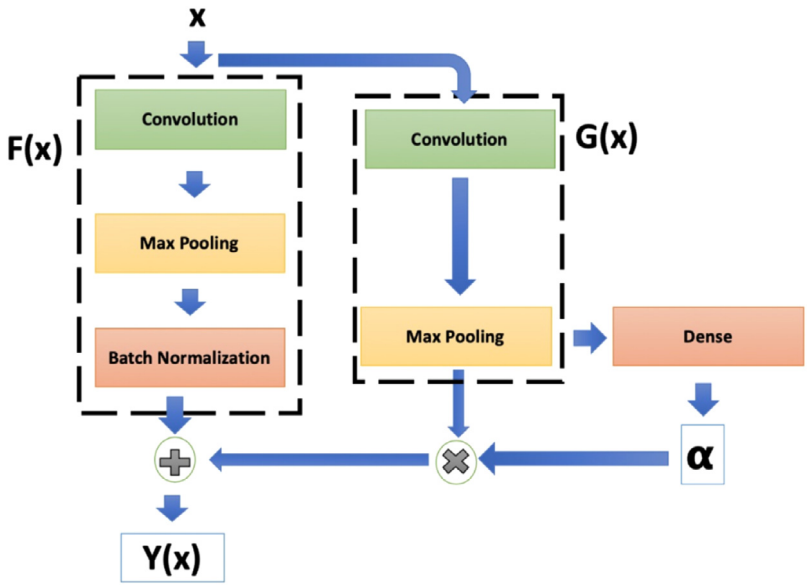



**Fig. 4.** A visual representation for generating the output Y(x) from an ARPFE block using attention based weights.

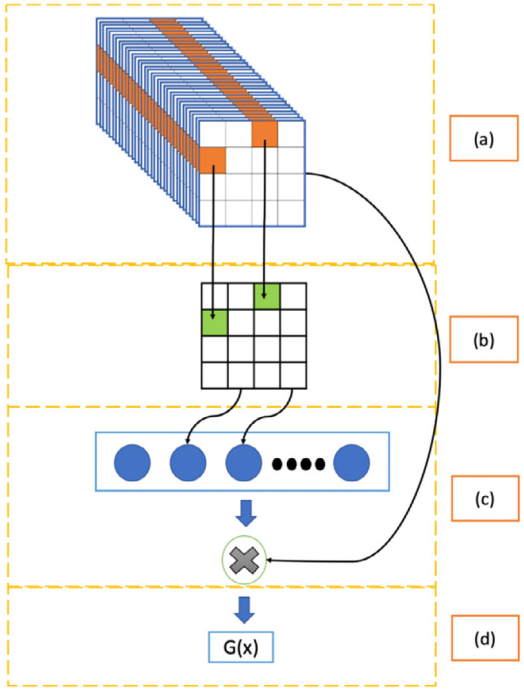



**Fig. 5.** A generic scheme for learning the attention function $\alpha_{ij}$. (a): Feature map G(x) (b): A ReLU-activated dense layer matrix with one unit for each cross-section (i,j) of $G(x)_{i,j}$. (c): A softmax activation layer following the dense layer from (b) to learn a probability distribution of weights for each $\alpha_{ij}$. The weighted sum of $\alpha$ s over G(x) yields G'(x) (d): Output G'(x) is computed as $\sum_{ij} \alpha_{ij} * G(x)_{ij}$.

ARPFE block, given by Eq. (4).

$$Y(x) = F(x) + G'(x) \tag{4}$$

The functional path G'(x) is computed as

$$G'(x) = G(x) * \alpha \tag{5}$$

where '$\alpha$' is the attention weight matrix whose dimensions are the same as the spatial dimensions of $G(x)$. The attention weight matrix '$\alpha$' is point-wise multiplied (broadcasted along the depth) across the corresponding cross-section of $G(x)$. So, '$\alpha$' is weighted function in $G(x)$ and $G'(x)$ is derived from '$\alpha$' as given by Eq. (5). This process is illustrated in Fig. 4.

The method for learning these weights is shown in Fig. 5 The residual feature map $G(x)$ is passed through a dense layer (with ReLU activation) that learns a parameter '$\alpha_{ij}$' for each pixel cross-section volume $G(x)_{(i,j)}$.

