Artificial Intelligence-Based Plant Diseases Classification

Lobna M. Abou El-Maged, Ashraf Darwish, and Aboul Ella Hassanien

Abstract Machine learning techniques are used for classifying plant diseases. Recently, deep learning (DL) is applied in the classification process of image processing. In this chapter, convolutional neural network (CNN) is used to classify plant diseases images. However, CNN suffers from the hyper parameters problem which can affect the proposed model. Therefore, Gaussian optimization method is used to overcome the hyper parameters problem in CNN. This chapter proposed an artificial intelligence model for plants diseases classification based on convolutional neural network (CNN). The proposed model consists of three phases; (a) preprocessing phase, which augmented the data and balanced the dataset; (b) classification and evaluation phase based on pre-train CNN VGG16 and evaluate the results; (c) optimize the hyperparameters of CNN using Gaussian method. The proposed model is tested on the plant's images dataset. The dataset consists of nine plants with thirty-three cases for diseased and healthy plant's leaves. The experimental results before the optimization of pre-trained CNN VGG16 achieve 95.87% classification accuracy. The experimental results improved to 98. 67% classification accuracy after applied the Gaussian process for optimizing hyperparameters.

Keywords Plants diseases · Deep learning · Convolution neural network · Hyper parameter optimization · Gaussian process

L. M. Abou El-Maged (✉)
Computer Science Department, MET High Institute, Mansoura, Egypt
e-mail: lobna_acd@hotmail.com
URL: http://www.egyptscience.net/

A. Darwish
Faculty of Science, Helwan University, Helwan, Egypt

A. E. Hassanien
Faculty of Computers and Artifical Intellgence, Cairo University, Giza, Egypt

A. E. Hassanien et al. (eds.), *Machine Learning and Big Data Analytics Paradigms: Analysis, Applications and Challenges*, Studies in Big Data 77,
https://doi.org/10.1007/978-3-030-59338-4_3

1 Introduction

The agricultural sector is one of the most important sectors of the economy for all countries, therefore it is one of the elements of sustainable development. Plant diseases significantly affect crop productivity and agricultural wealth. The economic importance of plant diseases in agricultural production in terms of their impact on the national economy is one of the main factors that affect the quantity of the crop.

Parasitic diseases are the most common diseases among plants, and there are many pathogens of the plant including fungi, viruses and bacteria. The simple farmer can't easily identify plant diseases, but he goes to an expert in plant diseases, which in turn may be specialized only in diseases of fruits or vegetables or even in one type of them. Therefore, in this chapter an intelligent system was built to classify different types of plant diseases of vegetables and fruits. This system may be a substitute for the human expert, which achieves the speed and accuracy in the identification and treatment of plant diseases quickly before the spread of the disease. In this chapter the researchers try to get optimal performance for the identification of the plants' diseases.

DL is an artificial neural network architecture which contains many processing layers. DL have been used in many fields like; image recognition, voice recognition, and also other huge applications that deal with the analysis of large volumes of data, like, e.g., self-driving cars and natural language processing systems [1–3]. The DL is differ than the traditional machine learning in how features are extracted. Traditional machine learning approaches use feature extraction algorithms as a preceding stage for the classifier, but in DL the features are learned and represented hierarchically in multiple levels. So DL is better for machine learning than the traditional machine learning approaches [4]. There are many types of DL tools, the most commonly used are the Convolutional Neural Networks (CNN) [3].

Various researches have been done in the field of agriculture, specifically in the identification of plants and the identification of plant diseases using CNN. In [1] two well-known architectures of CNNs are compared in the identification of 26 plant diseases, using an open database of leaves images of 14 different plants. For the corn specifically the result was for Corn (maize) Zea is 26.1%, Cercospora leaf spot is 35.2%, Common rust is 73.9%, and Northern Leaf Blight is 100%.

Two well-known and established architectures of CNNs AlexNet and GoogLeNet are compared in [19] in the identification of 26 plant diseases, using an open database of leaves images of 14 different plants. In [16] developed a similar methodology for plant disease detection through leaves images using a similar amount of data available on the Internet, which included 13 diseases and 5 different plants accuracy of their models, were between 91% and 98%, depending on the testing data. More recently, in [20] compared the performance of some conventional pattern recognition techniques with that of CNN models, in plants identification, using three different databases of images of either entire plants and fruits, or plant leaves, concluding that CNNs drastically outperform conventional methods. In [21] the researcher developed CNN models for the detection of 9 different tomato diseases and pests, with satisfactory

performance (36,573 no of plants). Finally in [1] perform plant disease detection and diagnosis through deep learning methodologies. Training of the models was performed with the use of an open database of 87,848 images, containing 25 different plants in a set of 58 distinct classes, he use several model architectures.

