Engineering Journal issn: 0125-8281

Article

Potato Leaves Blight Disease Recognition and Categorization Using Deep Learning

Jesmin Akther[1,a], **Al-Akhir Nayan**[2,b,*], **and Muhammad Harun-Or-Roshid**[3,c]

1 Department of Computer Science and Engineering (CSE), European University of Bangladesh (EUB), Dhaka, Bangladesh
2 Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand
3 Softrobotics Bangladesh Limited, Dhaka, Bangladesh
E-mail: [a]jesminnipu1@gmail.com, [b,*]asquiren@gmail.com (Corresponding author), [c]md.parvez28@gmail.com

Abstract. Potato cultivation is vital in numerous countries, contributing to food security and economic value. However, crop diseases, particularly early and late blight, pose significant challenges to potato production. The accurate diagnosis of these diseases remains unclear to many individuals. This study leverages the increasing penetration of smartphones and recent advancements in deep learning to develop a Convolutional Neural Network (CNN) model for real-time detection of early and late blight in potatoes. The dataset was pre-processed by normalizing, dividing, and extracting images using the Python data processing library. The approach incorporates slight variations in the network layers to optimize the model's performance. The method was evaluated using classification optimizers, metrics, and loss functions and further refined using layer-by-layer TensorBoard analysis. Hyperparameters such as features, labels, validation split, batch size, and training epochs were carefully selected. The final model demonstrated promising results, achieving an accuracy of 96.09% on the survey dataset. Experimental findings highlight the approach's potential for automatically detecting both early, late blight and healthy, thereby significantly improving the accuracy of disease diagnosis.

Keywords: Convolutional neural network, early blight, late blight, potato disease, deep learning.

ENGINEERING JOURNAL Volume 27 Issue 9
Received 12 June 2023
Accepted 11 September 2023
Published 30 September 2023
Online at https://engj.org/
DOI:10.4186/ej.2023.27.9.27

DOI:10.4186/ej.2023.27.9.27

1. Introduction

Potatoes are a significant agricultural commodity in Bangladesh, with a high export volume of over 27,811.6 tons in 2013-14, making it the world's fourth-largest food crop [1]. However, the potato industry faces challenges due to the prevalence of destructive diseases such as brown rot, potato tuber moth, and blight disease. Among these, late blight is the most common and devastating fungal disease, causing substantial losses ranging from 25% to 57% in crop yield within Bangladesh. On the other hand, early blight manifests as small, circular, or irregular dark brown to black spots on older leaves, gradually enlarging up to 3/8 inch in diameter and assuming angular shapes [2]. Late blight causes the leaves and tubers of potatoes to decay and turn brown or black quickly (Fig. 1). It affects both the health and appearance of the potato plants. Flour-like spots on the undersides of leaves can identify infections. If left unchecked, infected plants can perish within days [2].

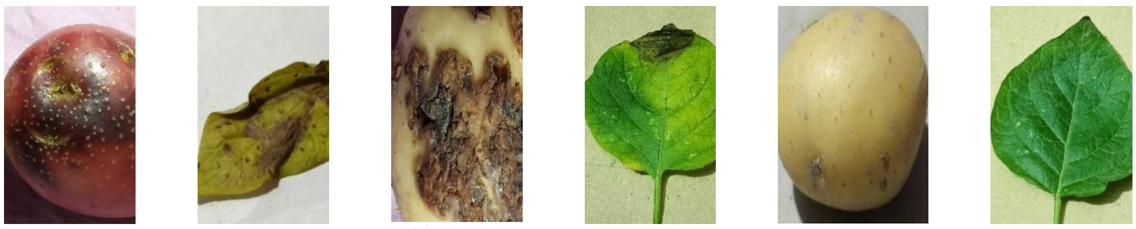

a) Early blight potato (b) Early blight leaf (c) Late blight potatoes (d) Late blight leaf e) Healthy potato f) Healthy leaf

Fig. 1. Affected Potatoes and Leaves.

Computer vision has witnessed significant progress recently, as demonstrated by benchmarks like the PASCAL VOC Challenge and ILSVRC. One particularly advanced area is deep convolutional neural networks (CNNs). These networks have achieved remarkable results, reducing classification error rates from 16.4% to 3.57%. The availability of large-scale annotated datasets has played a crucial role in enabling the accurate identification and classification of plant diseases using CNNs. Furthermore, transfer learning has proven effective using pre-trained CNN models such as VGGNet, ResNet, and InceptionNet, trained initially on ImageNet. By fine-tuning these models for plant leaf disease datasets, we can enhance performance while reducing the training requirements. These advancements have opened new possibilities for accurate and efficient plant disease detection and classification.