The dense matrix is flattened out, to form a feature vector. The activation values from the feature vector are passed through a Softmax layer. The weights '$\alpha_{ij}$' are now computed calculated as a Softmax probability distribution, such that summation of $\sum_{ij} \alpha_{ij} = 1$.

### Softmax Classifier

The proposed system uses a $k$-way softmax classifier to make classify the image to one among $k$ categories. This loss is given by Eq. (6).

$$CE = -\sum_{i}^{k} t_i log\ (f\ (s)_i) \tag{6}$$

where the $f(s)_i$ is the output conditional probability $P(\ y = \hat{y}_i |\ s_i\ )$ for some training example '$s_i$', predicted value $\hat{y}_i$. This probability function for softmax activation is given in Eq. (7).

$$f\ (s)_i = \frac{e^{s_i}}{\sum_j^k e^{s_i}} \tag{7}$$

The details of the tensor at each layer of this architecture are tabulated in Table 2.

## 4. Results and discussion

The proposed system was trained with the augmented collection of the benchmarked Plant Village Dataset. The source code was written in Tensorflow Deep Learning programming framework and compiled to run on the NVIDIA Tesla P100 GPU. The model was evaluated on a 5-fold cross validation set (of 120 K samples) with each fold stratified into roughly equal numbers for each class (by random sampling with replacement). The loss function was minimized using the Adaptive Moment Estimation (Adam) optimizer. This optimization algorithm uses the running average of both the gradient and the second moment.

Three different experiments were conducted. The first experiment applied a baseline model for disease detection and

**Table 2**
A tabulation of the connections between the layers and the dimensions of the output tensors at each layer, for the entire Residual CNN.

| Layer (type) | Output shape | No. of parameters | Connected to the previous layer |
|---|---|---|---|
| input_1 (InputLayer) | (None, 256, 256, 3) | 0 | |
| conv2d_1 (Conv2D) | (None, 256, 256, 32) | 4736 | input_1(0,0) |
| conv2d_2 (Conv2D) | (None, 128, 128, 32) | 9248 | conv2d_1(0,0) |
| max_pooling2d_1 (MaxPooling2D) | (None, 128, 128, 32) | 0 | conv2d_2(0,0) |
| conv2d_3 (Conv2D) | (None, 64, 64, 64) | 18496 | max_pooling2d_1(0,0) |
| max_pooling2d_2 (MaxPooling2D) | (None, 32, 32, 64) | 0 | conv2d_3(0,0) |
| dense_1 (Dense) | multiple | 65 | max_pooling2d_2(0,0), max_pooling2d_4(0,0) |
| dense_2 (Dense) | (None, 128, 128, 1) | 33 | max_pooling2d_1(0,0) |
| attention_weights (Activation) | multiple | 0 | dense_2(0,0), dense_1(0,0), dense_1(0,0) |
| max_pooling2d_3 (MaxPooling2D) | (None, 128, 128, 32) | 0 | conv2d_1(0,0) |
| multiply_1 (Multiply) | multiple | 0 | attention_weights(0,0), max_pooling2d_1(0,0), attention_weights(0,0), max_pooling2d_2(0,0), attention_weights(0,0), max_pooling2d_4(0,0) |
| batch_normalization_1 (BatchNorm) | (None, 128, 128, 32) | 128 | max_pooling2d_3(0,0) |
| add_1 (Add) | (None, 128, 128, 32) | 0 | multiply_1(0,0), batch_normalization_1(0,0) |
| conv2d_4 (Conv2D) | (None, 64, 64, 64) | 51264 | add_1(0,0) |
| conv2d_5 (Conv2D) | (None, 64, 64, 64) | 36928 | conv2d_4(0,0) |
| max_pooling2d_4 (MaxPooling2D) | (None, 32, 32, 64) | 0 | conv2d_5(0,0) |
| max_pooling2d_5 (MaxPooling2D) | (None, 32, 32, 64) | 0 | conv2d_4(0,0) |
| batch_normalization_2 (BatchNorm) | (None, 32, 32, 64) | 256 | max_pooling2d_5(0,0) |
| add_2 (Add) | (None, 32, 32, 64) | 0 | multiply_1(0,0), multiply_1(0,0), batch_normalization_2(0,0) |
| conv2d_6 (Conv2D) | (None, 32, 32, 128) | 204928 | add_2(0,0) |
| max_pooling2d_6 (MaxPooling2D) | (None, 16, 16, 128) | 0 | conv2d_6(0,0) |
| batch_normalization_3 (BatchNorm) | (None, 16, 16, 128) | 512 | max_pooling2d_6(0,0) |
| average_pooling2d_1 (AveragePooling) | (None, 8, 8, 128) | 0 | batch_normalization_3(0,0) |
| conv2d_7 (Conv2D) | (None, 1, 1, 64) | 401472 | average_pooling2d_1(0,0) |
| lambda_1 (Lambda) | (None, 1, 1, 64) | 0 | conv2d_7(0,0) |
| conv2d_8 (Conv2D) | (None, 1, 1, 4) | 260 | lambda_1(0,0) |
| reshape_1 (Reshape) | (None, 4) | 0 | conv2d_8(0,0) |