The CNN has many architectures that concerning the identification of plant diseases; AlexNet [5], Overfeat [6], AlexNetOWTBn [7], GoogLeNet [8], and VGG [9]. CNN has a number parameters called hyper parameters, these parameters determine the network structure, these parameters are variables such as; the number of hidden layers and the learning rate, the hyper parameters values are set before training the network [10]. In this chapter, Gaussian Optimization method has been used with CNN to find the optimal hyper parameters in a fine- tuning CNN.

The rest of this chapter is present as follows. Section 2 describes the basics and background. Section 3 presents the Materials and methods. Section 4 the results and discussion. Finally; Sect. 5 presents the conclusion.

2 Preliminaries

This section presents the basic information about CNN, VGG16 & fine-tuning and Gaussian optimization.

2.1 Convolutions Neural Network

CNN is an evolution of traditional artificial neural networks and it is network architecture for deep learning [1, 4, 11]. They are used to object recognition and image classifications. A CNN is like a multilayer neural network, it consists of convolutional layers (may be one or more), and then followed by fully connected layers (may be one or more). Each layer generates a successively abstraction of the input data with high level, called a feature map, which save essential unique information. Amazing performance could to be achieve in modern CNNs by employing a very deep hierarchy of layers [11]. The numbers of layers used in DL range from five to more than a thousand. Generally there are many CNN architectures like; AlexNet, AlexNetOWTBn, GoogLeNet, Overfeat, and VGG [1, 5–9].

In the image analysis tasks, the CNN can be trained to do image classification, object detection, image segmentation and finally image processing [12, 13].

The CNN architecture as shown in Fig. 1 are consist of two main layers; layers for feature extraction and layers for classification [4]. The feature extraction layers are convolution layer and max-pooling layer:

- ***Convolution Layer***
  The learnable kernels are used for convolving feature maps from previous layers. The output of the kernels go through a linear or non-linear activation function

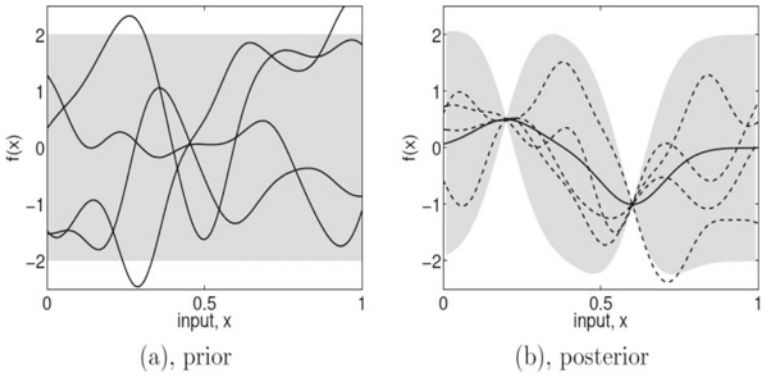

Fig. 1 **a.** The prior distribution. **b.** Shows the situation after two data points have been observed

specified as a sigmoid, inflated tangent, Softmax, rectified additive, and sameness functions to make the feature maps. Each of the output feature maps can be conjunctive with much than one input feature map.

$$\text{Suppose}: \qquad LR_i^l = f\left(\sum\nolimits_{j \in Mi} LR^{l-1} * k_{ji}^l + B_i^l\right) \tag{1}$$

Where LR_i^l the product of the present layer is, LR_i^{l-1} is product of the previous layer, k_{ji}^l is the kernel for the present layer, and B_i^l are biases for the present layer. F_i represents a selection of input maps. An additive bias B is given for each output map.

- **max-pooling Layer**
  This bed perfect distribution the dimension of the feature maps depending on the filler of the land distribution mask [4].

The classification layer uses the extracted features from a previous convolutional layer to classify each class. The layers are fully connected in the classification layer. The final layer are represented as vectors with scalar values which are passed to the fully connected layers. The fully connected feed-forward neuronal layers are misused as a soft-max classification layer [4]. Still, in most cases, two to quaternion layers feature been observed in assorted architectures including LeNet, AlexNet, and VGG Net [4].

Soft-max function calculates the probabilities distribution of different events. The soft-max function calculates the probabilities for each class over all possible classes. The calculated probabilities determining the target class for the present inputs [13].

CNN has a number of hyper parameters. The hyper parameters determine the structure of the network such as the number of hidden layers and the learning rate, the hyper parameters are set before the network training [10]. We need to optimize the hyper parameter to reduce the effort of the human necessary for applying machine

learning and to enhance the performance of machine learning algorithms. The hyper parameter can be real value such as "learning rate", integer value such "number of layers", binary, or categorical such as "choice of optimize" [14].

