Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 10, No. 2, 2022, 285-302.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

AgriNet: a New Classifying Convolutional Neural Network for Detecting Agricultural Products' Diseases

Farzaneh Salimian Najafabadi [1] and Mohammad Taghi Sadeghi [2*]

1. Azadi Campus, Yazd University, Yazd, Iran.

2. Department of Electrical Engineering, Yazd University, Yazd, Iran.

Article History:
Received 04 November 2021
Revised 08 January 2022
Accepted 13 March 2022

DOI:10.22044/jadm.2022.11360.2295

Keywords:
Plant Diseases Recognition, Convolutional Neural Network, ShuffleNet, SENet.

**Corresponding author: m.sadeghi@yazd.ac.ir(M. T. Sadeghi).*

Abstract

An important sector that has a significant impact on the economies of countries is the agricultural sector. The researchers are trying to improve this sector using the latest technologies. One of the problems facing the farmers in the agricultural activities is the plant diseases. If a plant problem is diagnosed soon, the farmer can treat the disease more effectively. This work introduces a new deep artificial neural network called AgriNet, which is suitable for recognizing some types of agricultural diseases in a plant using images from the plant leaves. The proposed network makes use of the channel shuffling technique of ShuffleNet and the channel dependency modeling technique of SENet. One of the factors influencing the effectiveness of the proposed network architecture is how to increase the flow of information in the channels after explicitly modelling the interdependencies between the channels. This is, in fact, an important novelty of this research work. The dataset used in this work is PlantVillage, which contains 14 types of plants in 24 groups of healthy and diseased. Our experimental results show that the proposed method outperforms the other methods in this area. AgriNet leads to an accuracy and a loss of 98% and 7%, respectively, on the experimental data. This method increases the recognition accuracy by about 2%, and reduces the loss by 8% compared to the ShuffleNetV2 method.

1. Introduction

Deep Neural Networks (DNN) is an improved type of artificial neural networks with a relatively large number of hidden layers. In the process of building DNN, after selecting the network architecture and preparing the associated training dataset, an iterative process is used in order to train the network [1]. In this process, the training data passes through all layers and non-linear functions of the network. The result obtained at the end of the network is normally different from the desired one. This difference determines the amount of error, and by back-propagating this error in the network and correcting the parameters, the network is updated. This process is repeated until the amount of error is minimized [2].

Deep learning has provided a bright future in various domains including agriculture. This emerging concept is known as smart farming. Smart farming has entered various fields of agriculture, and has received good feedback from the users. Smart farming can be used in cases such as 1) species management (species recognition and species breeding), 2) field condition management such as soil and water management, 3) crop management such as yield prediction, crop quality evaluation, disease detection, and weed detection [3].

One of the objectives of smart farming is to increase the quality and quantity of the agricultural products and achieve sustainable growth in productivity of production resources.

An early recognition of plant diseases is an important issue in smart farming. In this framework, a Convolutional Neural Network (CNN) has been designed by Liu *et al.* [4] for diagnosing apple leaf diseases. The design is based on the architectures of AlexNet [5] and GoogleNet [6], and leads to an accuracy of 97.62%. In a similar study by Yan *et al.* [7], some techniques such as global average pooling and Batch Normalization (BN) layers have been used, resultingd in an accuracy of 99.01%. The main drawbacks of these methods are that they study only the leaves of one type of plant (apple), and the adopted models have many parameters. In another study, DenseNet-121 has been used for identifying apple leaf diseases [8]. This study has achieved an accuracy of about 93% using the regression methods, multi-label classification, and loss function.

An important challenge in the study of plant leaf disease is separation of leaves from the image background, which has attracted considerable research attention. For example, in [9], soybean leaf area, edges, and defoliation have been found by Mahalanobis distance and Canny edge detection algorithm. Karlekar and Seal have suggested two modules. The first module extracts the leaf part from the whole image by subtracting the complex background. The second module introduces a deep learning convolution neural network (CNN), SoyNet, for soybean plant disease recognition using the segmented leaf image [10]. The "image database of plant disease symptoms", which involves 16 categories, was used in this study, where 98.14% identification accuracy was achieved. Kaur *et al.* have investigated soybean leaf diseases and identified healthy and diseased leaves using color and texture properties and a semi-automatic system based on the K-means rule [11]. This study was performed on the PlantVillage dataset, and led to an accuracy of about 90% on the training data, which is not very significant. In another study, the identification process of the corrosion areas in the leaf images has been strengthened by applying a color transformation and histogram equalization technique [12]. Also in [13], the Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) classifiers have been applied for detection and classification of plant disease. A feature extraction method based on an improved Local Binary Pattern (LBP) technique along with the Extreme Learning Machine (ELM) classifier have been proposed in [14] for recognition of the plant species.

Within the framework of neural networks, the scientists have found that adding more layers to a network not only makes training more difficult but also reduces the accuracy of network performance, and the network also suffers from the vanishing gradient problem. In order to solve this problem, ResNet was introduced in 2015. This network is a deep network that won the first place in the ILSVRC competition in 2015. The strength of the ResNet is its skip connections or additional residual connections. The shortcut connection goes through one or more layers and ignores them. It actually connects a layer to the farther layer. The proposed model also uses the shortcut connections.

In a review study [15], the performance of 9 different deep learning models on plant disease classification has been investigated. They performed the study based on two different approaches. In the first approach, using the transfer learning technique, the last three layers of the adopted networks were replaced with some other layers. In the second approach, which was faster and more accurate than the first one, the result of feature extraction in certain layers of models was taken, and fed to different machine learning classifiers. The highest reported accuracy is 97.45%. This result was obtained by extracting the features from ResNet101 with extreme learning machine classifier and 97.86% from Resnet-50 with an SVM classifier. Also an accuracy of 94.60% was achieved in transfer learning using small datasets with Resnet-50 [15].

The computational complexity and number of parameters are among the most important issues in deep neural networks that have a relatively complex architecture. In order to train such a network, an advanced hardware is required, which is not presumably accessible in all applications. On the other hand, for dealing with more complicated classification problems, a deeper architecture is usually required. Most recent studies in the field of diagnosing plant diseases using the image processing techniques have focused on one type of crop. However, the farmers usually cultivate a variety of crops, and in practice, such studies are not practically applicable. Also due to the fact that some plant diseases are rare, the plant disease datasets are commonly imbalance. Such an imbalanced dataset causes traditional classification networks that are based on cross-entropy loss function to not train and perform well [16]. Therefore, it is better to use modern networks such as deep convolutional networks, which have a structure consisting of several repeating blocks. In this framework, the operations such as Group Convolution (GConv), Pointwise Group Convolution (GConv(1×1)), and Depthwise Convolution (DWConv) are useful in order to extract more information from the associated training data. However, there are disadvantages for using these methods too. For example, if pointwise convolution is used in shallow networks with a small number of channels, the network accuracy is dropped. Using the stacked group convolutions stops the information flow between the channel groups, and weakens the representation. If the group convolution receives input from different groups, the input and output channels will be interdependent. In order to remove such problems, ShuffleNet makes use of the shuffling operation [17].

The other issue is the metric used for calculating the computational complexity of a network. Many networks use the indirect Float-Point Operations (FLOPs) metric for this purpose. According to [18], the FLOPs metric individually will not help us to succeed an optimal design. They used new approaches with two metrics, direct metric (speed), and indirect metric (FLOPs) to measure the computational complexity.

The number of channels is an important challenge in channel shuffling-based architectures. Better results can be achieved by using the interdependencies between the channels. In this framework, Squeeze and Excitation Network or SENet has been introduced by Hu, Shen [19]. They managed to improve the interdependency between the channels using this architecture with very little computational cost. In order to do this, some parameters were added to each channel of the convolution block so that the network could adjust the weights of each feature map adaptively. As a result, the shared low-level representations are strengthened and the informative features are emphasized.

In the proposed model of the present study, an attempt has been made to increase the network accuracy using SENet and recalibrating the channel-wise feature responses. For this purpose, the new AgriNet architecture inspired by ShuffleNetV2 (SHNet) and SENet is designed to diagnose and classify leaf diseases of the agricultural products. In order to increase the network accuracy, the channels with more information are first recalibrated. The channel shuffling process is then used in order to increase the information flow within the network. The proposed model identifies and classifies leaf diseases of the agricultural products successfully. The main objectives of designing AgriNet are increasing the model accuracy and reducing the network complexity. As a result, this technology can be used in general agricultural applications or in similar applications.

The main contributions of this research work can be summarized in two folds:

1- Technically:
   a. To increase the network accuracy and reduce the number of parameters; channels with more information are recalibrated at the end of each model block.
   b. To increase the flow of information in these channels; shuffling operations are then used.
   c. The network architecture is experimentally optimized, and it is shown that by repeating the network blocks appropriately, the number of parameters is significantly reduced, while a high level of accuracy is achieved.

2- Application:
   a. A relatively simple and accurate neural network is proposed for recognizing plant diseases.
   b. Most similar research works focus on one type of crop diseases, while in this work, 14 different types of crops with different diseases are taken into account.

2. Materials and Methods

In this section, you will first get acquainted with the PlantVillage dataset, and see some examples of the images in this dataset. You will then be introduced to some types of CNN networks such as the ShuffleNet, SHNet, and SENet networks. The proposed AgriNet architecture will then be discussed in the proposed architecture section.

