Journal of Computer Science

Original Research Paper

A Backward Regressed Capsule Neural Network for Plant Leaf Disease Detection

[1]Jennifer Jepkoech, [2,3]Benson Kipkemboi Kenduiywo, [1]David Muchangi Mugo and [4]Edna Chebet Tool

[1]Department of Computer Science, University of Embu, Kenya
[2]Department of Computing and Information Technology, University of Embu, Kenya
[3]Department of Geomatics Engineering, Jomo kenyatta University of Science and Technology, Kenya
[4]Department of Computing and Information Technology, Chuka University, Kenya

Article history
Received: 22-11-2021
Revised: 03-08-2022
Accepted: 07-08-2022

Corresponding Author:
Jepkoech Jennifer
Department of Computer Science, University of Embu, Kenya
Email: jepkoech.jennifer@embuni.ac.ke

Abstract: This study investigated the introduction of backward regression coupled with DenseNet features in a Capsule Neural Network (CapsNet) for plant leaf disease classification. Plant diseases are considered one of the main factors influencing food production and therefore fast crop disease detection and recognition are important in enhancing food security interventions. CapsNets have successfully been adopted for plant leaf disease classification however, backpropagation of signals to preceding layers is still a challenge due to low gradient flow. In addition, parameter and computational complexities exist due to complex features. Therefore, this study implemented a loop connectivity pattern to improve gradient flow in the convolution layer and backward regression for feature selection. We observed a 99.7% F1 score with backward regression and 87% F1 score without backward regression accuracy on testing our framework based on the standard Plant Village (PV) dataset comprising ten tomato classes with 9080 images. Additionally, CapsNet with backward regression showed relatively higher and stable accuracy when sensitivity analysis was performed by varying testing and training dataset percentages. In comparison Support Vector Machines (SVM), Artificial Neural Networks (ANN), AlexNet, ResNet, VGGNet, Inception V3, and VGG 16 deep learning approaches scored 84.5, 88.6, 99.3, 97.87, 99.14, and 98.2%, respectively. These findings indicate that the introduction of backward regression of features in the CapsNet model may be a decent and, in most cases superior and less expensive alternative for phrase categorization models based on CNNs and RNNs. Therefore, the accuracy of plant disease detection may be enhanced even further with the aid of the fusion of several classifiers and the integration of a backward regressed capsule neural network.

Keywords: DenseNet, Plant Leaf, Convolution Neural Network, Capsule Neural Network, Model Training, Deep Learning

Introduction

The basic goal of smart farming is to develop innovative solutions for the future sustainability of mankind (Patil and Kumar, 2020). However, plant disease is detrimental to this goal as it destroys crops or diminishes their overall quality. In addition, the use of pesticides to control the spread of plant diseases renders the soil contaminated, which after some time becomes unsuitable for sowing and planting (Jadhav *et al*., 2021). Therefore, in addition to other challenges, plant diseases contribute significantly to food insecurity, malnutrition, and poverty in Africa, where most people depend on agriculture (Hasan *et al*., 2020; Li *et al*., 2020).

Different plant species are affected by different plant diseases caused by factors related to climate change, soil and plant nutrients, pests, and organic soil content, among others (Barbedo, 2020). So far, different techniques have been used to recognize plant diseases (Barbedo, 2020). With the increasing use of smartphones and internet services, mobile phones can easily be used to detect plant diseases. Smartphones have high-resolution cameras and they can also be used to perform computational tasks (Jadhav *et al*., 2020).

Manual plant disease recognition methods (Jogekar and Tiwari, 2021) are widespread but limited, ineffective, costly, and time-consuming hence automatic and efficient recognition methods may be an alternative. Convolutional Neural Networks (CNNs) including other deep models such as ResNet (He *et al*., 2016), GoogleNet, VGG (Kim and Rhee, 2018) and AlexNet (Wang *et al*., 2019) have been applied in other jurisdictions in an attempt to solve these problems (Fuentes *et al*., 2018; Kwabena *et al*., 2020b; Zhang *et al*., 2019; Kwabena *et al*., 2020a). They have achieved impressive results in this domain but tend to be 'data-hungry', invariant and vulnerable to problems that can easily lead to mis-classifications (Al-Furas *et al*., 2019). The state-of-the-art CNN uses pooling that not only leads to the loss of important features but also increases the number of parameters and complexities in models (Al-Furas *et al*., 2019). A survey by Patrick *et al*. (2022) establishes that Capsule Neural Networks (CapsNets) perform better than traditional CNNs due to the aforementioned limitations. This is large because CapsNets can connect spatial data and convolution layers hence efficient for image classification.

CapsNets have been widely used as plant disease classifiers. When integrated with VGG-16 (OxfordNet), CapsNets can reduce over-fitting and improve detection accuracy (Simonyan and Zisserman, 2014; Patrick *et al*., 2022). CapsNet architectures require encoding image input and computation of the class probability using the SoftMax method. Hinton *et al*. (2018) showed that the use of small training datasets negatively impacts accuracy rates. Similarly, they have a limiting effect on the effectiveness of the training model. Contrary, Ferentinos (2018) notes that CapsNet architectures are effective when used on small image datasets. The classifications of plant leaves can as well be achieved through other methods such as the use of the Support Vector Machines (SVM) (Poojary and Shabari, 2018; Das *et al*., 2020), Euclidean classifier (KNN), and the Artificial Neural Network (ANN). Even though CapsNets do well on small data sizes, they have difficulty recognizing images in complex backgrounds (Patrick *et al*., 2022). The current state-of-the-art CapsNet has a very weak gradient flow because it is difficult for signals to be back-propagated (Jogekar and Tiwari, 2021). CapsNets also lack a technique for efficient parameter selection which leads to computational inefficiency due to limited feature diversification. The deeper a CapsNet becomes, the more complex it gets (Patrick *et al*., 2022).