For enhancing the deep networks training and to avoid the over-fitting, a lot of techniques were proposed. Typical entirety allows pot normalization (BN), and dropout [12]. To enhance the expressive commonwealth of CNNs, non-linear activation functions were well-studied, like as ReLU, ELU and SELU. Also, a lot of realm peculiar techniques were also mature to add fine-tunes networks on special application [12].

2.2 VGG16 Architecture

VGG16 is the CNN model proposed by Simonyan and Zisserman [9]. The model operated on a dataset contains over 14 million images categories of 1,000 classes. It achieves 92.7% as test accuracy in ImageNet dataset. VGG16 achieves improvement over AlexNet by replacing large kernel-sized filters with 3 × 3 kernel-sized filters one after another. NVIDIA Titan Black GPU's was used to train VGG16 for weeks.

In VGG16, the image with fixed size 224 × 224 RGB is an input to the convolution layer. There are stack of convolution layers where the image pass through them using filters with a very small receptive field: 3 × 3. In one of the configurations, it also utilizes 1 × 1 convolution filters, which can be seen as a linear transformation of the input channels then followed by non-linearity transformation. The stride is set to 1 pixel; the spatial padding of convolution. The input of the layer is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3 × 3 convolution layers. After that the spatial pooling is achieved by 5 max-pooling layers, which follow some of the convolution layers not all ones. Max-pooling is performed over a 2 × 2 pixel window, with stride 2. After the stack of convolutional layers, three Fully-Connected (FC) layers are come. The first two FC layers have 4096 channels each, the third one performs 1000-way ILSVRC classification, so it contains 1000 channels one for each class. The final layer is the soft-max layer. The fully connected layers' configuration is the same all networks. All hidden layers are prepared by the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response Normalization (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to more complexity for time and memory [9, 15].

The CNN architecture can be reused for new datasets which is differ than its own dataset, the new datasets may vary in size or nature of images. So; we can use fine-tuning deal with this situation. Fine-tuning tries to improve the effectiveness or efficiency of a process or function by making small modifications in the used CNN architecture to improve or optimize the outcome [16].

2.3 Hyper Parameters

The hyper parameters are the variables which determine the network structure such as the number of hidden layers and the variables which determine how the network is trained such the learning rate), the Hyper parameters are set before training the network [10]. Automated hyperparameters optimization has several important use cases; it can reduce the human effort necessary for applying machine learning. It used to improve the performance of machine learning algorithms.

Let A denote a machine learning algorithm with N hyperparameters. We denote the domain of the n-th hyperparameters by Sn and the overall hyperparameters configuration space as $S = s_1 \times s_2 \times \ldots s_N$. A vector of hyperparameters is denoted by $\lambda \in S$, and A with its hyperparameters instantiated to λ is denoted by $A\lambda$. The domain of a hyperparameters can be real-valued (e.g., learning rate), integer-valued (e.g., number of layers), binary (e.g., whether to use early stopping or not), or categorical (e.g., choice of optimizer) [14].

Given a data set DS, our goal is to find

$$\lambda^* = \arg min_{\lambda \in S} E_{(DS_{train}, DS_{valid}) \sim DS)} V(L, A_\lambda, DS_{train}, DS_{valid}) \quad (2)$$

where $V(L, A_\lambda, DS_{train}, DS_{valid})$ measures the loss of a model generated by algorithm A with hyperparameters λ on training data DS_{train} and evaluated on validation data DS_{valid}. In practice, it is only access to finite data DS ~ DS and thus need to approximate the expectation in Eq. 2 Popular choices for the validation protocol V(·, ·, ·, ·) are the holdout and cross-validation error for a user-given loss function (such as misclassification rate).

2.4 Gaussian Optimization Method

The optimization method is an iterative algorithm has two main parts, the probabilistic surrogate model and the acquisition function to determine which next point to be evaluated. The alternate modelling is fitted to all observations of the reference use made so far in each iteration. Then the acquisition function uses the prophetic dispersion of the probabilistic copy to determine the utility of various candidate points, trading off exploration and exploitation. Compared to evaluating the dear operate, the acquisition role is to compute and can thence be good optimized [14, 17]. The expected improvement (EI):

$$E[I(\lambda)] = E[\max(f^*_{\min} - Y, 0)] \quad (3)$$

EI can be computed in closed form if the model prediction Y at configuration λ according to a normal distribution:

$$E[I(\lambda)] = \left(f^*_{min} - \mu(\lambda)\right) \Phi\left(\frac{f^*_{min} - \mu(\lambda)}{\sigma}\right) + \sigma \emptyset\left(\frac{f^*_{min} - \mu(\lambda)}{\sigma}\right) \quad (4)$$

where ϕ(.) is the standard normal density, Φ(.) is the distribution function, and $f^*_{\min}$ is the optimal observed value [14, 17].