2.1. Dataset

The PlantVillage dataset used in this work can be downloaded from https://www.kaggle.com. There are color images from 14 types of plants with different diseases in this collection. As we will mention later, the images are resized to 128 × 128 × 3. The dataset contains 54,305 images, of which 43,456 are used as the training data and 10,849 of them as the test data. Each folder in the dataset is named for a disease. Some of the images in this dataset are shown in Figure 1. Rangarajan Aravind, Maheswari [20] have published the complete information on various diseases of the agricultural products.

Table 1 shows the number of training and testing data in the dataset. The numbers in parentheses are the number of test data, and the rest are the training data. In this table, 14 types of plants are classified into 24 types of healthy and diseased. Some crops such as tomatoes and potatoes have the same disease. The dash indicates the absence of that instance in the dataset.

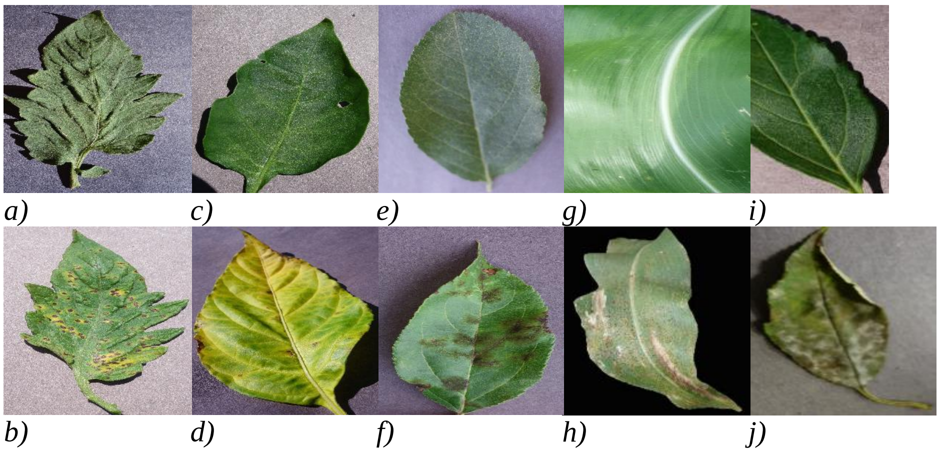


Figure 1. Some examples of pictures of plant diseases: a) Healthy tomato leaves, b) Tomato leaves with Septoria_leaf_spot disease, c) Healthy pear leaves, d) Pear leaves with bacterial spot disease, e) Healthy apple leaves, f) Apple leaves with disease Apple_scab, g) Healthy corn leaves, h) Corn leaves with Common_rust disease, i) Healthy cherry leaves, j) Cherry leaves with Powder_mildew disease.

Table 1. Number of training and test data in dataset.

Dataset / Leaf disease	Apple	Orange	Grape	Strawberry	Cherry	Peach	Raspberry	Blueberry	Potato	Pepper	Squash	Tomato	Soybean	Corn
Healthy	1316 (329)	-	339 (84)	365 (91)	684 (170)	288 (72)	297 (74)	1202 (300)	122 (30)	1183 (295)	-	1273 (318)	4072 (1018)	930 (232)
Apple_scab	504 (126)	-	-	-	-	-	-	-	-	-	-	-	-	-
Black_rot	497 (124)	-	944 (236)	-	-	-	-	-	-	-	-	-	-	-
Cedar_Apple_rust	220 (55)	-	-	-	-	-	-	-	-	-	-	-	-	-
Haunglongbing_(Citrus_greening)	-	4406 (1101)	-	-	-	-	-	-	-	-	-	-	-	-
Leaf_Blight_(Isariopsis_leaf_Spot)	-	-	861 (215)	-	-	-	-	-	-	-	-	-	-	-
Esca_(Black_Measles)	-	-	1107 (276)	-	-	-	-	-	-	-	-	-	-	-
Leaf_Scorch	-	-	-	888 (221)	-	-	-	-	-	-	-	-	-	-
Powder_mildew	-	-	-	-	842 (210)	-	-	-	-	-	1468 (376)	-	-	-
Bacterial_spot	-	-	-	-	-	1838 (459)	-	-	-	-	-	1702 (425)	-	-
Late_Blight	-	-	-	-	-	-	-	-	800 (200)	-	-	1528 (381)	-	-
Early_Blight	-	-	-	-	-	-	-	-	800 (200)	-	-	800 (200)	-	-
Bell_Bacterial_spot	-	-	-	-	-	-	-	-	-	798 (199)	-	-	-	-
Leaf_Mold	-	-	-	-	-	-	-	-	-	-	-	762 (190)	-	-
Septoria_Leaf_Spot	-	-	-	-	-	-	-	-	-	-	-	1417 (354)	-	-
Spider_Mites Two_Spotted_Spider_Mite	-	-	-	-	-	-	-	-	-	-	-	1341 (335)	-	-
Target_Spot	-	-	-	-	-	-	-	-	-	-	-	1124 (280)	-	-
Yellow_Leaf_Crul_ Virus	-	-	-	-	-	-	-	-	-	-	-	4286 (1071)	-	-
Mosaic_Virus	-	-	-	-	-	-	-	-	-	-	-	299 (74)	-	-
Cerospora_leaf_spot Gray_leaf_spot	-	-	-	-	-	-	-	-	-	-	-	-	-	411 (102)
Common_rust	-	-	-	-	-	-	-	-	-	-	-	-	-	954 (238)
Northern_Leaf_Blight	-	-	-	-	-	-	-	-	-	-	-	-	-	788 (197)

2.1. Convolutional neural networks

The convolutional neural networks are created by stacking of a number of different layers. Each layer has its own task. The most important layers are the convolution, pooling, and the Fully Connected (FC) layers. The convolution layer consists of a set of learnable filters. A pooling layer is usually used after each convolution layer in order to reduce the size of the resulting feature map, and therefore, the number of network parameters. In the pooling operations, similar to what happens in convolutions, a window is moved on the image. If the max pooling operation is considered, each window retains the maximum value and discards the rest but in Average (AVG) pooling, the average of the calculated values is used in the output. For example, if a max pooling with size 2 × 2 and stride 2 is applied to a feature map with size 8 × 8, a 4 × 4 output will be obtained. In a CNN, the last layer is usually a FC layer that represents the network output as a vector for categorizing the input images [21].

2.2.1. ShuffleNet architecture

In the deep neural networks, modern architectures such as Xception [22] and L2MXception [23] exploit repetition of one or more specific blocks. An architecture designed using this technique is ShuffleNet. In ShuffleNet, the group convolution is used instead of the point convolution in order to improve the network accuracy. By stacking group convolutions, the outputs of a certain group depend on the inputs within the group, and the information flow between the channels from different groups is stopped. In ShuffleNet, this problem is removed by the shuffling operation, which obtains the input data from different groups. The shuffling process makes the input and output channels completely interdependent. As one can see in Figure 2, ShuffleNet uses the operations such as depthwise convolution, pointwise group convolution, and shuffeling operations in order to reduce the computational cost. In the unit with stride = 2, in order to be able to connect the output of the main path and the shortcut, the AVG pooling technique is used with stride = 2 [17].

In the main path of the ShuffleNet units, the input channels first pass through a GConv layer. In the GConv, each kernel looks at only a few input feature maps, and combines some information across the channels. In this architecture, GConv is used for reducing the computational complexity of 1 × 1 convolution. You can see the pointwise group convolution in Figure 3.

The GConv output is normalized using the BN technique and passed through the activation function of Rectified Linear Unit (ReLU). The reason for using BN is that during the learning process, using the gradient descent algorithm, the network weights change after each step, and consequently, the shape of the data that enters the next layer changes. BN causes the next layer to receive the data as expected (similar to the previous step). As a result, higher learning rates and less dropout can be used. For the normalization purpose, suppose $x = (x^{(1)}, x^{(2)}, \dots x^{(d)})$ is a d-dimensional data, each dimension is normalized by (1) [24].

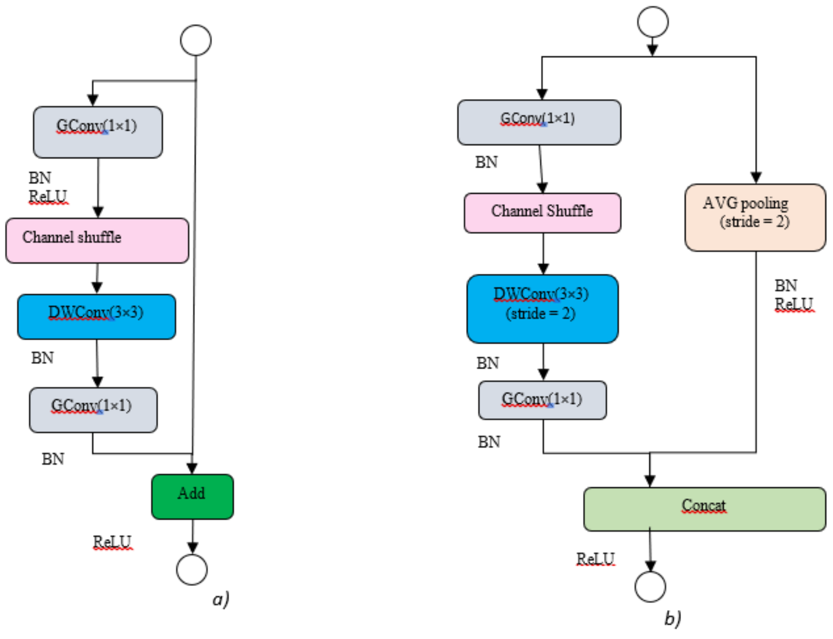


Figure 2. ShuffleNet architecture a) ShuffleNet unit; b) ShuffleNet unit with stride = 2 [17].