In an attempt to strengthen CapsNet, this study presents the following contribution: (1) We introduce loop connectivity to CapsNet in place of single normal convolutions to make it easier for signal backpropagation and to improve gradient flow. This minimizes errors during classification. (2) We introduce backward regression as a feature selection approach to capture significant parameters for further processing. This process was done to reduce the computational complexities and promote parameter efficiency in the model.

Related Work

Several studies have been conducted on plant disease detection. Sullca *et al*. (2019) used computer vision and machine learning to identify illnesses in blueberry leaves. In Kumar and Vani (2019) CNN was used to identify tomato leaf diseases. The CNN framework had two components. The first component consisted of a model for feature extraction made up of four convolution layers with the activation function ReLU and max pooling, while the second part was made up of two dense layers and a flattening layer. The activation for the second section was Softmax. However, only a small number of diseases were considered in the study because the data-gathering technique was laborious and time-consuming. CNN has also been used in a transfer learning framework to detect different types of tomato diseases and pests Llorca *et al*. (2018). The study concluded that an increase in the number of tomato disease classes reduced over-fitting problems. Another CNN-based study by Ferentinos (2018) examined plant disease classifications using Plant Village dataset and determined that transfer learning techniques were more accurate than the "from scratch" learning techniques. The results proved that CNN models could be adopted for different plant species and the architectures are more effective when used in large plant datasets.

According to Lowe *et al*. (2017), CNN architecture requires a substantial training time for the neurons, but the method is widely recognized due to its high classification accuracy. Ferentinos (2018) proposed a deep learning method that uses a large training image dataset collected from different geographical locations and cultivation conditions to increase accuracy potential. This showed that the training model is highly dependent on the volume and quality of the input data for better outputs. Pöpperli *et al*. (2019) further argue that CapsNet models perform better than CNN when an absolute value is used as the input data.

Other plant disease detection models in the literature have shown promising results. However, most of them are deep, complicated, invariant, not resilient, underperforming, and lacking in adaptability. They're also colorless, textureless, spatially inert, and deformable. Due to these flaws, CapsNets were developed with the ability to encode spatial information, texture, color, and deformation. Capsules are ideally suited for crop disease detection because texture and orientation play important roles in recognizing leaf sections that do not correspond to the rest of the leaf. As a result, we propose an enhanced backward regressed capsule neural network for plant disease diagnosis, which is particularly beneficial for inputs with uncertain probability distributions. Thus the main contributions of this study are to strengthen gradient

flow through the use of loop connectivity, promote computational and parameter efficiency through feature diversification, maintain low complexity by using a combination of complex and simple features, and lastly use backward regression for selection of significant feature maps in the model after the first convolution which reduced characters in the model.

Materials and Methods

Data

Despite the limited number of leaf disease categories and plant disease images, most studies have used the Plant Village (PV) dataset (Brahimi *et al*., 2017; Mohanty *et al*., 2016; Barbedo, 2018). The PV dataset is categorized into different classes of plant diseases. Studies have shown that one specific plant may be recorded in different plant disease classes within the PV dataset. At the same time, similar plant diseases having the same common name may equally be recorded in a different class of plant diseases. These variations are evident in the works of authors who used CNN architectures for crop disease classifications (Mohanty *et al*., 2016; Lowe *et al*., 2017; Too *et al*., 2019; Dou *et al*., 2019). Therefore, we adopted images from the PV database in our study. A total of 9080 tomato images were subdivided into ten disease classes namely: Mosaic virus, bacterial spot, early blight, late blight, leaf mold, healthy septoria spot, target spot, yellow leaf curl, and spider mite as collected by Hasan *et al*. (2019).

Overview of Capsule Neural Networks (CapsNets)

Capsules are groups of firmly and deeply fixed neurons while Capsule Neural Networks (CapsNets) are collections of capsules. The current state-of-the-art comprises a single convolution layer, a primary capsule layer, and a digit layer (Sabour *et al*., 2017). The input layer encompasses the pre-processing procedures where an image's size is modified to 28 × 28. The hidden layer comprises a convolutional layer with a kernel of magnitude 9 × 9 with a stride of one for feature extraction, followed by a ReLU function that helps in feature activation. The next layer is the primary capsule layer which has a kernel size of 9 × 9 with a stride of two and deals with feature map tensors. We introduced backward regression after the primary CapsNets layer to select only the significant features. The selection of features through regression is aimed at reducing complexities in the model. The output from the primary capsule layer is also activated by a ReLU function (Sabour *et al*., 2017). It contains a decoder network with three dense layers and a routing by agreement algorithm proposed by Hinton *et al*. (2018). Brahimi *et al*. (2017) used the algorithm to detect movements in movies.

Let the value of the lower capsule used to store image data be *j*, then the output of that lower capsule is $\hat{u}_j|I$, and its prediction of the higher-level capsule I am computed as:

$$\hat{u}_{j|i} = w_{ij} u_{j|i} \tag{1}$$

where, w_{ij} is a weighting matrix learned through back-propagation while $u_{j|I}$ denotes the vector that *I* use for the prediction of the *j*th capsule. The role of each capsule is to predict the output of the higher-level capsules. The coupling coefficients increase if the prediction conforms to the output of the higher-level capsule (Patrick *et al*., 2022). Equation (2) (Sabour *et al*., 2017) shows the SoftMax function that is used to calculate the coupling coefficients:

$$c_{ij} = \frac{\exp\left(b_{ij}\right)}{\sum_k \exp\left(b_{ik}\right)} \tag{2}$$