Previous transfer learning-based studies on potato blight disease classification have shown unsatisfactory performance due to several factors, such as limited training data, annotation errors, lack of interpretability, generalization issues with unseen variations, and a dependency on image quality. These drawbacks have hindered the accurate detection and classification of potato blight diseases. We employed a convolutional neural network (CNN) approach to address the challenges encountered in previous studies. Our dataset comprised 4,000 images representing various potato crop species, including diseased and healthy plants. We experimented with different layer sizes and utilized TensorBoard visualization to fine-tune the model parameters and improve accuracy in predicting the correct potato-disease pair. After a thorough evaluation, our best-performing model achieved an impressive accuracy of 0.9609 (96.09% overall accuracy). These results highlight the technical feasibility of our approach and lay the foundation for developing a smartphone-assisted plant disease diagnosis system.

The subsequent sections of this paper are organized as follows: section 2 provides a comprehensive literature review, highlighting the limitations and challenges associated with disease classification methods. In section 3, we present the experimental setup for potato disease detection. Section 4 focuses on disease classification and the development of the proposed model. Finally, in section 5, we summarize the conclusions drawn from this research and outline potential future directions in the field.

2. Related Works

Several studies were conducted to classify plant leaf diseases using CNN and deep learning. One study explored the application of deep CNNs and pooling techniques for identifying rice diseases, achieving an impressive accuracy of 95.48% [3]. The experiment evaluated different pooling methods, such as max pooling, mean pooling, and stochastic pooling. Results indicated that stochastic pooling outperformed the other ways in terms of accuracy. However, the study did not address the optimal number of layers and neurons required for optimal performance, highlighting a limitation in the research. Additionally, the availability of high-quality disease samples and the need for faster and more efficient algorithms were identified as areas for improvement.

Jihen Amara et al. [4] proposed a deep learning-based approach for classifying banana leaf diseases, specifically targeting Banana Sigatoka and Banana speckle. Their system effectively handled challenging conditions, including variations in illumination, complex backgrounds, and various image resolutions, sizes, and orientations. Although the approach demonstrated high accuracy, the study did not provide precise accuracy percentages or timing efforts, which is a limitation that needs to be addressed.

Hughes et al. [5] demonstrated the technical feasibility of using deep learning approaches to identify diseases in 14 crop species with 26 diseases or health conditions. They employed a deep learning model on a large dataset comprising 54,306 images. However, the complexity of extracting relevant features from the pictures was acknowledged as a limitation in the research.

Usama Mokhtar et al. [6] proposed a method for identifying healthy and infected rice leaves, achieving an impressive accuracy of 99.83%. Their approach involved preprocessing the input image by removing the background and eliminating noise using erosion techniques. Texture feature extraction was performed using the Gray Level Co-occurrence Matrix (GLCM) and classification using a Support Vector Machine (SVM). However, the study did not focus on distinguishing between different types of diseases or providing detailed information about the identified condition.

Zhang et al. [9] studied Multi-Task Learning for Food Identification and Analysis using deep CNN networks. While the study demonstrated the potential of deep CNNs for disease recognition, using Support Vector Machine (SVM) classifiers raised concerns about the process's time-consuming and less automated nature. Additionally, issues related to data security arose when user-provided data was used.

Most of the studies showed data and method issues and achieved unsatisfactory results. In this study, we have tried to overcome those issues by incorporating max pooling, experimenting with various architectures, and adjusting layers and neurons. Adding convolutional layers of varying sizes solved the feature extraction complexity issue. Data augmentation techniques increased data diversity, leveraging learned feature representations from our dataset, reduced complexity, and explored different architectures.

3. Methodology

3.1. Dataset Collection and Processing

The dataset consists of approximately 3,500 images, with 2,940 pictures allocated for training purposes, while the remaining photos are reserved for testing. The data collection process focused on two villages between two districts, namely Shimulia and Kajir Pagla Louhajong, located in Noakhali, Sonaimuri-3827 and Dhaka, Munshigang-1530. The images in the dataset were captured manually using smartphones, including devices such as iPhone 6, iPhone 7, Alcatel, and Redmi Note 10s, representing both Android and iOS platforms. The images encompass both healthy potato leaves and those affected by potato blight disease. Figure 2 showcases the survey datasets containing images of potato blight disease and healthy leaves, visually representing the dataset composition.