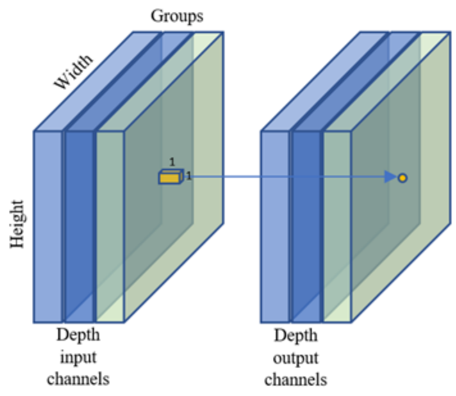


Figure 3. Pointwise group convolution.

In (1), E and Var are per-dimension mean and variance, respectively.
After BN, the ReLU activation function is used (2). This activation functions decide whether a particular neuron is active or inactive.

$$\hat{x}^{(k)} = \frac{x^{(k)} - E\left[x^k\right]}{\sqrt{Var\left[x^k\right]}} \tag{1}$$

$$ReLU\left(x\right) = \begin{cases} 0 & if \;\; x \le 0 \\ x & otherwise \end{cases} \tag{2}$$

In (2), x is the input of the function, and its output is zero for negative numbers and x for positive numbers.

The channels are then shuffled based on the shuffling operation. In this operation, the channel indices are changed, which occurs by re-shaping the related tensor and calculating its transposition. The output of the shuffling operation enters a DWConv. In this type of convolution, a 2D depth filter is applied to each depth level of input tensor and the kernel has different spatial dimensions but only one channel. The depthwise convolution keeps each channel separate. You can see DWConv in Figure 4.

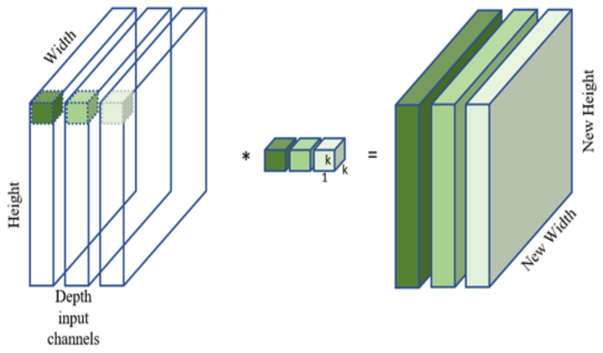


Figure 4. Depthwise convolution.

The output DWConv enters GConv to recover the channel dimension to match the shortcut path. The difference between the two units of this architecture is that in the unit with stride = 2 used AVG pooling in the shortcut path and channel concatenation to connect outputs. The purpose of concatenation is to make it easy to enlarge channel dimension with little extra computation cost.

2.2.2. ShuffleNetV2 (SHNet) architecture

In ShuffleNetV2, known in the present work as SHNet, four guidelines were adopted for designing an efficient network. An indirect metric used to calculate the computational complexity is the number of floating-point operations per second or FLOPs. This metric is not equivalent to a direct metric such as speed or delay. It has been proven that some networks with the same FLOPs have different speeds [18]. In the research work of [18], it has been stated that the use of FLOPs only as a metric of computational complexity damages the optimal network design. There is a difference between the indirect metrics (FLOPs) and the direct metrics (speeds) for two reasons. First, a factor such as the Memory Access Cost (MAC), which affects speed, is not considered by FLOPs. Secondly, the runtime in models with the same FLOPs is directly dependent on the type of hardware used [18].

These four guidelines are: first, equal channel width reduces MAC; secondly, use of group convolutions can increase MAC. In this guideline, it has been proven that as the number of group increases, the speed of network decreases. Thus it is important to pay attention to the choice of the number of groups in the model. For example, in the study of [17], the model based on group convolution did not consider this guideline; thirdly, if the network is fragmented, the degree of parallelism and the efficiency of the network is reduced. One of the architectures that contradicts this guideline is the architecture proposed in [18]; fourthly, since in GPU much time is spent on performing element-wise operators such as ReLU, AddTensor, AddBias, and even depthwise separable convolution, care must be taken in their use [18]. By following the above four guidelines, two units were designed, as shown in Figure 5.
In explaining the operations performed in each one of the blocks or units, it can be stated that: in block1, which is SHNet unit with stride = 2, there are two paths, main and shortcut. In the main path,

after applying Conv (1×1), BN and ReLU activation, a DWConv is used. BN is performed again, and the output of Conv (1×1) is concatenated to the output of the shortcut path after applying BN and ReLU. The number of input and output channels per convolution in the main path is equal. In the shortcut path, a DWConv is applied directly to the input, and the continuation of the path is the same as the main path. In the main path of block1, unlike ShuffleNet, the pointwise group convolution is not used because it increases MAC. In order to better understand this, pay attention to (3) and (4), which show the relations between MAC and FLOPs for the 1 × 1 group convolution [18].

$$MAC = hw\left(c_1+c_2\right)+\frac{c_1c_2}{g} = hwc_1+\frac{Bg}{c_1}+\frac{B}{hw} \qquad (3)$$

$$B=\frac{hwc_1c_2}{g} \qquad (4)$$

In (3) and (4), g is the number of group. The numbers of input and output channel are c_1 and c_2 respectively.

h and w are the spatial size of the feature map, and B is FLOPs. According to these equations, if the input shape ($c_1 \times h \times w$) and B are constant and only increase the number of groups, MAC will also increase [18].

Also the channel shuffling operation is eliminated. The shuffling operation is applied to the output obtained by concatenating these two paths finally.

In block2, using the split operation, the number of input feature channels is divided into the c and $c-c'$ channels in each path. If c is considered half of c', in each block, half of the feature channels are inserted directly into the next block, which reuses the features in this way. The main paths in block1 and block2 are exactly the same, and the output of the shortcut path is directly concatenated to the output of the main path.

In summary, the ShuffleNet architecture is one of the shallow architectures that has reached its ultimate goal based on the feature channels in the image, and this architecture is used in devices such as mobile phones. The problem of limiting the number of feature channels in shallow networks was fixed in the first version of ShuffleNet. In the ShuffleNet architecture, in order to raise the number of feature channels without changing the number of FLOPs, two methods of GConv and ShuffleNet units are used. Also the shuffling method is used to create information interdependence between the groups of channels and enhance the network accuracy.

SHNet considers ShuffleNet inefficient. The problems that SHNet raises for the first version of ShuffleNet are [18]:

1- By increasing the number of feature channels with point-wise group convolutions and bottleneck structures such as ShuffleNet units, the MAC value increases, which cannot be ignored for the light-weight models.
2- Reducing the network parallelism due to the large number of groups will have a negative effect on the accuracy and efficiency of the network.
3- Use of add operations increases the training time.
4- Information between different groups in this network is lost.

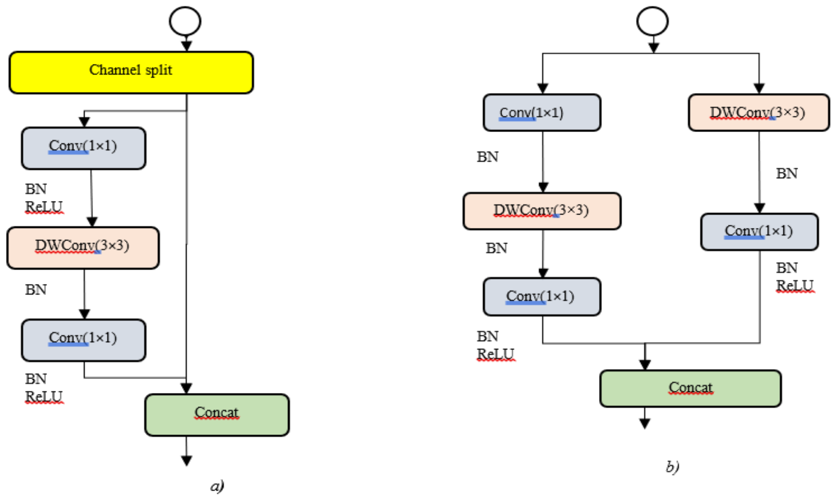


Figure 5. SHNet architecture a) SHNet unit or block2; b) SHNet unit with stride = 2 or block1 [18].

2.2.3. SENet architecture

In the study of [19] and based on the idea of explicit modeling of interdependencies between channels, the SENet architecture has been proposed. The nets are better able to map the channel dependency along with access to global information and recalibrate the filter outputs by SENet block. Finally, it improves the performance.