The variables in Eq. (2) encompass the coupling coefficient c_{ij}, the log probabilities b_{ij} which is set to zero at the start of routing by agreement and b_{ik} is a normalization term that ensures that all output values are within the range of 0 and 1, i.e., a valid probability distribution. The log probabilities are significant in determining whether the lower-level capsule *I* can be coupled with the higher-level capsule *j*. Using Eq. (2), the input vector to the higher capsules will be calculated as:

$$c_{ij} = \hat{u}_{j|i} \cdot u_j + b_j \; where; b \neq 0 \tag{3}$$

where, v_j is the output of capsule *j*. Since the probability of existence is represented by the length of the output vector, short vectors need to be reduced to an almost zero value while long vectors are increased to a near one value. To achieve this we used Eq. (4):

$$u_j = \frac{\left(\left\|s_j\right\|^2\right) s_j}{1 + \left\|s_j\right\| \left\|s_j\right\|} \tag{4}$$

where, v_j is the vector output of capsule *j* and s_j is its total input while b_{ij} is updated during routing by agreement between v_j and $u_{j|i}$. This follows the rule that says, 'if two vectors agree, the inner product will be large. The agreement a_{ij} performing updates between log probabilities b_{ij} and coupling coefficients c_{ij} is calculated as:

DOI: 10.3844/jcssp.2022.821.831

$$a_{ij} = u_j \cdot \hat{u}_{j|I}. \tag{5}$$

To describe the whole routing procedure for computation of high-level vector, Eq. (2) and (6) are used.

Various properties of the entities of an image such as size, orientation, and position are encapsulated by the directions of vectors to allow the capsules to learn the relationships between features within an image. The loss function is used to equalize the values between zero and one. Each capsule in the last layer is associated with the loss function l_k computed:

$$l_k = T_k \max\left(0, m^+ - \| v_k \|\right)^2 + \lambda\left(1 - T_k\right)\max\left(0, \left\|v_k\right\| - m^-\right)^2 \tag{6}$$

where, T_k is equal to one if a digit of class k is present otherwise zero and $m^+ = 0.9$ and $m^- = 0.1$. The λ down-weighting of the loss for absent digit classes stops the initial learning from shrinking the lengths of the activity vectors of all the digit capsules.

Backward Regression

To eliminate feature maps that did not contain any significant information for the classification, backward regression was used (Fig. 1). To begin with, the significance level p of the model was selected as 0.5, then fitting was done and all independent variables were included. The predictor with the highest p-value was then identified. If the P-value of a feature did not satisfy the set threshold it was rejected and removed from the dataset for failing to satisfy the 95% confidence level and the model fitted again. If the P-value of the feature, which was the highest in the set, was less than the significance level, we just stopped comparing and forwarded the feature maps to the primary capsule for further processing. This was done repeatedly until all the significant features were identified. For example, consider our model m with a total of n predictors/features i.e., $x = x_1, x2, \ldots, x_n$, the role of our backward regression is to estimate significant features to classify k classes as:

$$y\left(m_k\right) = b_0 + b_1 x_1 + \cdots + b_1 x_n + \epsilon \tag{7}$$

where, $y(m_k)$ are chosen significant features for model k, b_0 is the y-intercept, b_1 is the slope parameter, and ϵ is the error term. The backward regression process iterates over k models determining suitable features for each model. For instance given $k = 10$ which can form k models, i.e., m_{10}, if each class is executed individually with say $n = 10$ then for each of the models if one useless prediction is removed we remain with 9. To begin with, the original tree before regression is shown by Eq. (8):

$$\begin{aligned} y\left(m_k\right) = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4 + \\ b_5 x_5 + b_6 x_6 + b_7 x_7 + b_8 x_8 + b_9 x_9 + b_{10} x_{\leftarrow} + \epsilon \end{aligned} \tag{8}$$

Now if any of the features are considered insignificant and removed we end up with the model configurations in Eq. (9):

$$\begin{aligned}
y\left(m_{k-1}\right) &= b_0 + b_2 x_2 + b_3 x_3 + b_4 x_4 + b_5 x_5 + b_6 x_6 + b_7 x_7 + b_8 x_8 + b_9 x_9 + b_{10} x_{10} + \epsilon \\
y\left(m_{k-1}\right) &= b_0 + b_1 x_1 + b_3 x_3 + b_4 x_4 + b_5 x_5 + b_6 x_6 + b_7 x_7 + b_8 x_8 + b_9 x_9 + b_{10} x_{10} + \epsilon \\
y\left(m_{k-1}\right) &= b_0 + b_1 x_1 + b_2 x_2 + b_4 x_4 + b_5 x_5 + b_6 x_6 + b_7 x_7 + b_8 x_8 + b_9 x_9 + b_{10} x_{10} + \epsilon \\
y\left(m_{k-1}\right) &= b_0 + b_1 x_1 + b_2 x_2 + b_4 x_4 + b_5 x_5 + b_6 x_6 + b_7 x_7 + b_8 x_8 + b_9 x_9 + b_{10} x_{10} + \epsilon \\
y\left(m_{k-1}\right) &= b_0 + b_1 x_1 + b_2 x_2 + b_4 x_4 + b_5 x_5 + b_6 x_6 + b_7 x_7 + b_8 x_8 + b_9 x_9 + b_{10} x_{10} + \epsilon \\
y\left(m_{k-1}\right) &= b_0 + b_1 x_1 + b_2 x_2 + b_4 x_4 + b_5 x_5 + b_6 x_6 + b_7 x_7 + b_8 x_8 + b_9 x_9 + b_{10} x_{10} + \epsilon \\
y\left(m_{k-1}\right) &= b_0 + b_1 x_1 + b_2 x_2 + b_4 x_4 + b_5 x_5 + b_6 x_6 + b_7 x_7 + b_8 x_8 + b_9 x_9 + b_{10} x_{10} + \epsilon \\
y\left(m_{k-1}\right) &= b_0 + b1x1 + b_2 x_2 + b_4 x_4 + b_5 x_5 + b_6 x_6 + b_7 x_7 + b_8 x_8 + b_9 x_9 + b_{10} x_{10} + \epsilon \\
y\left(m_{k-1}\right) &= b_0 + b1x1 + b_2 x_2 + b_4 x_4 + b_5 x_5 + b_6 x_6 + b_7 x_7 + b_8 x_8 + b_9 x_9 + b_{10} x_{10} + \epsilon
\end{aligned} \tag{9}$$

The best model among the configurations in Eq. (9) was chosen based on R^2.