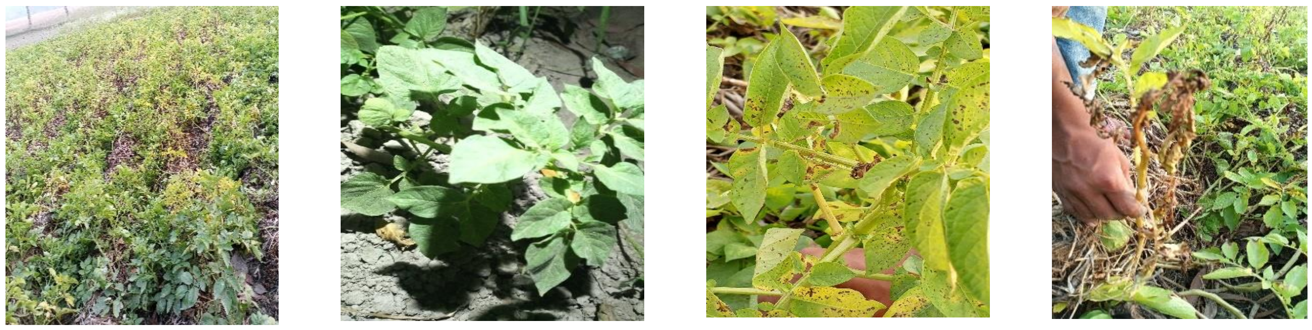


Fig. 2. Field Survey of Potato Leaves.

The images were captured in the RGB color space and saved in JPG format. For compatibility with CNN, image size, color, features, and capacity [11] were standardized. The images were modified using online web applications such as BulkreSize and the default editor in Windows, resulting in a size of 224 x 224 pixels and a noise-free JPG format. To preprocess the data, cloud storage and GPU support were utilized. The dataset was divided into three classes for training and testing: Early Blight, Late Blight, and Healthy. The test class data was used to evaluate the performance of the model. The training class contained 980 images each for early blight, late blight, and healthy categories.

The image data was processed using Python programming in Google Colab. The images were plotted and checked for a resolution of 224 x 224 pixels using libraries such as OpenCV and Matplotlib. They were then converted into image arrays of size 70 x 70 pixels and read in single-channel format. Care was taken to ensure all images were unique and not duplicates of other classes. The data were indexed and converted into categorical directories to facilitate training. The training data was shuffled using the random library [13]. The shuffle library was imported to mix the training data. Indexing provided validation results represented as 0 or 1.

Before deploying CNN, each image's features and labels were saved in the cloud. The image arrays were reshaped using a Python function: Data = image array * (-1, 70, 70, 1). The last number, 1, indicates that the images are grayscale. After reshaping the features, both the features and labels were normalized. Each pixel value was divided by 255.

3.2. Proposed Model

This study employed a sequential model [14] with two convolutional layers and an Adam optimizer, utilizing features and labels stored in the cloud. The model was trained with a batch size of 32, and validation data was consistently used throughout the training process. TensorBoard and GPU support were leveraged to optimize the model. The final model was then employed to predict the selected data. Figure 3 illustrates the proposed architecture of the CNN model.

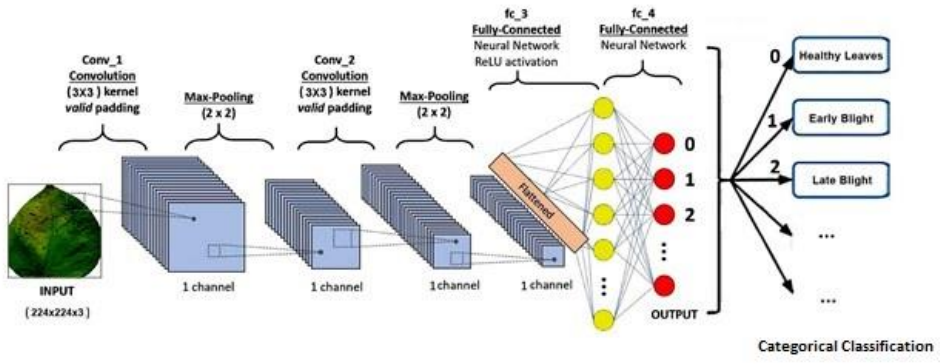


Fig. 3. Proposed CNN Model Architecture.