classification in tomato. The second experiment used the residual connections across the Progressive Feature Extraction blocks. The third experiment integrated both attention and residual connections in CNN.

### 4.1. Dataset

The proposed model for disease detection in Tomato was developed using the Plant Village Disease Classification Challenge dataset and further data augmentation techniques were applied to increase the size of the dataset. Table 3 presents the distribution of augmented samples for each fold in cross-validation process. The dataset used in our experiment includes one healthy class and 3 diseased classes. Table 4 shows the samples for each disease class and the effects of the data augmentation techniques on them.

Data augmentation techniques have been applied for increasing the data set, thereby reducing the overfitting. Central zoom was performed to produce a data set of images that have only the leaf and not the background information, random crop & zoom was performed to focus on specific parts of the leaf and various contrast levels were used to make the dataset robust to various lighting conditions. The stratified 5-fold cross-validation used to evaluate the proposed model and this ensured balance between the classes for each of the 5 folds due to random sampling.

### 4.2. Experiment 1: Application of the baseline model

The baseline model was a simple feed-forward CNN with no cross-connections or any learning aided mechanisms. It took around 1.5 days for training the model on the GPU. This resulted in an accuracy of 84%.

### 4.3. Experiment 2: Application of residual CNN

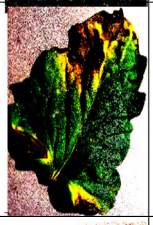
Building on experiment 1, the residual model includes skip connections from one block to the other. The skip connections take the feature map from the ReLU activated convolutional layer in RPFE block *b*, onto the convolutional layer in the RPFE block *b+1*, as described in Section 2.1. The dimensions at the both ends of the skip connection are matched by filtering the admitted feature maps with convolutional layers and trimming with max pooling.

It took around 10 h for training the model and it took approx. 150 epochs to reach convergence. The proposed residual based

**Table 3**
Details of distribution of samples in cross-validation process.

| Fold | Healthy | | Early blight | | Late blight | | Leaf mold | |
|---|---|---|---|---|---|---|---|---|
| | Training | Validation | Training | Validation | Training | Validation | Training | Validation |
| 1 | 50402 | 12601 | 20590 | 5148 | 17600 | 4400 | 7407 | 1852 |
| 2 | 50402 | 12601 | 20590 | 5148 | 17600 | 4400 | 7407 | 1852 |
| 3 | 50402 | 12601 | 20590 | 5148 | 17600 | 4400 | 7407 | 1852 |
| 4 | 50403 | 12600 | 20591 | 5147 | 17600 | 4400 | 7407 | 1852 |
| 5 | 50403 | 12600 | 20591 | 5147 | 17600 | 4400 | 7408 | 1851 |

**Table 4**
Sample results of data augmentation process.

| Category | Original Image | Contrast | Random Zoom and Crop | Central Zoom |
|---|---|---|---|---|
| Healthy |  |  |  |  |
| Early Blight |  |  |  |  |
| Late Blight |  |  |  |  |
| Leaf Mould |  |  |  |  |

network is subjected to 5 – fold cross-validation process and the resultant observations are presented in Table 5.