The Improved Capsule Neural Network Model

We used three convolutions and one primary capsule to capture diversified features where the convolutions followed the loop connectivity pattern of DenseNets (Fig. 2). The loop connectivity pattern was used to strengthen gradient flow by making it easy to propagate signals to earlier layers more directly, contributing to the parameter and computational efficiency through feature concatenation. It helped maintain low complexity in the model through the use of both complex and simple features.

The overall architecture contains three convolutional subnets which are arranged in a looping manner. Feature maps from each subnet are grouped like those in the work done by Jégou *et al*. (2017). To build primary capsules in the subnets, this study used convolutional layers with 3 × 3 kernels with a stride of 1, 5 × 5 kernels with a stride of 2, and 9 × 9 kernels with a stride of 2 respectively. This study also used padding to ensure equity in size. Figure 2 depicts the proposed architecture that we adopted. After each subnet, backward regression was used to select significant features only. Backward regression reduces the number of parameters hence minimizing computational complexity. The features from the subnets were squashed to form the PrimaryCaps layer. Routing by agreement (Jogekar and Tiwari, 2021) was used. xcolor We did not encounter any limitations in the methods used for the work.

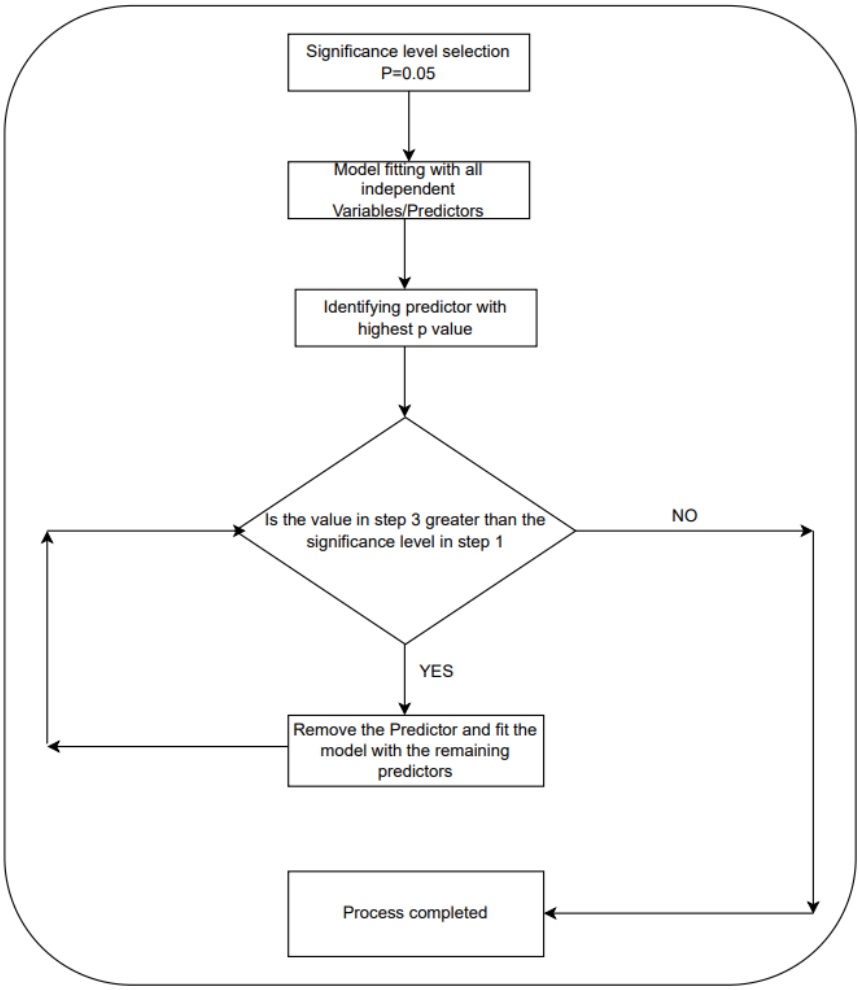

Fig. 1: Backward regression framework

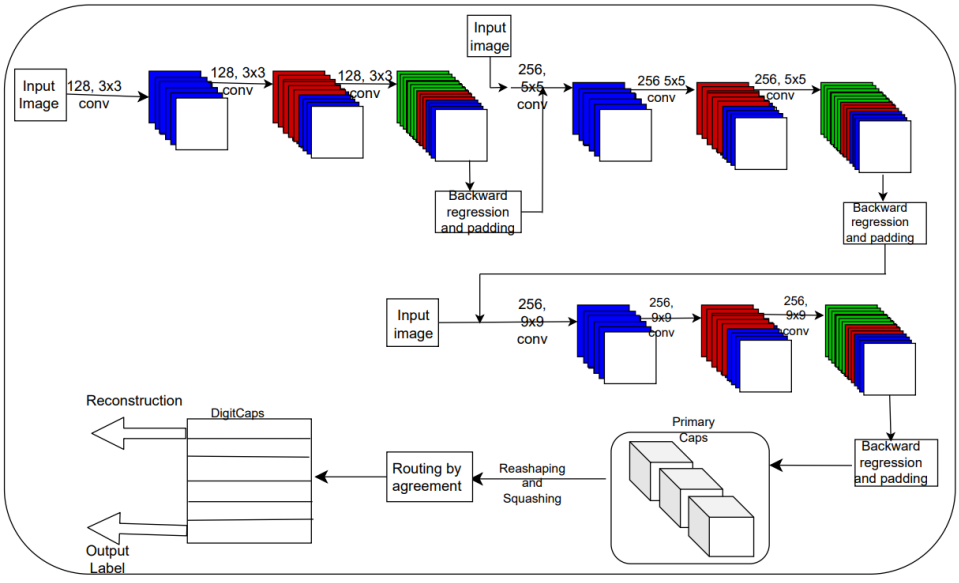

Fig. 2: Proposed architecture on routing by agreement

Results

The capsule neural network is based on parameters like momentum, batch size, learning rate, dropout, and learning rate decay. Because neural networks deal with datasets of the same size (Ferentinos, 2018) this study tuned the dataset parameters by modifying the input images to 28 × 28. Plant leaf disease classification was done using an improved Capsule neural network model and the conventional CapsNet. Confusion matrix, F1 scores, and graphical representations were used to find the accuracy of each model.

F1-score was computed as:

$$F1-score=\frac{2\times precision\times recall}{precision+recall} \tag{10}$$

Table 1 summarizes the disease classification accuracy based on the F1 score, precision, support, and recall metrics. The precision for the Tomato mosaic virus class was relatively low compared to the other classes of tomato leaf diseases. However, most of the predictions from the model were noted to be false positives. The recall results for the Early blight tomato leaf disease class were relatively low compared to the other classes of tomato leaf diseases. Only 90% of the actual Early blight tomato leaf diseases were correctly classified. The model perfectly classified the recall for the Target Spot tomato leaf disease class.

Figure 3 displays the accuracy and loss obtained from the backward regressed model while Fig. 4 shows the accuracy obtained from the conventional CapsNet model without backward regression. Generally, backward regressed CapsNet has a 99.9% F1-Score while the conventional CapsNet attained 87%.

Table 2 shows the sensitivity analysis of our new approach vis a' viz conventional CapsNet with no consideration for feature selection. CapsNet with regressed feature selection shows relatively higher accuracy than when featuring selection.