The SENet block shown in Figure 6. was used for this purpose. This block consists of three sections: squeeze module, excitation module, and scale module. In this block, input $X \in R^{H' \times W' \times C'}$ is mapped to feature maps $U \in R^{H \times W \times C}$ by convolution operator F_{tr}.

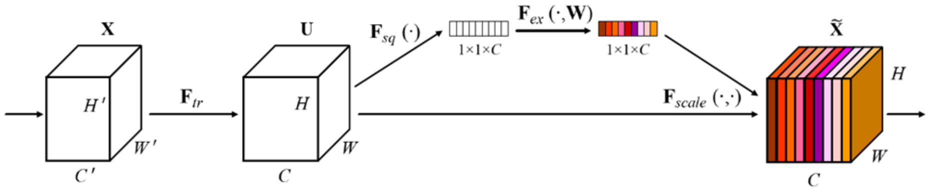


Figure 6. Squeeze and Excitation block in SENet architecture [19].

In (5), $V = [v_1, v_2, ...v_C]$ is the set of learned filter kernels, where v_c refers to the parameters of the c-th filter, and its output is $U = [u_1, u_2, ...u_C]$.

$$u_C = v_C * X = \sum_{S=1}^{C'} v_C^S * x^S \tag{5}$$

$$v_C = \left[v_c^1, v_c^2, ...v_c^{C'}\right] \tag{6}$$

$$X = \left[x^1, x^2, ..., x^{C'}\right] \tag{7}$$

$$u_c \in R^{H \times W} \tag{8}$$

In (5), * refers to the convolution operation. $\boldsymbol{v}_c^s$ denotes a 2D spatial kernel that represents a single channel $\boldsymbol{v}_C$ operating on the corresponding of channel X. In order to create a channel descriptor, the U features are passed through the squeeze operation, and the feature maps are aggregated in their spatial dimensions ($H \times W$). Once the feature maps are aggregated, the excitation operation is performed, which is a simple self-gating mechanism. By stacking the squeeze and excitation blocks, SENet is created. The notable point about this network is its low computational cost [19].

The squeeze operation in SENet uses a global averaging pooling. Compressing the global spatial information into channel descriptors by an average global pooling solves the problem of channel dependence. This pooling layer transforms U into Z, where the feature maps of C become $1\times1\times C$, for example, the c-th component of Z is calculated in (9).

$$z_C = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j) \tag{9}$$

where z_c refers to the output of the squeeze operation, $H \times W$ indicates the spatial dimension, and u refers to the feature map. The excitation operation is performed on the output of the squeeze operation. This operation is indeed a sigmoid activator that aims to use the learning parameters to model the interdependence between the feature channels and generate the weight of each feature channel. In order to limit the complexity of the model, two FC layers have been used between these two operations [19]. Figure 7 displays a block diagram of the SENet block.

The input dimensions to the SENet block are $H \times W \times C$. This input passes through the global pooling, and changes its dimensions to $1\times1\times C$. Via this technique, each channel is compressed to a single numeric value. Using a FC layer after the pooling, the number of channels is reduced to $\frac{C}{r}$ and then passed through a ReLU activation. Next, using another FC layer, the number of channels is returned to the primary value. Finally, the output of the sigmoid function is scaled with the input, and the output is created with the same initial dimensions as the primary input. These steps add a computational cost of less than 1% to each network. So far, SENet has been used in different convolutional networks [19].

In short, we can say about this architecture: 1- It has used a convolution block as an input. 2- In the squeeze module, each channel is squeezed into a single value using the average pooling. 3- In order to reduce the complexity of the output channels, a dense layer followed by a ReLU adds non-linearity. 4- Another FC layer followed by a sigmoid gives each channel a smooth gating function. 5- Finally, it weights each feature map of the convolutional block based on the side network, the "excitation".

2.3. Proposed architecture

In any image classification task, the pre-processing operations are usually required in order to achieve better results and reduce the effects of noise and distortions. CNN uses less pre-processing than the other classification methods. This is mainly due to the fact that the filtering processes are automatically performed in the convolution layers of CNNs. However, within the CNN framework, a required pre-processing step is to resize the input images to the same size. In the proposed architecture, the input images are resized to 128 x 128. Our experiments show that using such an image size improves the network's ability to capture the images information, while the computational complexity is restricted. Another pre-processing step of the proposed model is normalization for rescaling the pixel values. This puts the pixel values in a certain range.

Inspired by the lightweight architectures with very low computational complexity such as SENet and SHNet this work aims to preserve the benefits of these architectures such as emphasizing on the informative features and suppressing the less useful ones, and to reduce the number of network parameters in order to achieve a higher accuracy than the previous architectures.

The block diagram of the proposed model can be seen in Figure 8. It is observed that once the network input with dimensions of 128 × 128 × 3 has passed through the convolution layer and global maximum pooling, it goes through three main stages. Each one of these stages involves a combination of block1 and block2, previously shown in Figure 5. Finally, the last four layers of this architecture are convolution, global maximum pooling, FC, and Softmax.

the network outputs for classification, and the sum of its outputs is 1. Indeed, the Softmax output indicates the possibility of correctly classifying each one of the classes. Equation (10) expresses the Softmax activation function.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}} \tag{10}$$

where z is a vector of the inputs to the output layer (for example, if z has 5 elements, the output will be 5 units), i = 1, 2... K is the output units. The z_i values are the elements of the input vector to the output layer. The standard exponential function is e.

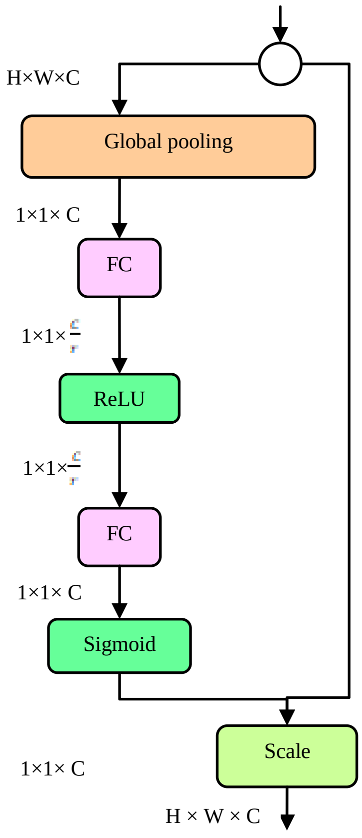


Figure 7. Block diagram of SENet [19].

The Softmax activation function is usually used at The difference between the two proposed architectures AgriNet131 and AgriNet373 is in the number of repetitions of the blocks in each one of the stages. For example, Figure 9 reveals a block diagram for the AgriNet131 model. In this model, in stage 1 and stage 3 after block 1, block 2 is located, while in stage 2 after block 1, block 2 is repeated three times. If the AgriNet373 architecture is considered, in each one of the stages after block 1, block 2 is repeated three times, seven times, and again three times, respectively, and the continuation of the process is similar to the AgriNet131 model. Due to the large architecture of AgriNet373, its block diagram is not shown.

applying BN and ReLU. The number of input and output channels per convolution in the main path is equal. In the shortcut path, a DWConv is directly applied on the input, and the continuation of the path is the same as the main path. In each one of the stages, after concatenating the outputs of the main and shortcut paths, the informative features are emphasized, and the less useful features are suppressed using SENet. Then the channels will be shuffled using the channel shuffling operations. Finally, after the completion of stage 3, as seen in Figure 10, the output (O) passes through the final 4 layers (convolution, global maximum pooling, FC, and SoftMax) and the input image is classified.

Section SHNet architecture describes the internal structure of each block.

Figure 10 shows the details of AgriNet131. It is observed that the network input has passed through the convolution layer and global maximum pooling, going through stages 1, 2, and 3. Each one of these stages involves a combination of blocks 1 and 2.

Each one of these blocks consists of the components such as convolution, Depthwise convolution, batch normalization, and ReLU. As shown in the figure, for example, there are two main and shortcut paths in block 1. In the main path, after applying Conv (1 × 1), BN, and ReLU activation, a DWConv is used. BN is performed again, and the output of Conv (1 × 1) is concatenated to the output of the shortcut path after applying BN and ReLU. The number of input and output channels per convolution in the main path is equal. In the shortcut path, a DWConv is directly applied on the input, and the continuation of the path is the same as the main path. In each one of the stages, after concatenating the outputs of the main and shortcut paths, the informative features are emphasized, and the less useful features are suppressed using SENet. Then the channels will be shuffled using the channel shuffling operations. Finally, after the completion of stage 3, as seen in Figure 10, the output (O) passes through the final 4 layers (convolution, global maximum pooling, FC, and SoftMax) and the input image is classified.

2.4. Evaluation metrics

In this section, the metrics used for evaluating the performance of the proposed method are introduced.