The CNN model comprises a sequential model with two hidden layers, incorporating Conv2D and Pooling layers. The first hidden layer performs convolution on the input data using 256 filters, applying padding with strides and a kernel of size 3x3. It is followed by a Leaky ReLU activation function and max pooling with a window size 2x2. This architecture aims to extract relevant features from the input data, address the dying ReLU problem, and reduce spatial dimensions through max-pooling before applying the ReLU activation function.

The second hidden layer mirrors the structure of the first hidden layer, conducting convolutional operations to extract features further. The data is then flattened into a one-dimensional format before passing through a fully connected dense layer with 512 neurons. Finally, it goes through another thick layer with a single neuron utilizing sigmoid activation, generating a probability score between 0 and 1. This score represents the model's prediction. The model is compiled with Categorical cross-entropy loss and the Adam optimizer, while performance evaluation is based on accuracy.

During training, the model was trained on the input data (trained features) and target labels (trained labels) for twenty epochs, utilizing a batch size of 32. A validation split of 30% was employed to monitor generalization and assist in detecting overfitting. The validation data is also instrumental in tuning hyperparameters through TensorBoard evaluation before selecting the final model.

4. Result and Discussion

4.1. Training Environment Setup

The proposed techniques and algorithms were employed to train the model, enabling it to learn patterns and features associated with different classes of potato blight disease. The training process involved optimizing the model's parameters for accuracy and performance. The training was conducted on a dataset, utilizing a GPU with a training speed of 1.86 IT/S for the early blight image class and 2.66 IT/S for the late blight image and healthy leaves classes. The GPU efficiently utilized its memory for accelerated image generation, stored in a frame buffer designed for display output. In this case, the training was performed on a laptop as the display device, as illustrated in Fig. 4.

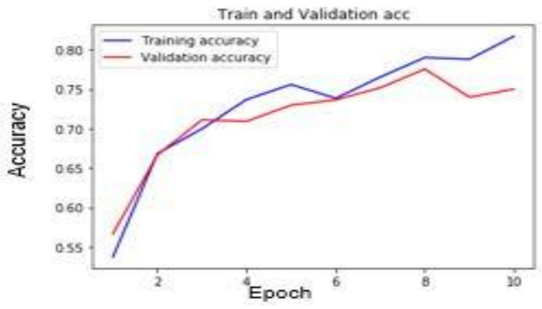
```
100%|██████████| 980/980 [00:02<00:00,
468.06it/s]
100%|██████████| 980/980 [00:02<00:00,
456.08it/s]
100%|██████████| 980/980 [00:10<00:00,
89.51it/s]
```

Fig. 4. Speed During Training.

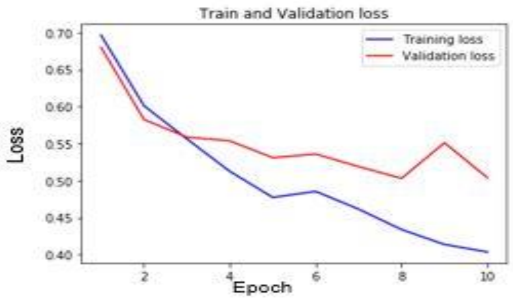
The dataset was divided into training and validation sets, comprising 2940 samples. The division of samples between the two sets was not specified. The dataset underwent processing in three stages, each denoted by a progress bar. The processing speeds for the first, second, and third bars were approximately 468.06, 456.08, and 89.51 samples per second, respectively. Of the 2940 samples, 2,940 were utilized for training, and the Shuffle library was imported to randomize the training data. The randomized training data list was then fed into the CNN model to ensure the reliability and accuracy of the training process.

4.2. CNN Deployment

Following three epochs of training, we achieved a validation accuracy of 78% on the dataset. To further enhance the validation accuracy, we decided to rerun the CNN deployment and train the model on 2,100 samples while validating it on 900 samples. The dataset was divided such that 30% was allocated for validation and 70% for accuracy testing, enabling us to conduct a more comprehensive analysis. It is important to note that the model presented at this stage is not the final version. This phase aimed to evaluate the training and testing data splitting results, illustrated in the Accuracy and Loss graphs depicted in Fig. 5.



(a) Accuracy Graph



(b) Loss graph

Fig. 5. Data Accuracy and Loss graph.

Figure 5 comprises two subplots: (a) Accuracy Graph and (b) Loss Graph. In Graph 5 (a), the validation accuracy closely aligns with the training accuracy, indicating that the model achieved an accuracy of over 80% after three epochs of training. Conversely, Graph 5 (b) represents the epochs on the horizontal axis and the loss percentage on the vertical axis. The training loss is depicted in blue, while the validation loss is shown in red, peaking at approximately 75%. A lower loss value indicates better validation of the data. After three epochs, the training loss is around 40%, while the validation loss is approximately 50%.