Assume that we are then supposition a dataset, the proper random functions showed in Fig. 1a, in Fig. 1b. The broken lines pretense have functions which are reconciled with D, and the solidified line depicts the tight quantity of much functions. Respond how the uncertainness is low nestled to the observations. The combining of the prior and the data leads to the tooth posterior distribution over functions [17].

3 Materials and Methods

3.1 Plant's Image Dataset

The dataset of the plant's diseases images is a "kaggle-dataset" [18], it consists 159.984 images of diseases and healthy leaves' plants present in Fig. 2; 128,028 of plants leaves images for training and 31,956 images for testing. The images are present diseased and healthy plant's leaves. The dataset is characterized by distinguishing images which may be taken at different angles and different backgrounds. Table 1 shows that the data set contains thirty-three categories of nine plants leaves; apple, cherry, corn, grape, peach, Pepper, potato, Strawberry and tomato.

3.2 The Proposed Plant's Diseases Classification Model

The AI proposed model is consisting of three main phases (a) preprocessing phase, (b) classification and evaluation phase and (c) hyperparameter optimization using Gaussian process phase. Each phase is composed of a number of steps as presented in Fig. 3. In the AI proposed model; the image of the plants are gained from the data set, after executing the preprocessing phase, the Boolean variable "optimize" is set true, then process classification and evaluation phase executed, the output of this phase is the evaluation of the results. If the results are satisfying and the value of the "optimize" variable is "false", the algorithm will end otherwise, the optimization phase will execute a number of n times. The detail of each process is given below.

(i) **Preprocessing phase.**

Preprocessing phase is used for processing and formatting images for use in the next stages, it consisting of two processes; data augmentation and balanced the dataset as in Fig. 3. The detail of each process is given below.

Data Augmentation The data was augmented to avoid overfitting and present more varieties in the data set. The augmentation for data was done in five different

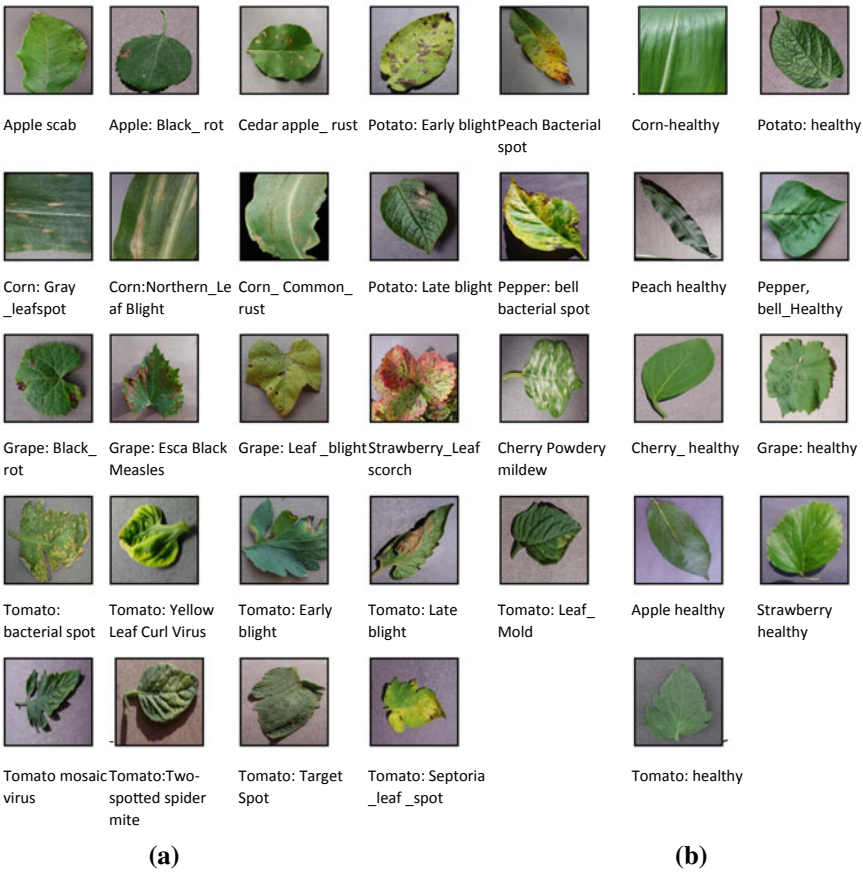

Fig. 2 **a.** Sample of diseased plants' leaves **b.** sample of healthy plants' leaves

methods. The augmentation methods are; rotate right 30°, rotate left 30 & +90°, flip horizontally about Y-axis and shear.