Accuracy: The most important metric for determining the performance of a classification algorithm is the classification accuracy or rate. This criterion calculates the total accuracy of a classifier. Indeed, this criterion is the most famous and general metric for calculating the performance of the classification algorithms, which shows that whether a model is being trained correctly and how it may perform generally [25].

$$Accuracy=\frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made} \tag{11}$$

Loss: One of the most important goals in designing a network is to enhance the forecasting accuracy, which is calculated by a cost function. This function fines the network when it makes a mistake. The best output occurs when there is the minimum cost, and the optimization algorithms are used to achieve it. The optimization algorithm, based on the cost function and data, determines how the network weights are updated to optimize the network. One of the appropriate algorithms that adjusts the learning rate during the training process is the Adam algorithm (Adam: a method for stochastic optimization) [25].

$$Precision=\frac{True\ Positives}{True\ Positives+False\ Positives} \tag{12}$$

Recall: The maximum value of this criterion is one or 100%, and the minimum value is zero. The recall is the ratio of the correct positive predictions to all the observations in the real class. This will be a good metric when the False Negative value is high [25].

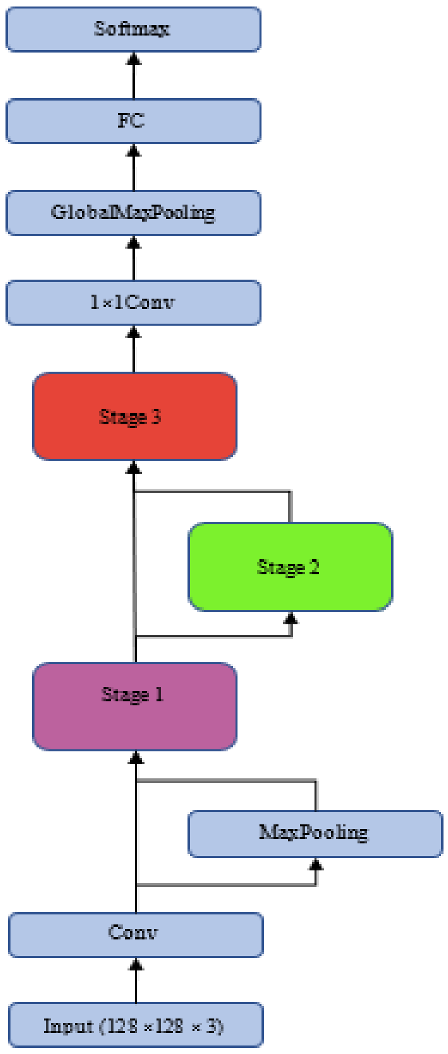


Figure 8. General structure of AgriNet architecture.

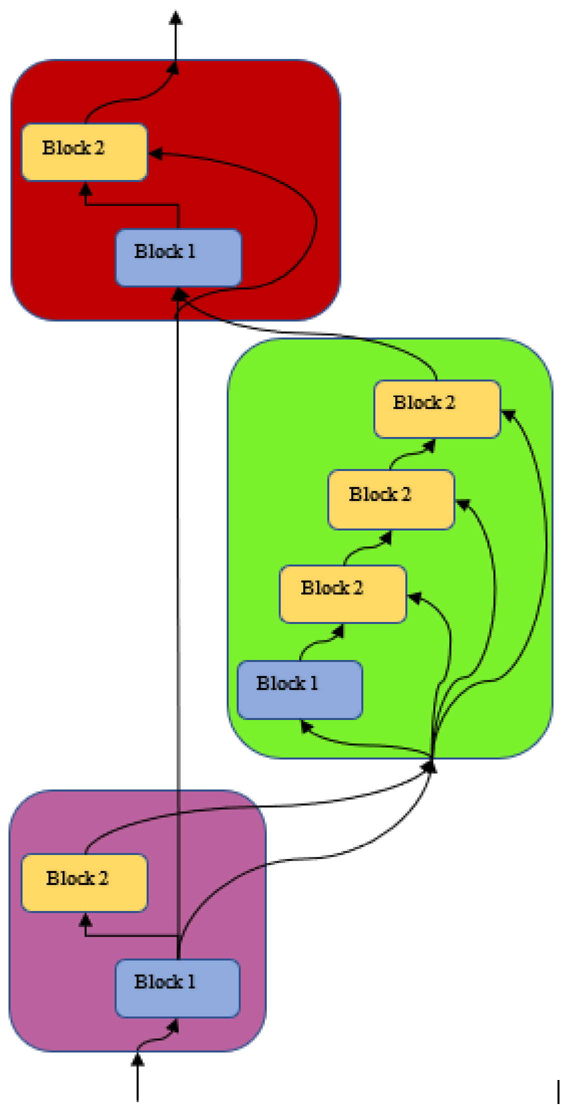


Figure 9. stage 1, stage 2, and stage 3 in AgriNet131 model.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negative} \quad (13)$$

F1_Score: One of the best criteria for evaluating the accuracy of a test is F1_Score. This criterion is 1 in the best case and 0 in the worst case, and is calculated based on the precision and recall [25].

$$F1\_score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (14)$$

2.5. Equipment

This work was developed using the Python's high-level interpretive programming language. Pycharm programming environment and Tensorflow-Keras framework have been used in all stages of the work. The GeForce graphics card used was the GTX1080 with 2560 CUDA cores and 8GB of memory. The computational time using this equipment will be discussed in the next section.

3. Experimental Results and Discussion

In this section, the details of our experimental study and the results are presented.

3.1. Experimental setting

When designing a DNN, consideration should be given to selecting the values such as the number of layers, number of neurons in each layer, number of epochs, and size of the batch used. Finding the right values for the mentioned cases is experimental, and should be selected in a way that it does not adversely affect the accuracy and speed. For example, if the number of hidden layers of the network is large, the network performance is better but the speed of the network is affected.

3.2. Parameter setting

In order to understand the two concepts of number of epochs and batch size, the concept of gradient descent should be introduced. This concept is generally used to optimize the network, which is applied to find the best response based on iteration. The gradient descent has a parameter called the learning rate. At the beginning, the steps are larger and the learning rate is higher, and as the steps shrink, so does the learning rate. By gradient descent, the cost is also reduced and the model is optimized [26].

In this work, in order to accelerate convergence, the learning rate is controlled, and if no improvement is seen in the convergence process after three epochs, the learning rate is automatically reduced. The new learning rate is obtained through multiplying the initial learning rate by 0.5. The lowest learning rate in the proposed architecture is 0.00001. At the end of an epoch, the entire training dataset is sent to the network once. Thus an epoch is a very large process, and it is better to break it into smaller batches. In this way, all data can be sent to the network several times, and the appropriate weights in the network can be obtained. The hyper-parameters in the proposed architecture are given in Table 2. The network is trained using the training set for 50 epochs with a batch size of 128 and compress rate 16 (all numbers were obtained experimentally using the validation dataset).

Table 2. Hyper-parameters of proposed model.

Parameters	Setting
Batch size	128
Image size	(128,128,3)
Optimization	Adam
Epoch	50
Compress rate	16

3.3. Model training

Use of all available data to train the network does not seem logical. It is best to divide the available dataset into two parts: training and validation. In this work, 70% of the available images are considered for the network training and the remaining 30% are considered as the validation sets.

One of the criteria for evaluating an architecture is accuracy. Accuracy refers to how close the value predicted by a machine algorithm is to its true value. The network performance is directly related to the accuracy criteria, and a network with a higher accuracy has a better performance. The right images of Figures 11 to 14 show a reduction in loss, and the left images show an increase in accuracy in 50 epochs. As the model learns, the loss successfully decreases, while the accuracy increases.

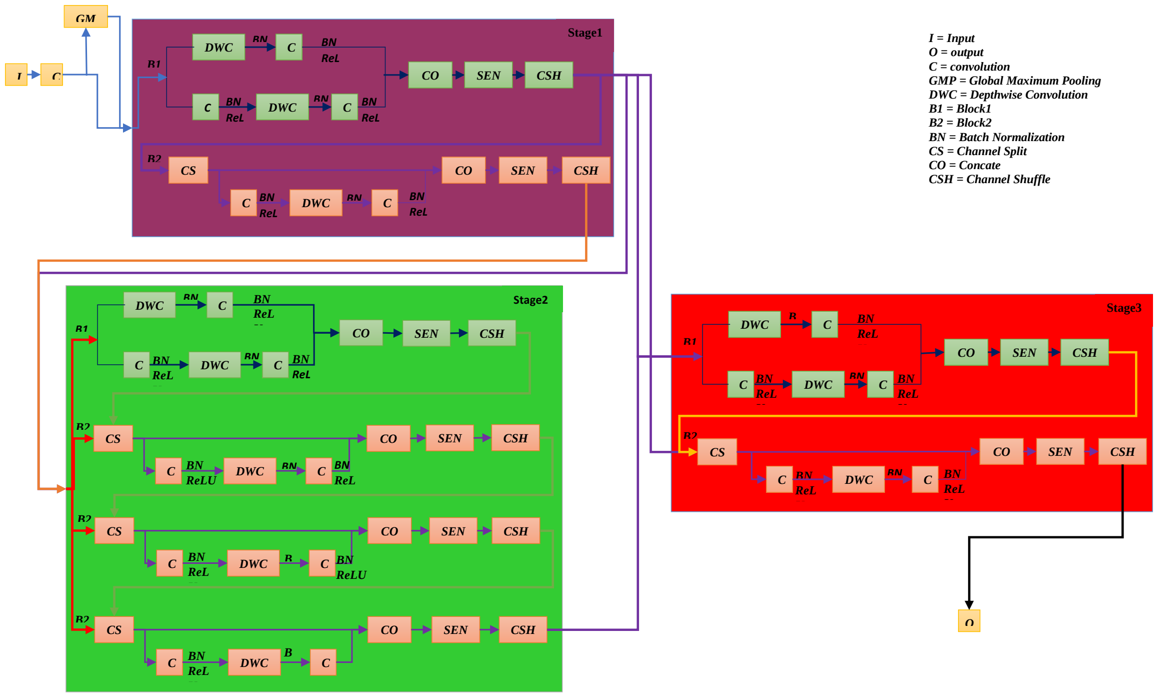


Figure 10. Figure with more details of Model AgriNet131.