We further compared the CapsNet models with different deep learning models namely: SVM, Artificial Neural Networks (ANN), AlexNet, ResNet, VGGNet, GoogleNet, and InceptionNet V3, and observed results as shown in Table 3. For this experiment, we used a PV dataset with a ratio of 20:80% for testing and training respectively. Our model had the best testing and training accuracy followed by AlexNet, Inception V3, ResNet, ANN, Normal CapsNet and the last one was SVM.

Research done by Kurup *et al*. (2019) used a capsule neural network model to diagnose plant diseases. They used a dataset size of 54,306 images and obtained an accuracy of 94 percent. Research done by Verma *et al*. (2020) used transfer learning to create capsule networks for the classification of potato illnesses and compared their performance to a few famous pre-trained CNN models, notably ResNet18, VGG16, and GoogLeNet. Colored images of healthy and sick leaves from the PlantVillage dataset were utilized to train the models. With 91.83% accuracy, CapsNet demonstrated comparable performance to state-of-the-art CNN models. Research done by Kwabena *et al*. (2020a) suggests the application of the Gabor and Capsule networks distinguish hazy, distorted, and previously unknown tomato and citrus illness images. The suggested model achieves a test accuracy of 98.13%. According to the researchers, the technique may be applied to other crops and might be a valuable tool for detecting invisible plant illnesses under poor weather and lighting circumstances. Mensah *et al*. (2021) used the squared Euclidean distance, sigmoid function, and a'simple-squash' function instead of the dot product, SoftMax normalizer, and squashing function present in the dynamic routing method. Extensive trials on the three datasets revealed that the proposed model improves test accuracy consistently across the three datasets while also allowing for an increase in the number of routing iterations with no performance impact. On the tomato dataset, the suggested model beat a baseline CapsNet by 8.37 percent, with an overall test accuracy of 98.80 percent, equivalent to state-of-the-art models on the same datasets. Samin *et al*. (2021) built a deep learning architecture model (CapPlant) that uses plant photos to detect whether it is healthy or infected. The prediction procedure does not need handmade features; rather, the architecture extracts representations from the incoming data series automatically. To extract and categorize features, many convolutional layers are used. The last convolutional layer in CapPlant is replaced with a cutting-edge capsule layer that incorporates orientational and relative spatial relationships between distinct components of a plant in an image to more precisely forecast illnesses. The suggested architecture is validated using the PlantVillage dataset, which includes over 50,000 images of healthy and diseased. When compared to existing plant disease classification models, the CapPlant model showed significant gains in prediction accuracy. The generated model's testing findings obtained an overall test accuracy of 93.01 percent, with an F1 score of 93.07 percent. We compared the work done using the state-of-the-art CapsNet and observed that our work displayed a higher percentage of accuracy. This means that CapsNet performs better when improved through regression and the use of dense connectivity loops.

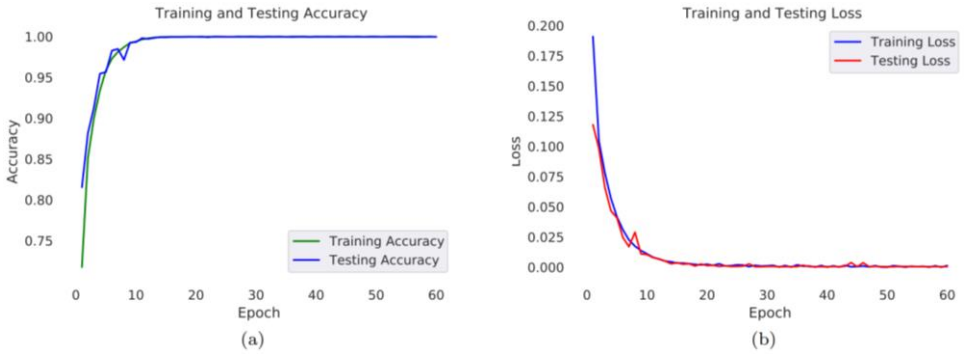

Fig. 3: Training and testing performance of CapsNet based on backward regression feature selection: (a) Percentage accuracy and (b) percentage loss