It could be observed that the Residual CNN was able to detect the type of disease in tomatoes with an accuracy of 90%–95%. Also, the loss of the network decays appreciably during the training phase, leading to precise classification.

### 4.4. Experiment 3: Application of residual CNN with attention

Building on the second experiment, the attention based residual model adds on a weighing scheme to the output feature map $G(x)$ from the skip connections. The weighing scheme computes an attention matrix '$\alpha'_{ij}$ is point-wise multiplied (broadcasted along the depth) across the corresponding cross-section of $G(x)_{ij}$as described in Section 2.2. These attention weights '$\alpha'_{ij}$ are learnt dynamically upon seeing new training batches.

It took around 10 h for training the model and it took approx. 150 epochs to reach convergence. The proposed residual based network is subjected to 5-fold cross-validation process and the resultant observations are presented in Table 6.

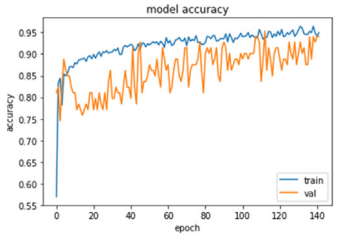
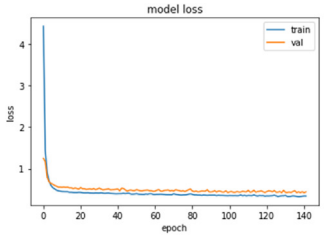
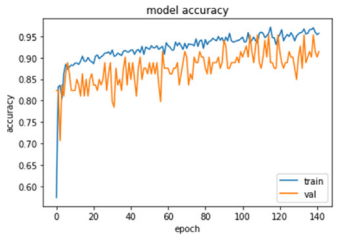
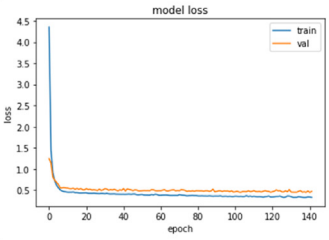
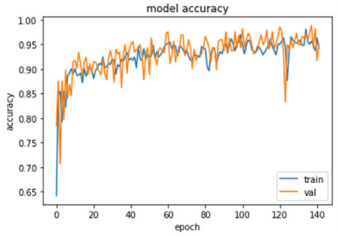
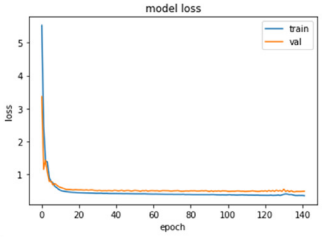
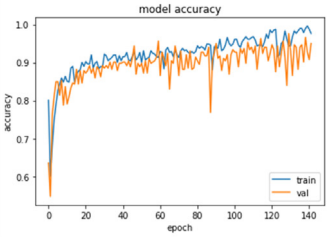
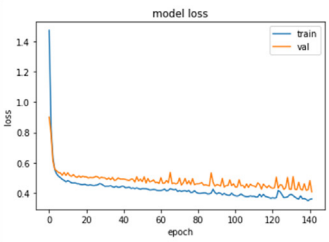
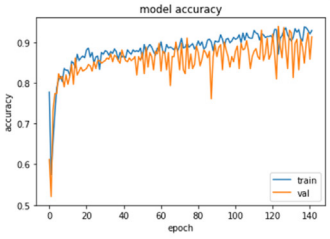
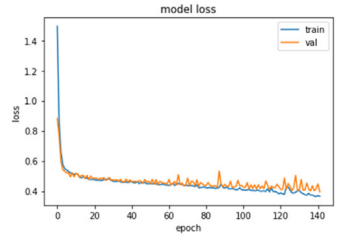
It was evident that the proposed attention based residual CNN was able to converge better than residual CNN.

### 4.5. Performance analysis

In this research, three different deep architectures were analyzed to detect the performance of disease detection. The first method applied a baseline model for disease detection in tomato leaves. The second approach was based on the residual connections across the Progressive Feature Extraction blocks. The third approach integrated both attention mechanism and residual connections in CNN. The observations of these experiments are tabulated in Table 7. It could be observed that the proposed attention based residual CNN performed better in detecting the type of infection with an accuracy of 98%.