Figure 11 depicts the accuracy and loss diagrams in both training and validation in the AgriNet373 and SHNet373 networks. The training accuracy and loss in both networks has the best value, 1 and 0, respectively, while in validation, AgriNet373 performed better.

Figure 12 indicates the loss and accuracy curves of the two architectures AgriNet131 and SHNet131 in the two phases of training and validation. As it can be seen in the figures, both networks have reached accuracy 1 in the training phase, and are very close in the validation phase. Also the loss figures are very close in both phases.

The two curves related to the proposed architecture are compared in Figure 13. In the initial model (AgriNet373) and reduced parameters model (AgriNet131), it could be seen that the loss in both networks as well as in the training and validation phase reached zero after approximately 20 epochs, and remained unchanged. Note that the accuracy of the validation phase was better in the AgriNet131 architecture than in the AgriNet373

When comparing the two curves related to the SHNet networks in the two initial model (SHNet373) and parameter reduction model (Figure 14), it is observed that the lowest and highest loss in the validation phase are related to SHNet131 and SHNet373, respectively. Concerning accuracy in the validation phase, the highest value is related to the SHNet131 states. In the training phase, both networks operate almost identically.

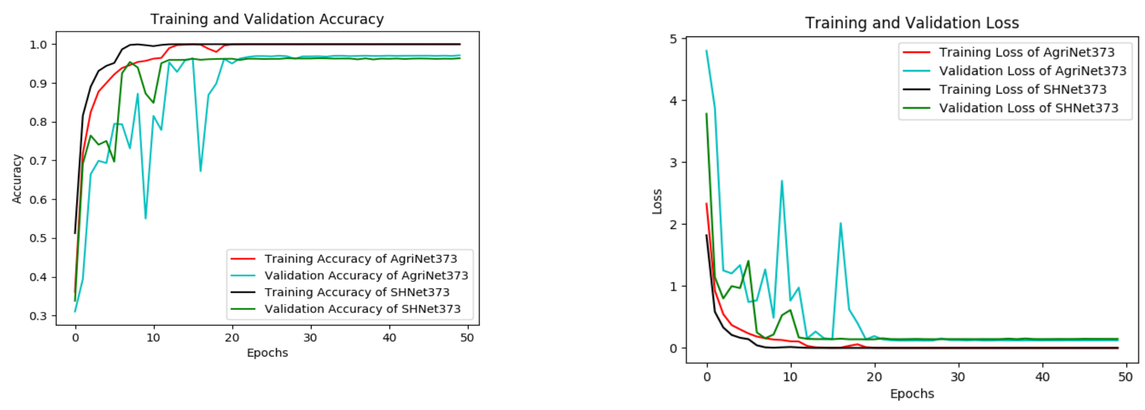


Figure 11. Comparison of accuracy and loss in AgriNet373 and SHNet373 architecture.

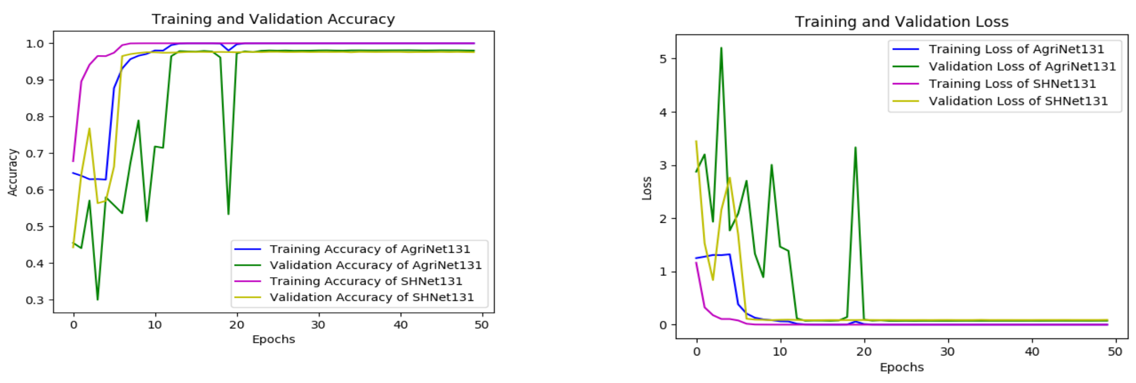


Figure 12. Comparison of accuracy and loss in AgriNet131 and SHNet131 architecture.

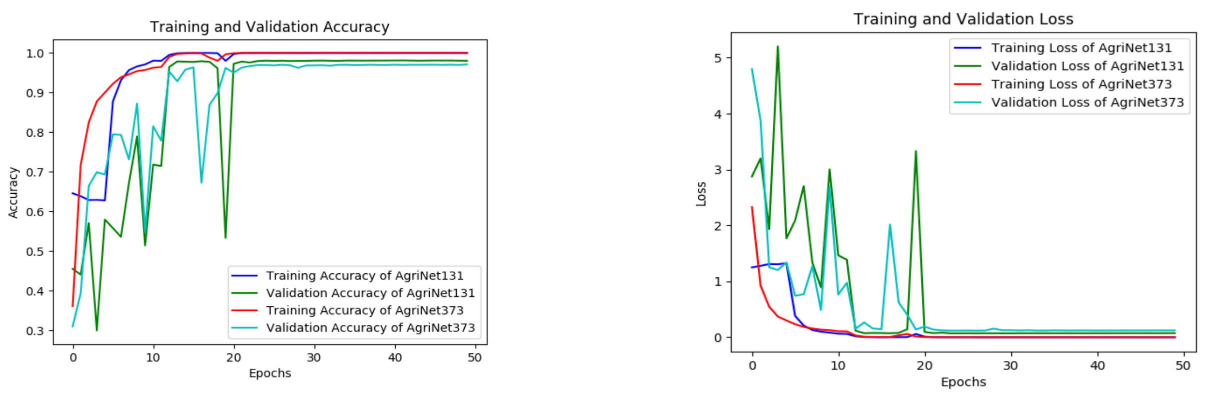


Figure 13. Comparison of accuracy and loss in AgriNet373 and AgriNet131 architectures.

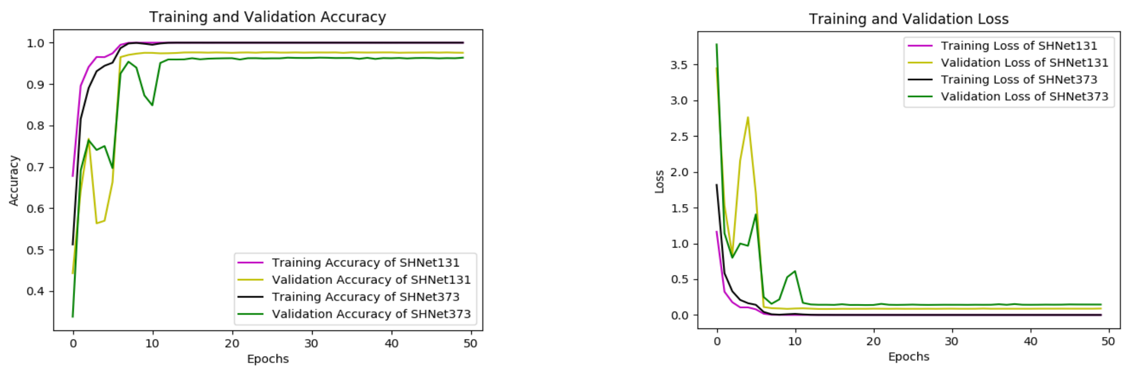


Figure 14. Comparison of accuracy and loss in SHNet373 and SHNet131 architectures.

In all of the above curves except Figure 14, after about 20 epochs, the amount of loss and the amount of accuracy have reached their minimum and maximum values, respectively, and have remained almost unchanged. In Figure 14, this situation occurred in about 10 epochs.

Table 3 reports the Accuracy, F1_score, Loss, Precision, and Recall in the training, validation, and testing phases in SHNet373, SHNet131, AgriNet373, and AgreNet131 architectures.

All the experiments were performed on "PlantVillage" with 14 types of plants in 24 healthy and diseased groups.