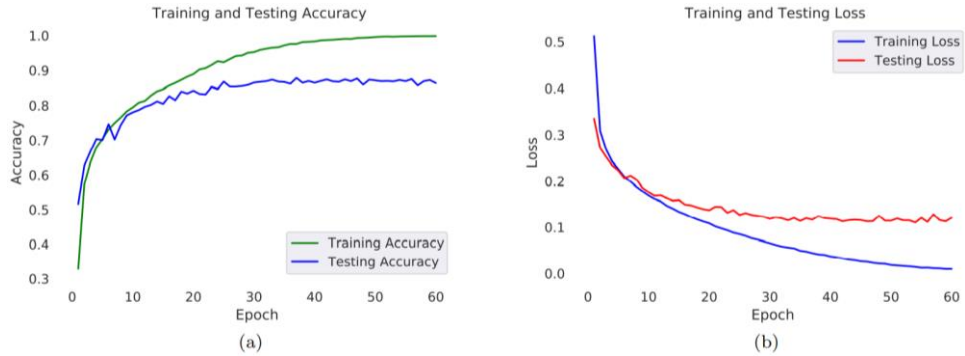

Fig. 4: Training and testing performance of CapsNet without backward regression feature selection: (a) Percentage accuracy and (b) percentage loss

Table 1: F1-score, precision, support, and recall metrics for individual disease classes using backward regressed and conventional CapsNets. P1 corresponds to precision, R1 recall with backward regression, P2 precision without backward regression, R2 recall without backward regression, F1(A) F1-score with backward regression, and F1(B) F1-score without backward regression

Class	P1	R1	P2	R2	F-1 (A)	F1 (B)	Support
Target spot	0.96	1.00	0.86	0.90	0.98	0.88	25
Tomato mosaic virus	0.94	0.97	0.84	0.87	0.96	0.85	143
Yellow leaf curl virus	0.97	0.99	0.87	0.88	0.98	0.87	183
Bacteria spot	0.97	0.96	0.88	0.86	0.96	0.87	208
Early blight	0.95	0.90	0.84	0.84	0.92	0.82	101
Healthy	0.96	0.96	0.86	0.85	0.96	0.86	191
Late blight	0.97	0.94	0.86	0.83	0.95	0.84	99
Leaf mold	0.97	0.95	0.86	0.85	0.96	0.86	184
Septoria leaf spot	0.98	0.99	0.85	0.86	0.99	0.86	522
Spider mites	0.99	0.99	0.86	0.86	0.99	0.87	160

Table 2: Sensitivity analysis of CapsNet with and without feature selection based on backward regression

Testing-Training (%)	F1-score without regression	F1-Score with regression
10-90	0.96	0.99
20-80	0.97	0.98
30-70	0.97	0.98
40-60	0.97	0.97
50-50	0.95	0.96
60-40	0.96	0.96
70-30	0.95	0.95
80-20	0.94	0.94
90-10	0.92	0.94

Table 3: Comparison of CapsNet models and other deep learning models based on F1-score percentage accuracy measure

Model	% Testing accuracy	%Training accuracy
SVM	84.50	99.0
ANN	88.60	100.0
AlexNet	99.30	100.0
ResNet	97.85	98.5
Inception v3	99.14	100.0
VGG 16	98.20	99.0
CapsNet	87.80	100.0
Backward Regressed CapsNet	99.90	100.0

Discussion

A backward regressed capsule neural network for plant leaf disease detection was proposed in this study. The PV datasets were split into an 80:20% ratio for training and testing for all the models. The margin and reconstruction losses make up the loss function that was used to train the model. The loss function's default settings for m+, m- and were kept in this implementation. The Capsule models were trained using three rounds of routing. Random adjustments to parameter values and intermediate layers were made in all of the models to see how responsive each model is to the modifications. The CNN models performed poorer than the planned Capsule models as a result of these adjustments. The CapsNets models' performance was unaffected by changes in momentum, batch size, learning rate, dropout, and learning rate decay. The number of routing iterations was the single most critical hyperparameter that substantially influenced the performance of the CapsNet models, with three yielding some performance values. The input images were downsized from 256 × 256 to 48 × 48, 68 × 68, and 224 × 224, and the models were trained, to show the CapsNets' versatility and resilience. On the other hand, CapsNet models generated fairly constant results when the picture size was increased. Increasing the picture size to 224 × 224, on the other hand, needed more computer resources and training time and hence was not feasible for this investigation.

We recommend the adoption of DenseNet architecture (Jégou *et al*., 2017) to strengthen the gradient, maintain low feature complexity and provide computational and parameter efficiency. We recommend the use of dense connectivity and backward regression because we were able to reduce model complexity in terms of time and number of parameters hence a record accuracy of 99.9% was observed. Research done by Pleiss *et al*. (2017) demonstrated that the use of DenseNet connectivity provides more accurate results because it is easier to propagate error signals more directly to earlier layers. To enhance the CapsNet, this study also used backward regression to reduce the number of parameters and ease computation in the model. Unlike pooling, which discards any data, regression only selects data with vital information. This idea improved the accuracy of this model, enhanced computational efficiency, and reduced complexities related to computation as depicted in Table 3. Sensitivity analysis of the model to different portions of training data also illustrates the robustness of CapsNet with regression compared when no regression is used (Table 2. Our model retained a higher accuracy overall, though the value dropped when the ratio of traintest was at 90-10%, it was still higher than CapsNet with no regression. Generally, it's expected that a model's accuracy decreases with a decrease in training data but generally, our model showed a good stable trend.