Balance Dataset: The dataset showed in the Table 1 is imbalance dataset, the imbalanced lead to inaccurate results, so we calculate the class weight for the categories in the dataset to use it later for our building model. The class weight will be, for example, the first weight is '1.92135642' which presents 2017 images for the apple scab is less than third weight '4.40165289' presents 880 images for the cedar-apple rust.

(ii) **classification and evaluation phase**

The classification and evaluation phase is consists of 4 processes; hyperparameter setting, proposed CNN architecture, CNN Training, and evaluation the results. The detail of each step is given below.

Table 1 Data set description

#	Plant	Disease	Train samples	Test samples
1	Apple	Scab	2017	505
2		Black rot	1988	496
3		Cedar apple rust	880	202
4		Healthy	5264	1316
5	Cherry	Healthy	2736	680
6		Powdery mildew	3368	840
7	Corn	Cercospora leaf spot Gray leaf spot	1644	408
8		Common rust	3816	952
9		Healthy	3720	928
10		Northern Leaf Blight	3125	788
11	Grape	Black rot	3776	944
12		Esca_(Black Measles)	4428	1104
13		Healthy	1356	336
14		Leaf blight_(Isariopsis_Leaf_Spot)	3444	860
15	Peach	Bacterial_spot	7352	1837
16		Healthy	1152	228
17	Pepper	Bacterial_spot	3193	796
18		Healthy	4725	1180
19	Potato	Early_blight	3200	800
20		Healthy	488	120
21		Late_blight	3200	800
22	Strawberry	Healthy	1460	364
23		Leaf scorch	3552	884
24	Tomato	Bacterial spot	6808	1700
25		Early blight	3200	800
26		healthy	5089	1272
27		Late blight	6109	1524
28		Leaf Mold	3048	760
29		Septoria leaf spot	5668	1416
30		Two-spotted spider mite	5364	1340
31		Target Spot	4496	1120
32		Mosaic virus	1196	296
33		Yellow Leaf Curl Virus	17,144	2484
Total			128,028	38,836

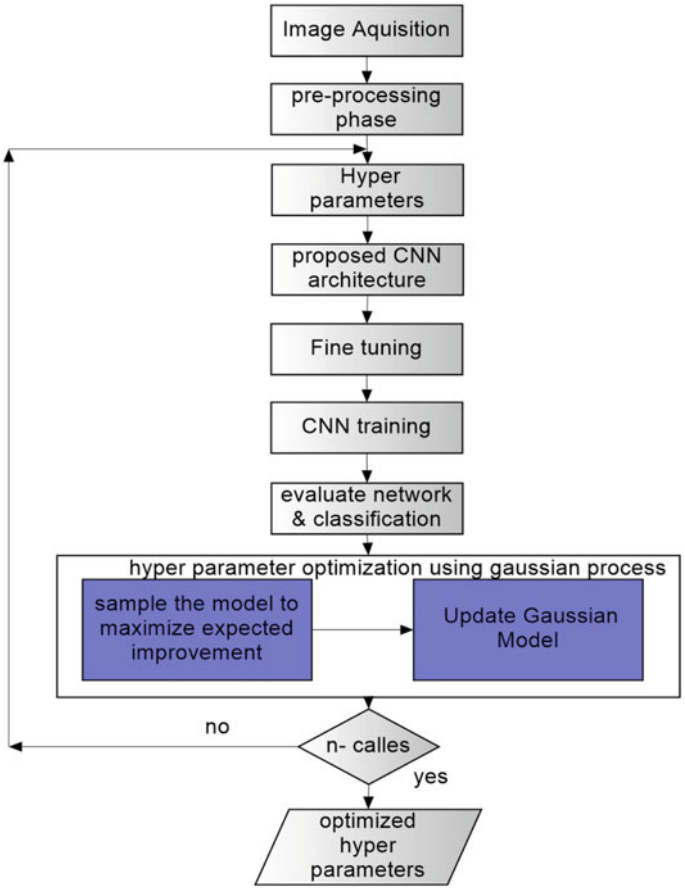

Fig. 3 The proposed model for hyper parameter optimization in fine-tuning CNN using Gaussian process

Hyperparameter Setting: This process is responsible for setting values of the hyper-parameters, this work uses three types of hyperparameters; learning rate, drop_out value, and activation function type.

Proposed CNN Architecture: The proposed architecture depends on the usage of a pre-train CNN VGG16 architecture which is a very convincing way since it has been used previously and has given good results. The proposed architecture is consists of two main steps; the first one is to create a CNN VGG16 then the second one is to make some adaptation in the architecture to fit the dataset. Figure 4 presents the proposed CNN architecture.