The performance of the proposed attention based residual CNN is compared against the existing methods reported in the literature and the resultant observations sorted according to accuracy obtained are highlighted in Table 8.

**Table 5**
Observation of the proposed Residual CNN.

| Folds | Accuracy | Loss |
|---|---|---|
| Fold 1 |   |   |
| Fold 2 |   |   |
| Fold 3 |   |   |
| Fold 4 |   |   |
| Fold 5 |   |   |

It could be observed that, an accuracy of 83 to 97% was obtained for machine learning methods that employed hand crafted features for disease detection [8,12,13,20]. Also, the model was trained with less than 4k samples, which is very less to generalize all feature patterns. Recent deep learning researches employed well trained architectures like VGG16, ResNet, GoogleNet etc. for disease detection in tomato leaves [21,22,24,25,27]. In addition to these works, certain deep-layered CNN architectures were also proposed for infestation detection in tomato leaves [30,34,35,37]. Though these works yield appreciable results, the accuracy of these works were in the range of 76 to 97%. As the proposed model employed attention mechanism to learn and weight significant features, it was able to achieve an accuracy of 98%, which is a significant improvement when compared to other works.

**Table 6**
Observation of the proposed Attention based Residual CNN.

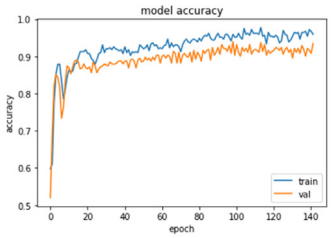
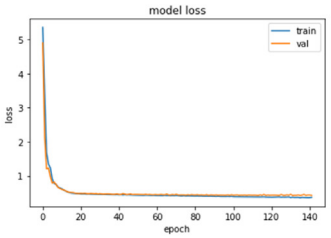
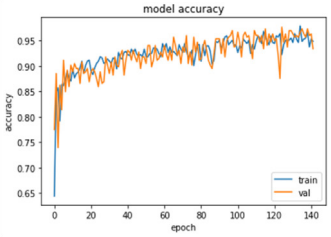
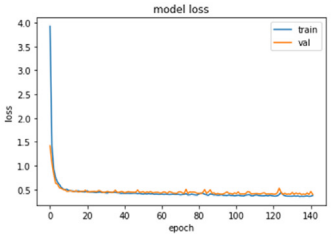
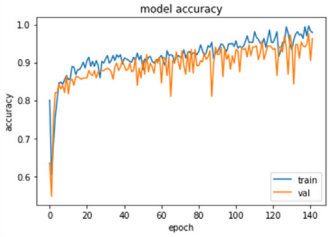
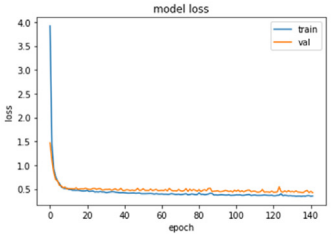
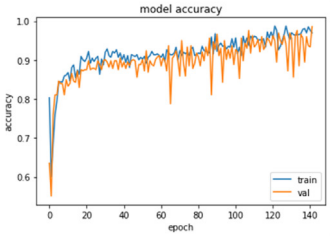
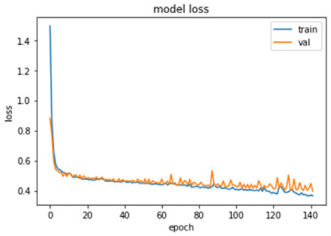
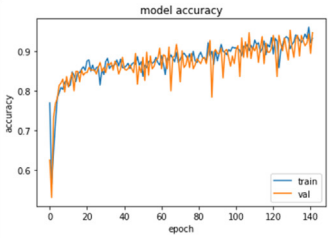
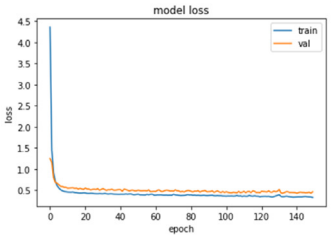
| Folds | Accuracy | Loss |
|---|---|---|
| Fold 1 |   |   |
| Fold 2 |   |   |
| Fold 3 |   |   |
| Fold 4 |   |   |
| Fold 5 |   |   |