In this study, the PV dataset with 10 disease classes was used with the CapsNet model with backward regression. The PV provides a standard dataset that has been used to train and test several models with success (Ferentinos, 2018). Compared to Ferentinos (2018), we used a small plant database to develop the model. Ferentinos (2018) used five CNN architectures that included AlexNetOWTBn, VGG, GoogLeNet, AlexNet, and Overfeat. Each of the original images' architectures had a success rate of 99.44, 99.48, 97.27, 99.06, and 98.96%, respectively. The success rate percentages of the pre-processed images were 99.07% (AlexNetOWTBn), 98.87% (VGG), 97.06% (GoogLeNet), 98.64% (AlexNet) and 98.26% (Overfeat). Chaki and Parekh [56] support these results, who argued that the accuracy and success rate of plant leaf image detections should normally fall between 90 and 100%. This means that neural networks effectively use the leaf feature for plant disease classifications. Although this study used a relatively smaller dataset than Ferentinos (2018), the difference in accuracy was negligible hence there is no major effect if a large or small dataset is used. The use of a large dataset is essential regardless of any augmentation techniques and transfer learning systems used in the model (Llorca *et al*., 2018). However, according to Barbedo (2020), creating a large database with all the plant species and their related plant diseases is impractical a matter echoed by Kamilaris and Prenafeta-Boldu (2018). Therefore, novel models should ideally use moderate to small training datasets with perceivable minimum misclassification and high accuracy. Therefore, future efforts could also focus on architecture optimization and models that can utilize real-time smartphone models for plant disease classification in a bid to balance training data needs.

Conclusion

The adoption of DenseNet intuition based on loop connectivity patterns promoted strong gradient flow through easy error signal propagation, parameter, computational efficiency through channel-wise concatenation, and use of both complex and simple features that maintained low complexity. In addition, the dynamic routing eased predictability in the model after the primary capsule level. Further, backward regression in every subnet reduced the number of characters, a technique that selected the significant features and discarded those that were of no use to the model all while making a positive impact on detection accuracy.

Author's Contributions

Jennifer Jepkoech: Inception of idea, manuscript writing, Data collection and data analysis.

Benson Kipkemboi Kenduiywo: Coordinated the data-analysis and contributed to the writing of the manuscript.

David Muchangi Mugo: Data collection, analysis and proofreading.

Edna Chebet Tool: Experiments and result interpretation.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Al-Furas, A., El-Dosuky, M., & Hamza, T. (2019, December). Multi-column Deep Neural Network Based on Particle Swarm Optimization. In *2019 First International Conference of Intelligent Computing and Engineering (CHOICE)* (pp. 1-5). IEEE. https://doi.org/10.1109/ICOICE48418.2019.9035162

Barbedo, J. G. (2018). Factors influencing the use of deep learning for plant disease recognition. *Biosystems Engineering*, *172*, 84-91. https://doi.org/10.1016/j.biosystemseng.2018.05.013

Barbedo, J. G. A. (2020). Detecting and classifying pests in crops using proximal images and machine learning: A review. *AI*, *1*(2), 312-328. https://doi.org/10.3390/ai1020021

Brahimi, M., Boukhalfa, K., & Moussaoui, A. (2017). Deep learning for tomato diseases: classification and symptoms visualization. *Applied Artificial Intelligence*, *31*(4), 299-315. https://doi.org/10.1080/08839514.2017.1315516

Das, D., Singh, M., Mohanty, S. S., & Chakravarty, S. (2020, July). Leaf disease detection using a support vector machine. In *2020 International Conference on Communication and Signal Processing (ICCSP)* (pp. 1036-1040). IEEE. https://doi.org/10.1109/ICCSP48568.2020.9182128

Dou, Z. Y., Tu, Z., Wang, X., Wang, L., Shi, S., & Zhang, T. (2019, July). Dynamic layer aggregation for neural machine translation with routing-by-agreement. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 86-93). https://doi.org/10.1609/aaai.v33i01.330186

Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. Computers and Electronics in Agriculture. https://doi.org/10.1016/j.compag.2018.01.009

Fuentes, A. F., Yoon, S., Lee, J., & Park, D. S. (2018). High-performance deep neural network-based tomato plant diseases and pests diagnosis system with the refinement filter bank. *Frontiers in Plant Science*, *9*, 1162. https://doi.org/10.3389/fpls.2018.01162

Hasan, M., Tanawala, B., & Patel, K. J. (2019, March). Deep learning precision farming: Tomato leaf disease detection by transfer learning. In *Proceedings of 2nd international conference on advanced computing and software engineering (ICACSE)*. https://doi.org/10.2139/ssrn.3349597

Hasan, R. I., Yusuf, S. M., & Alzubaidi, L. (2020). Review of the state of the art of deep learning for plant diseases: A broad analysis and discussion. *Plants*, *9*(10), 1302. https://doi.org/10.3390/plants9101302

He, K., Zhang, X., Ren, S., & Sun, J. Deep residual learning for image recognition (2015). Cite. *arXiv preprint arxiv:1512.03385*. https://doi.org/10.1109/CVPR.2016.90

Hinton, G. E., Sabour, S., & Frosst, N. (2018, February). Matrix capsules with EM routing. In *International conference on learning representations*.