As seen in the early section, the VGG16 is a CNN architecture used to classify 1000 classes, it consists of 16 layers. In the proposed architecture, the convolutions layers in the VGG16 is used, then modifying the rest architecture by adding two

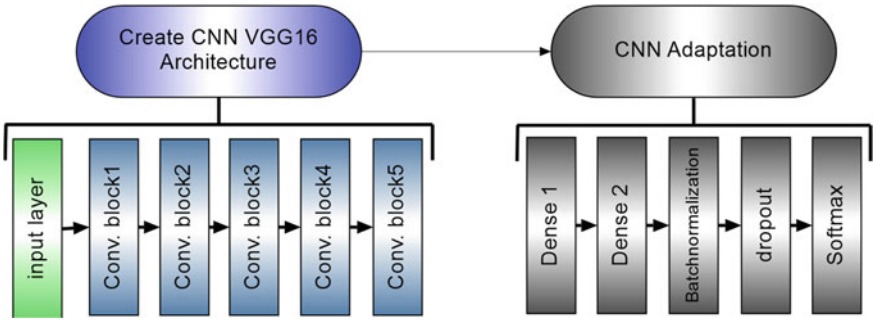

Fig. 4 CNN proposed architecture

dense layers, Batch Normalization layer, Dropout layer to prevent overfitting, and finally the classification layer.

The CNN Training Process: In the training process the fine-tuning is used Fig. 5. The Fine-Tuning is optimized both the weights of the new classification layers that have been added and also all of the layers from the VGG16 model [16]. The training process is based on the hyper-parameter values which sat in the previous process

Evaluation the Results: This process is responsible for evolution the accuracy of the proposed AI model, the evaluation results used to decide a road chart of the model as seen in Fig. 3.

(iii) Hyper Parameter Optimization for CNN using Gaussian process phase.

The optimization method is used for two reasons; the first one is to get the optimal hyper_ parameters' values and the other is to provide the human effort in trying to get the optimal hype_ parameters' values by doing the system with this task.

The proposed model setting parameters used the learning rate = 1e−6 with Adam optimizer, dropout rate = 0.2 and activation function is 'relu' as initial values. In the Gaussian method; the model is sampled with expected improvement, then update

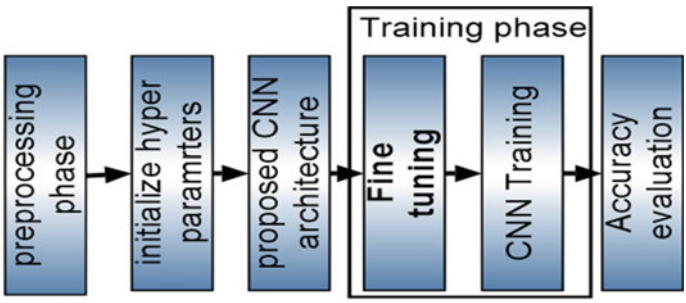

Fig. 5 Proposed model with fine—tuning

the Gaussian model according to the classification accuracy, after that check if the number of iteration calls is maximum or not to decide whether re-execute the model or not. After finishing the n calls iteration, it will give optimized hyperparameters.

4 Experiments Results and Discussion

The experiments are done using tensor flow and Keras with GPU google colab. The experiments are done in three experiments. In the first experiment is implementing the proposed model without optimization. In the second experiment, perform the optimization process using the Gaussian method. In the third experiment, implement the proposed model after optimization.

4.1 Experiment (I): Without Optimization

In this experiment, the structure of VGG16 is modified by adding two dense layers, adding a BatchNormalization layer, adding a Dropout layer, and finally a classification layer. In this experiment, we improved both the weights of the new classification layers and all layers of the VGG16 model.

The hyper_parameters used are learning rate = 6−1e, drop_out = 0.2 and the activation function is "relu". Figure 6 shows the Test-set classification accuracy: 95.87%.