**Table 7**
Summary of the proposed experiments.

| S. no | Method | Accuracy (in %) |
|---|---|---|
| 1 | Baseline CNN model | 84 |
| 2 | Residual CNN model | 95 |
| 3 | Attention embedded Residual CNN model | 98 |

## 5. Conclusion

This research presents an efficient mechanism to detect the type of infestation in tomato leaves. To the best of our knowledge, this is the first attempt to employ the attention gating mechanism in residual CNN for disease detection in tomatoes. The main contribution of this work is the integration of attention mechanism on top of the Residual network for effective feature learning. It helps to selectively weigh the features different layers at the inception of a single layer. Hence, the receptive field at a

**Table 8**
Performance comparison of proposed work with other existing works.

| S. No | Source | Type of features | Method | Size of dataset | Accuracy (in %) |
|---|---|---|---|---|---|
| 1 | Ferdouse et al. [32] | Automatic | CNN | 3000 | 76 |
| 2 | Chit Su Hlaing et al. [20] | Hand-Crafted features | Quadratic SVM | 3535 | 83.5 |
| 3 | Melike Sardogan et al. [35] | Automatic | CNN with LVQ | 500 | 86 |
| 4 | P. B. Padol et al [13] | Hand-Crafted features | SVM classifier | 137 | 88.89 |
| 5 | J. Shijie et al. [21] | Automatic | VGG16 based CNN | 7040 | 89 |
| 6 | Azeddine Elhassouny et al. [37] | Automatic | CNN | 7176 | 90.3 |
| 7 | Semary et al. [8] | Hand-Crafted features | SVM | 708 | 92 |
| 8 | P. Tm et al. [27] | Automatic | LeNet based CNN | 54306 | 95 |
| 9 | Suryawati et al. [22] | Automatic | VGGNet based CNN | 18160 | 95.24 |
| 10 | Jayme Garcia et al. [24] | Automatic | GoogleNet based CNN | 40409 | 96 |
| 11 | Sladojevic et al [30] | Automatic | CNN | 30880 | 96.3 |
| 12 | Halil Durmus et al. [38] | Automatic | SqueezeNet | 54309 | 97.22 |
| 13 | Keke Zhang et al. [25] | Automatic | ResNet | 5550 | 97.28 |
| 14 | Sabrol et al. [12] | Hand-Crafted features | Decision Tree | 383 | 97.3 |
| 15 | Proposed approach | Automatic | Attention based Residual CNN | 95999 | 98 |

layer is extended to look at feature maps from different levels of the processing hierarchy. The current layer can now process its input with more contextual information. Learning at the layers preceding the current layer is now aided by the perception of the features at the current layer. This is due to back propagation of the tensors along the skip connections.

The proposed network learnt around 600 K parameters to detect the type of infection, which is comparatively less than the existing deep learning approaches reported in the literature. Experimental results indicate that the proposed attention based residual network was able to detect the type of infection with an accuracy of 98%. It could also be noted that the ARPFE blocks establish the extensibility of the design of the proposed system to any input size.

## Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2019.105933.

## References


[1] R. Anand, S. Veni, J. Aravinth, An application of image processing techniques for detection of diseases on brinjal leaves using k-means clustering method, in: 2016 International Conference on Recent Trends in Information Technology, ICRTIT, 2016, pp. 1–6.

[2] K. Thangadurai, K. Padmavathi, Computer vision image enhancement for plant leaves disease detection, in: 2014 World Congress on Computing and Communication Technologies, 2014, pp. 173–175.

[3] C. Mattihalli, E. Gedefaye, F. Endalamaw, A. Necho, Real time automation of agriculture land, by automatically detecting plant leaf diseases and auto medicine, in: 2018 32nd International Conference on Advanced Information Networking and Applications Workshops, WAINA, 2018, pp. 325–330.

[4] Y. Liu, S. Zhou, J. Sun, Detection of Ginseng leaf cicatrices base on K-means clustering algorithm, in: 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, CISP-BMEI, 2017, pp. 1–5.