4.3. Analyze and Optimize Model

Various metrics such as accuracy, precision, recall, and F1 score were calculated to assess the model's performance in classifying potato blight disease samples. Visualizations and graphs were employed to gain insights into the model's decision-making process. Furthermore, the model underwent optimization by fine-tuning its hyperparameters. Techniques like grid or random search were utilized to find the optimal combination of hyperparameters that improved the model's performance. The optimization process aimed to enhance the model's accuracy and robustness. This study evaluated the model's accuracy using the TensorBoard callback. The top 10 layers were analyzed using TensorBoard to determine the best layers for the model. Based on this evaluation, three layers were selected for further analysis, and their corresponding validation accuracy is presented in Table 1. Among these layers, the "1-conv-64-nodes-0-dense-1583741819" layer achieved an accuracy of 90% and a validation accuracy of 88%. The optimization efforts were primarily focused on adjusting the layers and parameters to finalize the model.

Table 1. Three best layers table.

Layers	Validation Accuracy	Accuracy
1-conv-64-nodes-0-dense-1583741819	88%	90%
2-conv-256-nodes-0-dense-1583741819	83%	86%
3-conv-64-nodes-0-dense-1583741819	78%	81%

In this study, TensorBoard was incorporated into Keras using a Keras callback. The TensorBoard callback was employed to analyze the model's accuracy hierarchy, including adding an activation function to the dense layer. The log_dir argument was utilized to specify the directory path where the log files were saved for subsequent parsing in TensorBoard. Table 2 summarizes the model implementation using Keras with 64 convolutional layers. The provided information reflects the summary of a small neural network model trained using Keras, as evident from the callback history.

DOI:10.4186/ej.2023.27.9.27

Table 2. Model Summary.

Model: Sequential		
Layer(type)	Output Shape	Param #
Conv2d(Conv2D)	(None,68,68,64)	2560
Leaky_re_lu(LeakyReLU)	(None,68,68,64)	0
Max_pooling2d(MaxPooling2D)	(None,34,34,64)	0
Conv2d(Conv2D)	(None,32,32,64)	590080
Leaky_re_lu(LeakyReLU)	(None,32,32,64)	0
Max_pooling2d(MaxPooling2D)	(None,16,16,64)	0
flatten (Flatten)	(None, 65536)	0
dense (Dense)	(None, 64)	4194368
dense (Dense)	(None, 1)	65
activation(Activation)	(None, 1)	0

Table 3. Model Validation Accuracy.

Total params: 4,787,073
Trainable params: 4,787,073
Non-trainable params: 0
Epoch 1/3 65/65 - 6s - loss: 0.3477 - accuracy: 0.6667 - val_loss: 0.0297 - val_accuracy: 0.7879 - 6s/epoch - 54ms/step
Epoch 2/3 65/65 - 3s - loss: 0.3337 - accuracy: 0.6223 - val_loss: 0.0207 - val_accuracy: 0.7879 - 3s/epoch - 31ms/step
Epoch 3/3 65/65 - 2s - loss: 0.3007 - accuracy: 0.6223 - val_loss: 0.0199 - val_accuracy: 0.7879 - 2s/epoch - 30ms/step
<keras.callbacks.History at 0x7fdfb015a4d0>

The training process involved three epochs, each representing a complete iteration through the training dataset. Throughout the training, the model's loss and accuracy were continuously monitored. The loss value indicates the model's performance, with lower values suggesting better performance. Accuracy, however, measures the proportion of correctly classified samples during training.

For each epoch, the model underwent training on the training dataset and evaluation on the validation dataset. Table 3 showcases the training and validation metrics for selected epochs, offering insights into the model's performance progression. The model's accuracy remains steady at 0.6223 throughout the training process, while the loss slightly decreased from epoch 1 to epoch 3. However, the validation accuracy remained constant at 0.7879, indicating a 78% validation split.

To optimize the model, various parameters were adjusted, such as layer size and node count. Different configurations were evaluated, and the validation accuracy was employed as the benchmark for comparison. Increasing the layer size to 128 improved performances, achieving an 80% validation accuracy. Further experimentation with a layer size of 256 yielded even better results, reaching approximately 95% validation accuracy. Table 4 presents the parameters used during practical use, including loss, optimizer, metrics, batch size, epoch, and validation split. These parameters were fine-tuned to optimize the model's performance.