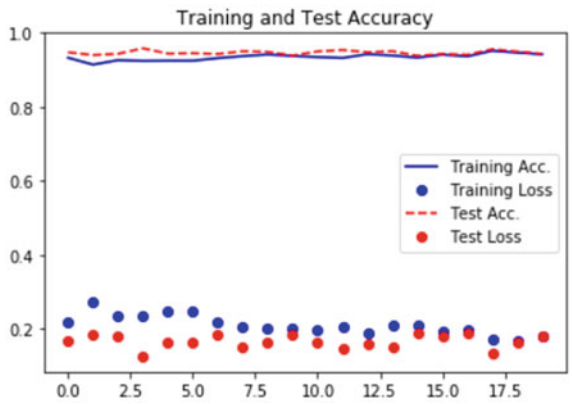

Fig. 6 Training history of CNN network before hyperparameter optimization

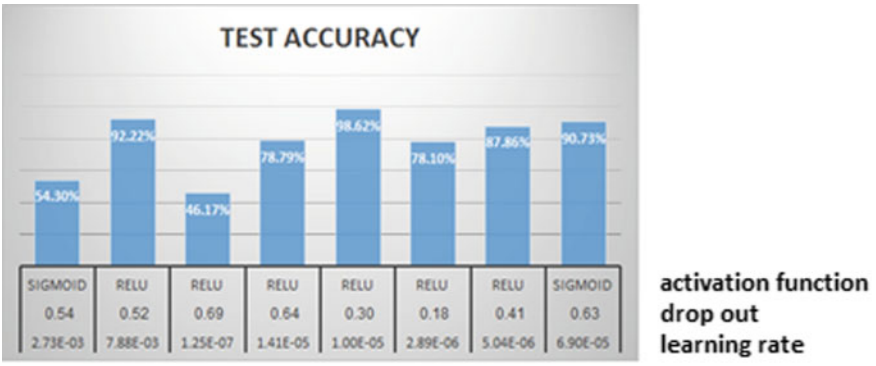

Fig. 7 Test accuracies during optimization process

4.2 Experiment (II): Hyper_ Parameter Optimization Using Gaussian Process

In this experiment, Gaussian process optimization is used with EI and with minimum 11 iteration calls. The hyper-parameters are 'learning_rate' within range (from low = 1e−6 to high = 1e−2), 'drop_out' within range (from low = 0.1 to high = 0.9) and finally 'activation' with categories ('relu' and 'sigmoid'). The fitness function is the function that creates and trains a neural network with the given hyper-parameters, and then evaluates its performance on the data set. The function then returns the so-called fitness, which is the negative classification accuracy on the validation set. It is negative because the performance minimization instead of in Fig. 7.

For example test accuracy is about 91% when learning rate = 6.8.99, Drop_out = 0.63 and activation function is "sigmoid". It is illustrated that the best values are; learning rate = 1e−5 with droup out = 0.3 and activation function is 'relu'.

4.3 Experiment (III): Hyperparameters Optimization

In this experiment, the CNN architecture is run with new parameters such as learning rate = 1e−5, drop out = 0.3 and activation function = 'relu'. It gives Test-set classification accuracy: 98.67%. Figure 8 shows the historical training.

The results as shown in Table 2 shows that the total test accuracy is improved with 2.8%. At the plants level, test accuracy for most plants are improved but test accuracy for a small number of plants doesn't change such as cherry and corn. It is noted that the plant, whose accuracy has not changed, has fairly high accuracy.

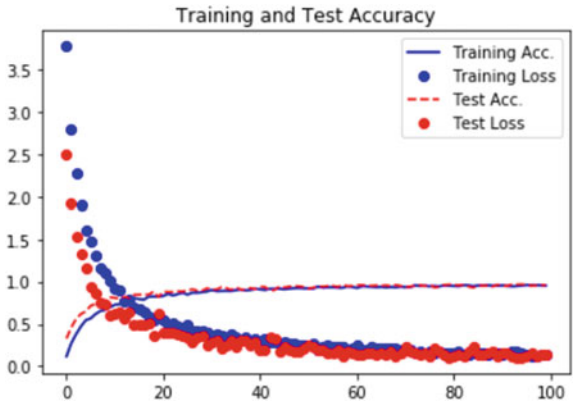

Fig. 8 Training history of CNN network of proposed AI model

5 Conclusion and Future Work

Agriculture plays an important role in the economic development for many countries so classification the plant's diseases are very important. Deep learning mimics human thinking, it has been used extensively in the last decade. VGG16 is an architecture of CNN. Using fine-tuning CNN VGG16 model by adding several dense layers, Batch Normalization layer, Dropout layer and finally the classification layer, this lead to accuracy 95.87%. Automated hyperparameters optimization is important to help the human by reducing his effort necessary for applying machine learning. It used to enhance the performance of the algorithm for machine learning. The Gaussian method can be used for hyperparameters optimization. It uses the predictive distribution of the probabilistic model. The Gaussian method identifies the importance of different candidate points, trading off exploration and exploitation, this function is cheap. Using Gaussian process optimization with EI illustrated that the best values are; learning rate = 1e−5 with drop_ out = 0.3 and activation function is 'real'. Using fine-tuning with new parameters: learning rate = 1e−5, drop_out = 0.3 and activation function = 'relu', It gives accuracy 98.76%. I.e. Gaussian process optimization helps in improving accuracy from 95.87% to 98.67%. In future work, we will use the swarm algorithm for hyperparameters optimization as a trial to enhance accuracy.