[5] V. Singh, Varsha, A.K. Misra, Detection of unhealthy region of plant leaves using image processing and genetic algorithm, in: 2015 International Conference on Advances in Computer Engineering and Applications, 2015, pp. 1028–1032.

[6] P. Lottes, J. Behley, A. Milioto, C. Stachniss, Fully convolutional networks with sequential information for robust crop and weed detection in precision farming, IEEE Robot. Autom. Lett. 3 (4) (2018) 2870–2877.

[7] A. Akhtar, A. Khanum, S. A. Khan, A. Shaukat, Automated Plant Disease Analysis (APDA): Performance comparison of machine learning techniques, in: Proceedings of the 11th International Conference on Frontiers of Information Technology, 2013, pp. 60–65.

[8] N.A. Semary, A. Tharwat, E. Elhariri, A.E. Hassanien, Fruit-based tomato grading system using features fusion and support vector machine, Intell. Syst. (2015) 401–410.

[9] S. Prasad, S.K. Peddoju, D. Ghosh, Multi-resolution mobile vision system for plant leaf disease diagnosis, Signal, Image Video Process. 10 (2) (2015) 379–388.

[10] D. Ashourloo, H. Aghighi, A.A. Matkan, M.R. Mobasheri, A.M. Rad, An investigation into machine learning regression techniques for the leaf rust disease detection using hyperspectral measurement, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 9 (9) (2016) 4344–4351.

[11] Parikh, M.S. Raval, C. Parmar, S. Chaudhary, Disease detection and severity estimation in cotton plant from unconstrained images, in: 2016 IEEE International Conference on Data Science and Advanced Analytics, DSAA, 2016, pp. 594–601.

[12] H. Sabrol, K. Satish, Tomato plant disease classification in digital images using classification tree, in: 2016 International Conference on Communication and Signal Processing, ICCSP, 2016, pp. 1242–1246.

[13] P.B. Padol, A.A. Yadav, SVM classifier based grape leaf disease detection, in: Proceedings of the Conference on Advances in Signal Processing, CASP, 2016, pp. 175–179.

[14] S. Kaur, S. Pandey, S. Goel, Semi-automatic leaf disease detection and classification system for soybean culture, IET Image Process. 12 (6) (2018) 1038–1048.

[15] T. Mehra, V. Kumar, P. Gupta, Maturity and disease detection in tomato using computer vision, in: 2016 Fourth International Conference on Parallel, Distributed and Grid Computing, PDGC, Waknaghat, 2016, pp. 399–403.

[16] Annis. Fathima, R. Karthik, V. Vaidehi, Image stitching with combined moment invariants and SIFT features, Elsevier Procedia Comput. Sci. 19 (2013) 420–427.

[17] R. Karthik, Annis Fathima, V. Vaidehi, Panoramic view creation using invariant moments and SURF features, in: Third IEEE International Conference on Recent trends in Information technology, ICRTIT, 2013.

[18] R. Menaka, R. Karthik, A novel feature extraction scheme for visualisation of 3D anatomical structures, Int. J. Biomed. Eng. Technol. 21 (1) (2016) 49–66.

[19] Y. Dandawate, R. Kokare, An automated approach for classification of plant diseases towards development of futuristic decision support system in Indian perspective, in: Proceedings of the International Conference on Advances in Computing, Communications and Informatics, ICACCI, 2015, pp. 794–99.

[20] C.S. Hlaing, S.M. Maung Zaw, Tomato plant diseases classification using statistical texture feature and color feature, in: 2018 IEEE/ACIS 17th International Conference on Computer and Information Science, ICIS, Singapore, 2018, pp. 439–444.

[21] J. Shijie, J. Peiyi, H. Siping, s. Haibo, Automatic detection of tomato diseases and pests based on leaf images, in: 2017 Chinese Automation Congress, CAC, 2017, pp. 2537–2510.

[22] E. Suryawati, R. Sustika, R.S. Yuwana, A. Subekti, H.F. Pardede, Deep structured convolutional neural network for tomato diseases detection, in: 2018 International Conference on Advanced Computer Science and Information Systems, ICACSIS, 2018, pp. 385–390.

[23] Aravind Krishnaswamy Rangarajan, Raja Purushothaman, Aniirudh Ramesh, Tomato crop disease classification using pre-trained deep learning algorithm, Procedia Comput. Sci. 133 (2018) 1040–1047.