As it can be seen in Table 3, in the training phase, the best performance in term of accuracy is 1 that belongs to all models. This indicates that the network is well-trained. In the validation and test phase, the AgriNet131 network has shown the best performance in terms of accuracy. This indicates that the proposed model (with the lowest repetition rate in blocks) is able to classify the leaf disease images with a very low error rate. As a result, in the proposed architecture, by reducing the number of iterations of units, in addition to reducing the number of parameters from 4051690 in AgriNet373 to 2661978 in AgriNet131 (the number of parameters is almost halved), accuracy has increased from 0.97 to 0.98 in the test and validation phase. Since the difference in the accuracy of the two test phases in the AgriNet373 and AgriNet131 models is very small, it is preferable to use the AgriNet131 model with fewer parameters. In the training phase, reducing the number of blocks, and consequently, the number of parameters did not change the accuracy of the network. On the other hand, in the validation and test phase, it improved the accuracy of AgriNet and SHNet by 1%. The best F1_Score has been achieved in the validation and testing phase of all models. The lowest loss among the models and in all phases belongs to the SHNet131 model. The best precision has been achieved in the validation and testing phase of all models. The recall values are all acceptable in all cases.

Table3. Accuracy, F1_score, Loss, Precision, and Recall in training, validation, and testing phases in SHNet373, SHNet131, AgriNet373, and AgreNet131 architectures.

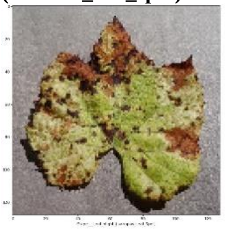
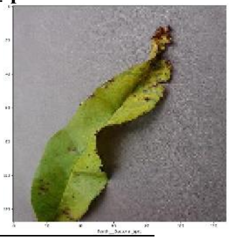
SHNet373 50 epochs	Accuracy	F1_Score	Loss	Precision	Recall
Train	**1**	0.994	$6.676e^{-5}$	0.988	**1**
Validation	0.963	**1**	0.145	**1**	**1**
Test	0.962	**1**	0.153	**1**	**1**
SHNet131 50 epochs	Accuracy	F1_Score	Loss	Precision	Recall
Train	**1**	0.944	$\mathbf{6.093e^{-5}}$	0.988	**1**
Validation	0.975	**1**	0.090	**1**	**1**
Test	0.976	**1**	0.091	**1**	**1**
AgriNet131 50 epochs	Accuracy	F1_Score	Loss	Precision	Recall
Train	**1**	0.994	0.0006	0.988	**1**
Validation	0.980	**1**	0.074	**1**	**1**
Test	0.980	**1**	0.075	**1**	**1**
AgriNet373 50 epochs	Accuracy	F1_Score	Loss	Precision	Recall
Train	**1**	0.994	0.0001	0.988	**1**
Validation	0.971	**1**	0.122	**1**	**1**
Test	0.979	**1**	0.125	**1**	**1**

Table 4 shows examples of the network performance on healthy and diseased crops. Each column mentions the name of the product and the type of leaf disease related to it; in each row of the table, the probability of belonging to the class in the architecture and the name of the class predicted by the network are also mentioned.

It is shows that in the diagnosis of healthy grape leaves, all architectures have correctly identified the leaf class, and the AgriNet architecture outperformed the SHNet. On the other hand, the healthy peach leaf in the AgriNet131 architecture is mistakenly known as the leaf with Peach_bacterial_spot disease. The reason for this misclassification can be attributed to the very similarity of Peach_bacterial_spot to Peach_health. The architecture has also worked well for defective peach leaves. In general, except for one error in other cases, the proposed network can assign the images to the correct class with the highest probability. This shows the high accuracy and efficiency of the proposed model.

Table 5 reports the accuracy and hardware of the networks that used the PlantVillage dataset. Three studies [27], [28] and [29] were limited to the tomato products from this dataset. They classified tomatoes into 10 different classes including healthy tomato leaves and 9 diseased leaf samples. The highest accuracy in these three studies was 98.4%. Only in the present study and the study [30], all 38 classes in the PlantVillage dataset are covered. In [30], two networks, AlexNet and GoogleNet, are used in two modes: transfer learning and training from scratch. The AlexNet architecture is categorized as shallow architectures. The GoogleNet architecture with 9 Inception modules has only 5 million parameters, which is 12 times less than the number of parameters that can be learned in AlexNet. However, the proposed architecture has only 2661978 learnable parameters. Given the size of the network and the limited number of learnable parameters, the accuracy obtained from the proposed network has been acceptable.

Table 4. Examples of results obtained from AgriNet373, AgriNet131, SHNet131, and SHNet373 architectures.

Sample name	**Grape-healthy**	**Grape_leaf_blight (isariosis_leaf_spot)**	**Peach_healthy**	**Peach_bacterial_spot**
Sample image				
Possibility of belonging to the desired class in AgriNet373 architecture/ Network predicted class	0.99 Grape-healthy	**1** Grape_leaf_blight (isariosis_leaf_spot)	0.99 Peach_healthy	0.99 Peach_bacterial_spot
Possibility of belonging to the desired class in AgriNet131 architecture/ Network predicted class	0.99 Grape-healthy	0.99 Grape_leaf_blight (isariosis_leaf_spot)	0.92 Peach_bacterial_spot	0.99 Peach_bacterial_spot
Possibility of belonging to the desired class in SHNet131 architecture/ Network predicted class	0.91 Grape-healthy	**1** Grape_leaf_blight (isariosis_Leaf_spot)	0.99 Peach_healthy	0.99 Peach_bacterial_spot
Possibility of belonging to the desired class in SHNet373 architecture/ Network predicted class	0.98 Grape-healthy	0.99 Grape_leaf_blight (isariosis_leaf_spot)	0.97 Peach_healthy	0.99 Peach_bacterial_spot

Due to the fact that SHNet is 40% faster than ShuffleNetV1, the runtime of the proposed model is very close to ShuffleNetV2. The network training time on our GPU and with the same batch size is 9739 s for the SHNet and 9856 for AgriNet. The test time in the proposed model is about 15 ms (for all test samples), while it is about 18 ms in the SHNet. We believe that the small additional computational cost incurred by AgriNet is justified by its contribution to model performance. In order to compare the hardware used in the present study and [30], which is in the 4th row, some points should be noted. There are two types of technologies in the manufacturing of Nvidia graphics card: CUDA Cores technology and Tensor Cores. The CUDA technology allows hundreds of separate Nvidia graphics chips to process the data in parallel. The tensor technology is designed to accelerate the learning capabilities of an artificial intelligence system. This technology is 47 times faster than a CPU-based server and 12 times more powerful than Nvidia's previous GPU products [31]. Due to the very strong technology used in the study of [30], the results obtained in this study are remarkable.

Challenges that may be faced using the new test data are:

1- The biggest challenge for a big dataset is hardware constraints. How to store and import such a volume of data into the model also has its own problems.
2- If the dataset is small, in order to achieve acceptable results, the number of input data should be increased in different ways such as data augmentation.
3- In the datasets that have a more crowded background, more initial pre-processing is required.

In order to fill the research gaps, a new method has been proposed for automatic detection as well as classification of plant leaf diseases using AgriNet. The advantages of the proposed algorithm are as follow:

1- Recalibration has been used to increase the network accuracy.
2- In order to increase the information flow in the network, the shuffling process has been used.
3- In addition to reducing the number of parameters, the complexity of the network has been reduced.

Table 5. Comparison of accuracy and hardware of networks working on the PlantVillage dataset.

Number	Method or author	Dataset	System specifications	Number of images	Number of crops reviewed	crop	Number of classes	Test accuracy
1	Mohammed Brahimi, Kamel Boukhalfa, Abdelouahab Moussaoui [27]	PlantVillage	GPU: Quadro K 5000 4GB, 1536 cores 128 GB RAM	14,828	1	Tomato	10	AlexNe without pre-training: 97.35% GoogleNet without pre-training: 97.71%
2	Mohit Agarwal, Suneet Kr. Gupta, K.K. Biswas [28]	PlantVillage	GPU: NVIDIA DGX v. 100 40600 CUDA cores 128 GB RAM	18160	1	tomato	10	**98.4%**
3	Halil Durmuú, Ece Olcay Güneú, Mürvet KÕrcÕ [29]	PlantVillage	GPU: Nvidia Jetson Tx1 256 CUDA cores, 4GB RAM,	-	1	tomato	10	AlexNet: 95.65% SqueezeNet: 94.3%
4	Sharada P. Mohanty, David P. Hughes, Marcel Salathé [30]	PlantVillage	GPU: NVIDIA Tesla V100 640 tensor cores 32GB RAM	54306	14	Apple,Blueberry Cherry,Corn Grape,Orange Peach,Bell Pepper Potato,Raspberry Soybean,Squash Strawberry,Tomato	38	Training from scratch AlexNet: 97.82% GoogleNet: Training from scratch: 98.36%
5	SHNet373	PlantVillage	GPU: GeForce GTX 1080 2560 CUDA cores, 8GB RAM,	54306	14	Similar to row 4	38	96.20%
6	SHNet131	PlantVillage	GPU: GeForce GTX 1080 2560 CUDA cores, 8GB RAM,	54306	14	Similar to row 4	38	97.60%
7	AgriNet373	PlantVillage	GPU: GeForce GTX 1080 2560 CUDA cores, 8GB RAM,	54306	14	Similar to row 4	38	97.90%
8	AgriNet131	PlantVillage	GPU: GeForce GTX 1080 2560 CUDA cores, 8GB RAM,	54306	14	Similar to row 4	38	98%

4. Conclusion

In this work, a new method was proposed for diagnosing and classifying plant diseases of various crops such as corn and apples using CNN. In this way, the learnable parameters of the proposed network were reduced. Based on the results of the implementation of the proposed architectures, the high efficiency and accuracy of the AgriNet131 architecture in diagnosing and classifying diseases of agricultural products were proved. Given the variety of diseases in the database, the accuracy of 98% seems acceptable. In the future, in order to improve the network performance, after using SENet, the channels can be grouped, and then those groups can be shuffled through channel shuffling operations, whereby better results can be obtained.