Table 2 Accuracy test results

#	Plant	Disease	Test accuracy before hyper parameters optimization (%)	Test accuracy after hyper parameters optimization (%)
1	Apple	Scab	91.47	94.44
2		Black rot	96.58	100.00
3		Cedar apple rust	97.33	99.55
4		Healthy	99.69	100.00
5	Cherry	Healthy	98.21	98.21
6		Powdery mildew	98.68	98.68
7	Corn	Cercospora leaf spot Gray leaf spot	95.15	98.04
8		Common rust	99.79	100.00
9		Healthy	92.89	96.70
10		Northern Leaf Blight	99.89	99.89
11	Grape	Black rot	97.88	97.88
12		Esca_(Black Measles)	96.24	99.73
13		Healthy	97.93	100.00
14		Leaf blight_(Isariopsis_Leaf_Spot)	99.41	100.00
15	Peach	Bacterial _spot	99.89	99.89
16		Healthy	92.91	95.49
17	Pepper	Bacterial _spot	97.86	98.99
18		Healthy	98.74	99.75
19	Potato	Early _blight	98.27	99.38
20		healthy	93.20	98.24
21		Late _blight	90.08	98.33
22	Strawberry	Healthy	99.55	99.66
23		Leaf scorch	96.15	100.00
24	Tomato	Bacterial spot	95.49	99.00
25		Early blight	86.90	99.59
26		Healthy	96.72	98.62
27		Late blight	91.31	98.42
28		Leaf Mold	97.30	98.80
29		Septoria leaf spot	97.52	99.55
30		Two- spotted spider mite	90.63	98.21
31		Target Spot	97.97	99.11
32		Mosaic virus	84.80	92.23
33		Yellow Leaf Curl Virus	97.22	99.84

(continued)

Table 2 (continued)

#	Plant	Disease	Test accuracy before hyper parameters optimization (%)	Test accuracy after hyper parameters optimization (%)
Total			95.87	98.67

References

1. Ferentinos, K.P.: Deep learning models for plant disease detection and diagnosis. Comput. Electron. Agric. **145**, 311–318 (2018)
2. Zhang, X., et al.: Identification of Maize leaf diseases using improved deep convolutional neural networks. IEEE Xplore Digital Library, Digital Object Identifier (2018). https://doi.org/10.1109/ACCESS.2018.2844405
3. Barbedo, J.G.A.: Factors influencing the use of deep learning for plant disease recognition. Biosyst. Eng. **72**, 84–91 (2018)
4. The History Began from AlexNet: A comprehensive survey on deep learning approaches. https://arxiv.org/abs/1803.01164
5. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Proceedings of 25th International Conference on Neural Information Processing Systems, Lake Tahoe, Nevada, 03–06 December 2012, vol. 1, pp. 1097–1105 (2012)
6. Sermanet, P., et al.: Overfeat: integrated recognition, localization and detection using convolutional networks. arXiv:1312.6229 (2013)
7. Krizhevsky, A.: One weird trick for parallelizing convolutional neural networks. arXiv:1404.5997 (2014)
8. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015)
9. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (2014)
10. https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a. Accessed May 2019
11. Yann, L., et al.: Gradient based learning applied to document recognition. In: Proceedings of the IEEE, November 1998
12. Wang, T., Huan, J., Li, B.: Data dropout: optimizing training data for convolutional neural networks. arXiv:1809.00193v2 (2018)
13. Islam, M.S., et al.: InceptB: a CNN based classification approach for recognizing traditional Bengali games. In: 8th International Conference on Advances in Computing and Communication (ICACC 2018). Elsevier (2018)
14. Feurer, A., Hutter, F.: Hyperparameter optimization, Chap 1. In: Automated Machine Learning: Methods, Systems, Challenges. Springer (2019)
15. https://neurohive.io/en/popular-networks/vgg16/. Accessed March 2019
16. Hussain, M., Bird, J.J., Faria, D.R.: Advances in computational intelligence systems. In: UKCI 2018. AISC, vol. 840, pp. 191–202 (2019)
17. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press, Massachusetts Institute of Technology (2006). www.GaussianProcess.org/gpml, ISBN 026218253X
18. https://storage.googleapis.com/kaggle-datasets/53092/100927/plantdisease.zip
19. Hughes, D., Salathé1, M.: Using deep learning for image-based plant disease detection Sharada Prasanna Mohanty. Front. Plant Sci. **7**, 1419 (2016)

20. da Silva Abade, A., et al.: Plant diseases recognition from digital images using multichannel convolutional neural networks. http://www.insticc.org/Primoris/Resources/PaperPdf.ashx?idPaper=73839VISAPP_2019_144_CR.pdf. Accessed March 2019
21. Fuentes, A., Im, D.H., Yoon, S., Park, D.S.: Spectral analysis of CNN for tomato disease identification. In: ICAISC 2017, Part I, LNAI, vol. 10245, pp. 40–51 (2017)