[24] Jayme Garcia Arnal Barbedo, Plant disease identification from individual lesions and spots using deep learning, Biosyst. Eng. 180 (2019) 96–107.

[25] Keke Zhang, Qiufeng Wu, Anwang Liu, Xiangyan Meng, Can deep learning identify tomato leaf disease?, Adv. Multimedia 2018 (2018) 10, 6710865.

[26] S. Liang, W. Zhang, Accurate image recognition of plant diseases based on multiple classifiers integration, in: Y. Jia, J. Du, W. Zhang (Eds.), Proceedings of 2019 Chinese Intelligent Systems Conference, CISC 2019, in: Lecture Notes in Electrical Engineering, vol. 594, Springer, 2020.

[27] P. Tm, A. Pranathi, K. SaiAshritha, N.B. Chittaragi, S.G. Koolagudi, Tomato leaf disease detection using convolutional neural networks, in: 2018 Eleventh International Conference on Contemporary Computing, IC3, 2018, pp. 1–5.

[28] Q.H. Cap, H. Tani, H. Uga, S. Kagiwada, H. Iyatomi, Super-resolution for practical automated plant disease diagnosis system, in: 2019 53rd Annual Conference on Information Sciences and Systems, CISS, Baltimore, MD, USA, 2019, pp. 1–6.

[29] Qiaokang Liang, Shao Xiang, Yucheng Hu, Gianmarc Coppola, Dan Zhang, Wei Sun, PD2SE-Net: Computer-assisted plant disease diagnosis and severity estimation network, Comput. Electron. Agric. 157 (2019) 518–529.

[30] Srdjan Sladojevic, Marko Arsenovic, Andras Anderla, Dubravko Culibrk, Darko Stefanovic, Deep neural networks based recognition of plant diseases by leaf image classification, Comput. Intell. Neurosci. (2016) 11, 3289801.

[31] Alvaro Fuentes, Sook Yoon, A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition, Sensors 17 (2017) 1–21.

[32] M. Ferdouse Ahmed Foysal, M. Shakirul Islam, S. Abujar, S. Akhter Hossain, A novel approach for tomato diseases classification based on deep convolutional neural networks, in: Uddin M. Bansal J. (Ed.), Proceedings of International Joint Conference on Computational Intelligence, in: Algorithms for Intelligent Systems., Springer, 2020.

[33] N. Ruedeeniraman, M. Ikeda, L. Barolli, Performance evaluation of vegeCare tool for tomato disease classification, in: L. Barolli, H. Nishino, T. Enokido, Takizawa M. (Eds.), Advances in Networked-Based Information Systems. NBiS - 2019 2019, in: Advances in Intelligent Systems and Computing, vol. 1036, Springer, 2020.

[34] A. Fuentes, D.H. Im, S. Yoon, D.S. Park, Spectral analysis of CNN for tomato disease identification, in: L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. Zadeh, J. Zurada (Eds.), Artificial Intelligence and Soft Computing. ICAISC 2017, in: Lecture Notes in Computer Science, vol. 10245, Springer, 2017.

[35] M. Sardogan, A. Tuncer, Y. Ozen, Plant leaf disease detection and classification based on CNN with LVQ algorithm, in: 2018 3rd International Conference on Computer Science and Engineering, UBMK, 2018, pp. 382–385.

[36] H.F. Pardede, E. Suryawati, R. Sustika, V. Zilvan, Unsupervised convolutional autoencoder-based feature learning for automatic detection of plant diseases, in: 2018 International Conference on Computer, Control, Informatics and its Applications, IC3INA, 2018, pp. 158–162.

[37] A. Elhassouny, F. Smarandache, Smart mobile application to recognize tomato leaf diseases using Convolutional Neural Networks, in: 2019 International Conference of Computer Science and Renewable Energies, ICCSRE, Agadir, Morocco, 2019, pp. 1–4.

[38] H. Durmuş, E.O. Güneş, M. Kırcı, Disease detection on the leaves of the tomato plants by using deep learning, in: 2017 6th International Conference on Agro-Geoinformatics, Fairfax, VA, 2017, pp. 1–5.