Table 4. Parameter table.

Function Name	Keyword Params	Measure Value
Compile	Loss Optimizer metrics	Percentages Adam Percentages
Cov2D	Leaky ReLU alpha Strides Padding	0.2 1 valid
Model Fit	Feature label Batch Size Epoch Validation Split	Horizontal Pixel values Vertical Pixel values 32 20 0.3%

During the training process, the model's performance was assessed using essential metrics such as loss, accuracy, validation loss, and validation accuracy. The results for the top five layers are displayed in Table 5, and a corresponding bar graph in Figure 6 illustrates the model's performance across different epochs.

Table 5. Five best layer result.

Loop	Loss	Accuracy	Val_loss	Val_accuracy
16/20	0.0633	0.9786	0.1573	0.9418
17/20	0.0672	0.9745	0.1220	0.9518
18/20	0.0622	0.9724	0.1376	0.9590
19/20	0.0391	0.9878	0.1305	0.9601
20/20	0.0173	0.9949	0.1439	0.9609

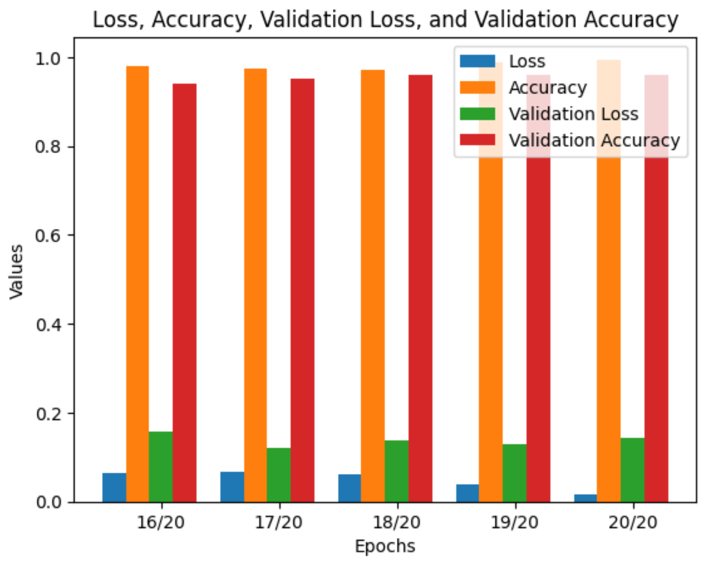
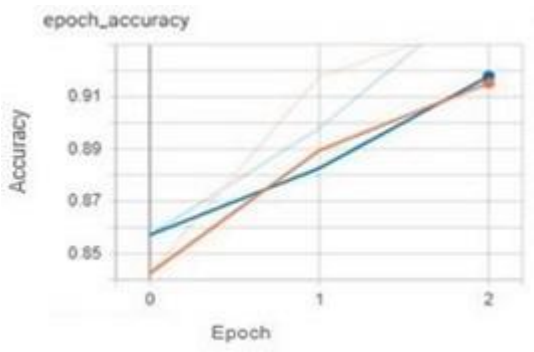


Fig. 6. Complete Model Visualization in Bar Graph.

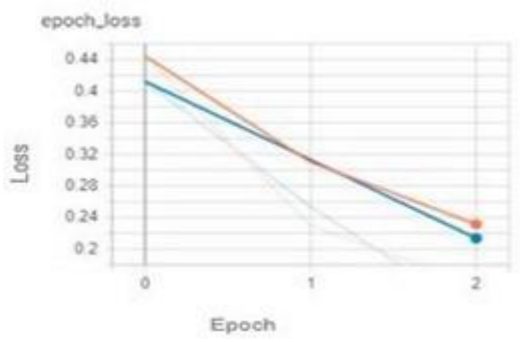
Figure 6 provides a visual representation of the metrics, namely loss, accuracy, validation accuracy, and validation loss, using distinct colors: blue for loss, orange for accuracy, green for validation accuracy, and red for validation loss. The left vertical margin depicts the values in percentages, while the bottom vertical axis corresponds to the last five epochs out of a total of 20 epochs. Starting from epoch 16, a gradual increase in validation accuracy can be observed, reaching approximately 96%, while the accuracy remains consistently above 99%. The validation loss shows fluctuations during these five epochs, with the loss going to its minimum in the final epochs.