Jadhav, S. B., Udupi, V. R., & Patil, S. B. (2021). Identification of plant diseases using convolutional neural networks. *International Journal of Information Technology*, *13*(6), 2461-2470. https://doi.org/10.1007/s41870-020-00437-5

Jadhav, S., Udupi, V., & Patil, S. (2020, May). Classification of Soybean Diseases Using Pre-trained Deep Convolutional Neural Networks. In *International Conference on Image Processing and Capsule Networks* (pp. 746-756). Springer, Cham. https://doi.org/10.1007/978-3-030-51859-2_68

Jégou, S., Drozdzal, M., Vazquez, D., Romero, A., & Bengio, Y. (2017). The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 11-19). https://doi.org/10.1109/CVPRW.2017.156

Jogekar, R. N., & Tiwari, N. (2021). A review of deep learning techniques for identification and diagnosis of plant leaf disease. *Smart Trends in Computing and Communications: Proceedings of SmartCom 2020*, 435-441. https://doi.org/10.1007/978-981-15-5224-3_43

Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, *147*, 70-90. https://doi.org/10.1016/j.compag.2018.02.016

Kim, J. W., & Rhee, P. K. (2018). Image recognition is based on adaptive deep learning. *The Journal of The Institute of Internet, Broadcasting and Communication*, *18*(1), 113-117.

Kumar, A., & Vani, M. (2019, July). Image-based tomato leaf disease detection. In *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1-6). IEEE. https://doi.org/10.1109/ICCCNT45670.2019.8944692

Kurup, R. V., Anupama, M. A., Vinayakumar, R., Sowmya, V., & Soman, K. P. (2019, September). Capsule network for plant disease and plant species classification. In *International conference on computational vision and bio-inspired computing* (pp. 413-421). Springer, Cham. https://doi.org/10.1007/978-3-030-37218-7_47

Kwabena, P. M., Weyori, B. A., & Mighty, A. A. (2020a). Gabor capsule network for plant disease detection. *International Journal of Advanced Computer Science and Applications*, *11*(10). https://doi.org/10.14569/IJACSA.2020.0111048

Kwabena, P. M., Weyori, B. A., & Mighty, A. A. (2020b). Gabor capsule network for plant disease detection. *International Journal of Advanced Computer Science and Applications*, *11*(10). https://doi.org/10.14569/IJACSA.2020.0111048

Li, Z., Yu, T., Paul, R., Fan, J., Yang, Y., & Wei, Q. (2020). Agricultural nanodiagnostics for plant diseases: Recent advances and challenges. *Nanoscale Advances*, *2*(8), 3083-3094. https://doi.org/10.1039/C9NA00724E

Llorca, C., Yares, M. E., & Maderazo, C. (2018, March). Image-based pest and disease recognition of tomato plants using a convolutional neural network. In *Proceedings of international conference technological challenges for a better world*.

Lowe, A., Harrison, N., & French, A. P. (2017). Hyperspectral image analysis techniques for the detection and classification of the early onset of plant disease and stress. *Plant Methods*, *13*(1), 1-12. https://doi.org/10.1186/s13007-017-0233-z

Mensah, P. K., Weyori, B. A., & Ayidzoe, M. A. (2021). Capsule network with K-Means routingfor plant disease recognition. *Journal of Intelligent & Fuzzy Systems*, *40*(1), 1025-1036. https://doi.org/10.3233/JIFS-201226

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, *7*, 1419. https://doi.org/10.3389/fpls.2016.01419

Patil, R., & Kumar, S. (2020). Bibliometric survey on the diagnosis of plant leaf diseases using artificial intelligence. *International Journal of Modern Agriculture*, *9*(3), 1111-1131.

Patrick, M. K., Adekoya, A. F., Mighty, A. A., & Edward, B. Y. (2022). Capsule networks–a survey. *Journal of King Saud University-Computer and Information Sciences*, *34*(1), 1295-1310. https://doi.org/10.1016/j.jksuci.2019.09.014

Pleiss, G., Chen, D., Huang, G., Li, T., Van Der Maaten, L., & Weinberger, K. Q. (2017). Memory-efficient implementation of densenets. *arXiv preprint arXiv:1707.06990*.

Poojary, H., & Shabari, S. B. (2018, March). A survey on plant disease detection using a support vector machine. In *2018 International Conference on Control, Power, Communication and Computing Technologies (ICCPCCT)* (pp. 292-295). IEEE. https://doi.org/10.1109/ICCPCCT.2018.8574314

Pöpperli, M., Gulagundi, R., Yogamani, S., & Milz, S. (2019, June). Capsule neural network-based height classification using low-cost automotive ultrasonic sensors. In *2019 IEEE Intelligent Vehicles Symposium (IV)* (pp. 661-666). IEEE. https://doi.org/10.1109/IVS.2019.8813879

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. *Advances in neural information processing systems*, *30*.

Samin, O. B., Omar, M., & Mansoor, M. (2021). CapPlant: A capsule network-based framework for plant disease classification. *PeerJ Computer Science*, *7*, e752. https://doi.org/10.7717/peerj-cs.752

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sullca, C., Molina, C., Rodríguez, C., & Fernández, T. (2019). Diseases detection in blueberry leaves using computer vision and machine learning techniques. *International Journal of Machine Learning and Computing*, *9*(5), 656-661. https://doi.org/10.18178/ijmlc.2019.9.5.854

Too, E. C., Yujian, L., Njuki, S., & Yingchun, L. (2019). A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture*, *161*, 272-279. https://doi.org/10.1016/j.compag.2018.03.032

Verma, S., Chug, A., & Singh, A. P. (2020). Exploring capsule networks for disease classification in plants. *Journal of Statistics and Management Systems*, *23*(2), 307-315. https://doi.org/10.1080/09720510.2020.1724628

Wang, H., Polden, J., Jirgens, J., Yu, Z., & Pan, Z. (2019, July). Automatic rebar counting using image processing and machine learning. In *2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control and Intelligent Systems (CYBER)* (pp. 900-904). IEEE. https://doi.org/10.1109/CYBER46603.2019.9066509

Zhang, S., Huang, W., & Zhang, C. (2019). Three-channel convolutional neural networks for vegetable leaf disease recognition. *Cognitive Systems Research*, *53*, 31-41. https://doi.org/10.1016/j.cogsys.2018.04.006