Acknowledgment
This research work did not receive any specific grant from funding agencies in the public, commercial or not-for-profit sectors.

References

[1] K. Kiani, R. Hematpour, and R. Rastgoo, "Automatic Grayscale Image Colorization using a Deep Hybrid Model," *Journal of AI and Data Mining,* vol. 9, no. 3, pp. 321-328, 2021.

[2] N. Sharma, V. Jain, and A. Mishra, "An Analysis Of Convolutional Neural Networks For Image Classification," *Procedia Computer Science,* vol. 132, pp. 377-384, 2018.

[3] J. Jung, M. Maeda, A. Chang, M. Bhandari, A. Ashapure, and J. Landivar-Bowles, "The potential of remote sensing and artificial intelligence as tools to improve the resilience of agriculture production systems," *Current Opinion in Biotechnology,* vol. 70, pp. 15-22, 2021.

[4] B. Liu, Y. Zhang, D. He, and Y. Li, "Identification of Apple Leaf Diseases Based on Deep Convolutional Neural Networks," *Symmetry,* vol. 10, no. 1, pp. 11, 2018.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM,* vol. 60, no. 6, pp. 84-90, 2017.

[6] C. Szegedy, L. Wei, J. Yangqing, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions." 2015, pp. 1-9.

[7] Q. Yan, B. Yang, W. Wang, B. Wang, P. Chen, and J. Zhang, "Apple Leaf Diseases Recognition Based on An Improved Convolutional Neural Network," *Sensors,* vol. 20, no. 12, 2020.

[8] Y. Zhong, and M. Zhao, "Research on deep learning in apple leaf disease recognition," *Computers and Electronics in Agriculture,* vol. 168, pp. 105146, 2020.

[9] W.-z. Liang, K. R. Kirk, and J. K. Greene, "Estimation of soybean leaf area, edge, and defoliation using color image analysis," *Computers and Electronics in Agriculture,* vol. 150, pp. 41-51, 2018.

[10] A. Karlekar, and A. Seal, "SoyNet: Soybean leaf diseases classification," *Computers and Electronics in Agriculture,* vol. 172, pp. 105342, 2020.

[11] S. Kaur, S. Pandey, and S. Goel, "Semi-automatic leaf disease detection and classification system for soybean culture," *IET Image Processing,* vol. 12, no. 6, pp. 1038-1048, 2018.

[12] J. Xiong, D. Yu, Q. Wang, L. Shu, J. Cen, Q. Liang, H. Chen, and B. Sun, "Application of Histogram Equalization for Image Enhancement in Corrosion Areas," *Shock and Vibration,* vol. 2021, pp. 8883571, 2021/01/23, 2021.

[13] Anjna, M. Sood, and P. K. Singh, "Hybrid System for Detection and Classification of Plant Disease Using Qualitative Texture Features Analysis," *Procedia Computer Science,* vol. 167, pp. 1056-1065, 2020.

[14] M. Turkoglu, and D. Hanbay, "Leaf-based plant species recognition based on improved local binary pattern and extreme learning machine," *Physica A: Statistical Mechanics and its Applications*, 2019.

[15] R. I. Hasan, S. M. Yusuf, and L. Alzubaidi, "Review of the State of the Art of Deep Learning for Plant Diseases: A Broad Analysis and Discussion," *Plants,* vol. 9, no. 10, pp. 1302, 2020.

[16] Y. S. Aurelio, G. M. de Almeida, C. L. de Castro, and A. P. Braga, "Learning from Imbalanced Data Sets with Weighted Cross-Entropy Function," *Neural Processing Letters*, vol. 50, no. 2, pp. 1937-1949, 2019.

[17] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices." 2018, pp. 6848-6856.

[18] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," 2018, pp. 122-138.

[19] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks." 2018, pp. 7132-7141.

[20] K. Rangarajan Aravind, P. Maheswari, P. Raja, and C. Szczepański, "Chapter nine - Crop disease classification using deep learning approach: an overview and a case study," *Deep Learning for Data Analytics*, pp. 173-195: Academic Press, 2020.

[21] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognition,* vol. 77, pp. 354-377, 2018.

[22] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions." 2017, pp. 1800-1807.

[23] N. Yao, F. Ni, Z. Wang, J. Luo, W.-K. Sung, C. Luo, and G. Li, "L2MXception: an improved Xception network for classification of peach diseases," *Plant Methods,* vol. 17, no. 1, pp. 36, 2021.

[24] L. Chen, H. Fei, Y. Xiao, J. He, and H. Li, "Why batch normalization works? a buckling perspective." 2017, pp. 1184-1189.

[25] N. D. Marom, L. Rokach, and A. Shmilovici, "Using the confusion matrix for improving ensemble classifiers." 2010, pp. 555-559.

[26] E. M. Dogo, O. J. Afolabi, N. I. Nwulu, B. Twala, and C. O. Aigbavboa, "A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks." 2018, pp. 92-99.

[27] M. Brahimi, M. Arsenovic, S. Laraba, S. Sladojevic, K. Boukhalfa, and A. Moussaoui, "Deep Learning for Plant Diseases: Detection and Saliency Map Visualisation," *Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent*, pp. 93-117, Cham: Springer International Publishing, 2018.

[28] M. Agarwal, S. K. Gupta, and K. K. Biswas, "Development of Efficient CNN model for Tomato crop disease identification," *Sustainable Computing: Informatics and Systems,* vol. 28, pp. 100407, 2020.

[29] H. Durmus, E. O. Günes, and M. Kirci, "Disease detection on the leaves of the tomato plants by using deep learning," in 2017 6th International Conference on Agro-Geoinformatics, 2017, pp. 1-5.

[30] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using Deep Learning for Image-Based Plant Disease Detection," *Frontiers in Plant Science*, vol. 7, no. 1419, pp. 1-7, 2016.

[31] L.-C. Chang, E. El-Araby, V. Q. Dang, and L. H. Dao, "GPU acceleration of nonlinear diffusion tensor estimation using CUDA and MPI," *Neurocomputing*, vol. 135, pp. 328-338, 2014.

AgriNet: یک شبکه عصبی پیچشی طبقه‌بند جدید جهت تشخیص بیماری‌های محصولات کشاورزی

فرزانه سلیمیان نجف‌آبادی [۱] و محمدتقی صادقی [۲،*]

[۱] پردیس آزادی، دانشگاه یزد، یزد، ایران.

[۲] دانشکده مهندسی برق، دانشگاه یزد، یزد، ایران .

ارسال ۲۰۲۱/۱۱/۰۴؛ بازنگری ۲۰۲۲/۰۱/۰۸؛ پذیرش ۲۰۲۲/۰۴/۱۳

چکیده:

یکی از بخش‌های مهم و موثر بر اقتصاد کشورها، بخش کشاورزی است. توسعه و بهبود فعالیت‌ها در این بخش از موضوعات مورد توجه محققان است. یکی از مشکلاتی که کشاورزان در فعالیت‌های کشاورزی با آن مواجه هستند، بیماری‌های گیاهی است. اگر مشکل گیاه به زودی تشخیص داده شود، کشاورز می‌تواند بیماری را به طور موثرتری درمان کند. در این تحقیق، یک شبکه عصبی مصنوعی عمیق جدید به نام AgriNet معرفی می‌شود که از تصاویر برگ‌های گیاهان برای تشخیص برخی از انواع بیماری‌ها در محصولات کشاورزی استفاده می‌کند. شبکه پیشنهادی از تکنیک‌های به هم زدن کانال (ShuffleNet) و مدل‌سازی وابستگی کانال(SENet) استفاده می‌کند. یکی از عوامل موثر بر اثربخشی معماری شبکه پیشنهادی، نحوه افزایش جریان اطلاعات در کانال‌ها پس از مدل‌سازی صریح وابستگی‌های متقابل بین کانال‌ها است. این در واقع یک نوآوری مهم این پژوهش است. مجموعه داده مورد استفاده در این کار PlantVillage است که شامل ۱۴ نوع گیاه در ۲۴ گروه سالم و بیمار است. نتایج تجربی ما نشان می‌دهد که روش پیشنهادی از روش‌های دیگر در این زمینه بهتر عمل می‌کند. AgriNet به ترتیب به دقت و هزینه ۹۸٪ و ۷٪ در داده‌های تجربی دست یافته است. این روش در مقایسه با روش ShuffleNetV2 دقت تشخیص را تا حدود ۲٪ افزایش داده و مقدار هزینه نهایی را به مقدار ۸٪ کاهش می‌دهد.

کلمات کلیدی: تشخیص بیماری‌های گیاهی، شبکه‌های عصبی پیچشی، شبکه ShuffleNet، شبکه SENet.