TensorBoard evaluation output was utilized and visually represented in Fig. 7 to assess the model's predictions. This evaluation was performed on 30% of the available data, and the results were compared with TensorBoard. The progression of accuracy and loss over epochs is illustrated in Figs. 7(a) and 7(b), respectively. Furthermore, Figs. 7(c) and 7(d) showcase graphical representations of accuracy and loss obtained without employing TensorBoard. The results from the model's predictions and the monitoring of its performance using TensorBoard demonstrate the effectiveness and progress of the model throughout the training process.

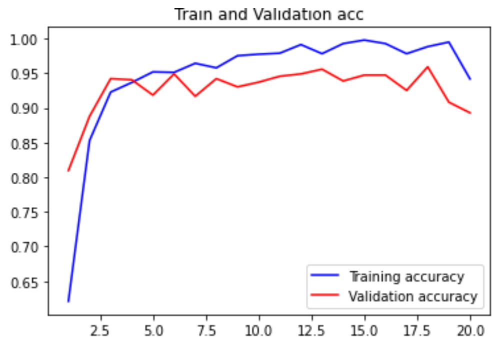
DOI:10.4186/ej.2023.27.9.27



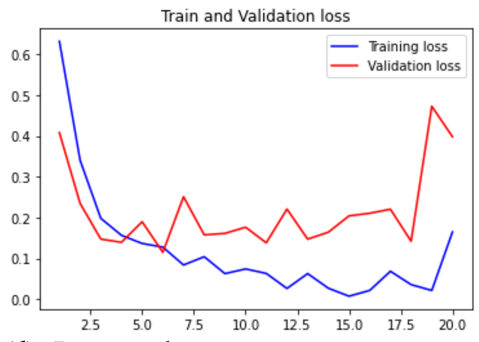
(a) Epoch accuracy



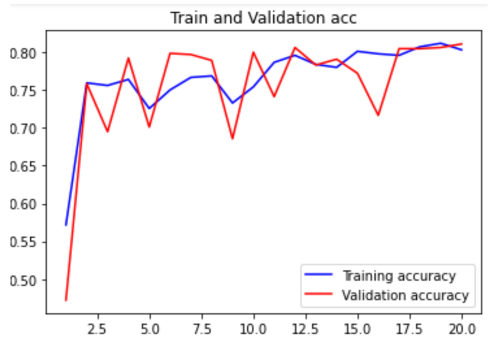
(b) Epoch loss



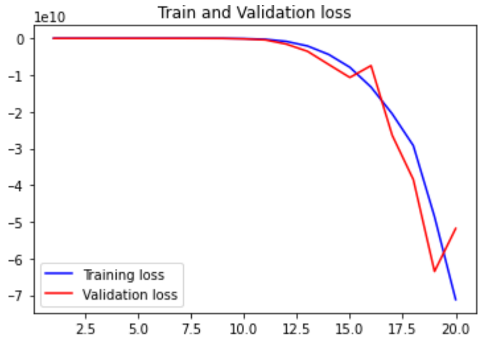
(c) Accuracy Graph



(d) Loss graph



(e) Accuracy Graph



(f) Loss graph

Fig. 7. Complete Model Visualization in TensorBoard with Graphs.

In Fig. 7(e), the percentage accuracy is represented on the vertical axis, while the horizontal axis corresponds to the epochs. The graph showcases a blue line indicating the progress of model training accuracy, which reaches 93%. Additionally, a red line represents the validation accuracy, which reaches 96%. Figure 7(f) exhibits the loss in percentage on the vertical axis, with the epochs displayed on the horizontal axis. The graph features a blue line representing the training loss, 0.2, and a red line representing the validation loss, slightly above 0.24. To monitor the model's progress, the same TensorBoard backend is reused by issuing a command, enabling continuous tracking of the model's performance.

4.4. Proposed Model's Performance on PlantVillage Dataset

The dataset used in this study, PlantVillage, consists of 54,303 images of leaves that have been categorized into 38 groups based on species and disease. These categories encompass a range of both healthy and diseased samples. For our specific analysis, we focused on three classes from this dataset: potato healthy, early blight, and late blight. The outcomes of our model's performance on this dataset are presented in Table 6 and Fig. 8.

Table 6. Five Best Layer Results of Platvillage Dataset.

	loss	accuracy	val_loss	val_accuracy
16/20	0.0428	0.9847	0.4443	0.8748
17/20	0.0439	0.9840	0.2562	0.9285
18/20	0.0607	0.9784	0.1690	0.9431
19/20	0.0562	0.9819	1.0119	0.8098
20/20	0.0434	0.9833	0.6378	0.8699

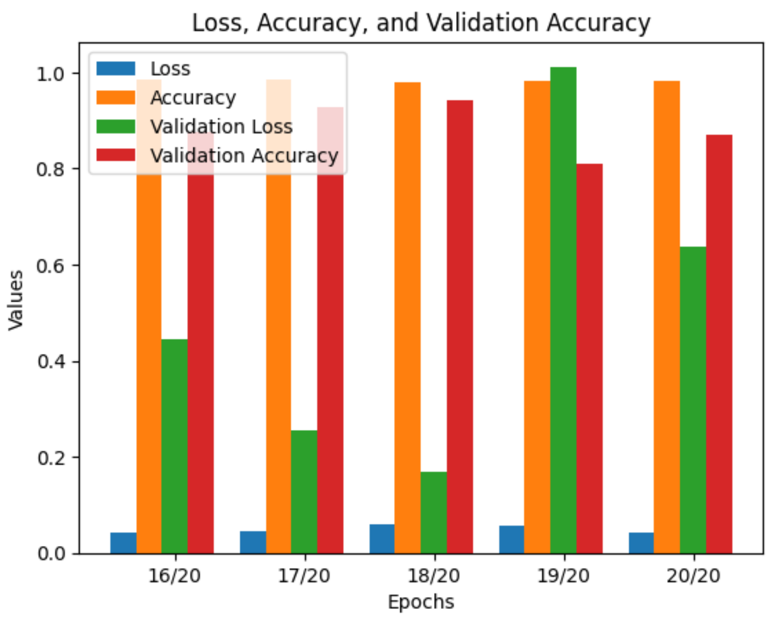


Fig. 8. Complete Model Visualization in a Bar graph for the Plant Village Dataset.

Figure 8 visually represents different metrics, such as loss, accuracy, validation accuracy, and validation loss, using distinct colors: blue for loss, orange for accuracy, green for validation accuracy, and red for validation loss. The left vertical margin represents the values in percentages, while the bottom vertical axis focuses on the last five epochs out of a total of 20 epochs. Starting from epoch 16, the validation accuracy exhibits fluctuations throughout the remaining epochs, reaching approximately 87%, while the accuracy remains consistently high at around 98%. On the other hand, the validation losses experience a decline from epoch 16 to epoch 18, followed by a sharp increase in the subsequent epoch, reaching nearly 99%. However, there is a sudden decrease in the final epochs, resulting in a value of 64% within the five selected epochs. The loss bar stabilizes in the concluding epochs, indicating a constant value.

We compare our model's performance on the surveyed samples (Table 5) with the PlantVillage dataset (Table 6) using loss and validation accuracy as metrics. Figure 9 shows the overall evaluation of our model.

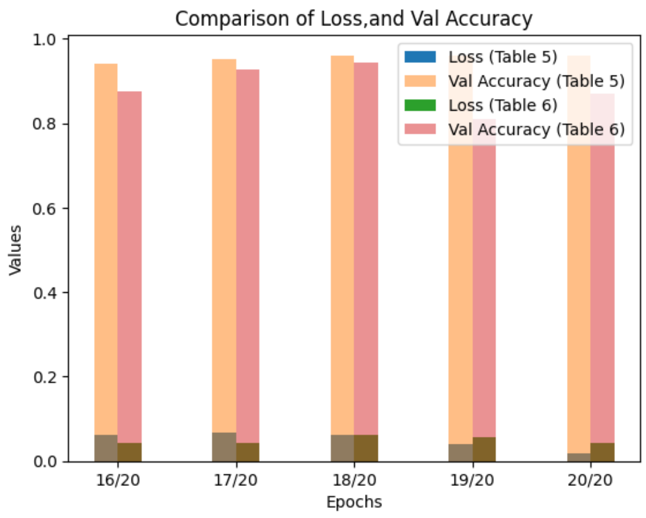


Fig. 9. Loss vs. Accuracy for plant Village Dataset.

Figure 9 visually illustrates the performance of different metrics, specifically the loss and validation accuracy, as presented in Tables 5 and 6. For Table 5, the loss metric is represented by blue, while a dull orange indicates the validation accuracy. For Table 6, the loss metric is in green; the validation accuracy is shown in pink. The left vertical margin of the graph represents the values in percentages, while the bottom vertical axis focuses specifically on the last five epochs out of the total 20 epochs. It is evident that our model, trained on our dataset, achieved minimal loss from epoch 16 to epoch 20, steadily reaching the highest validation accuracy above 96%. The PlantVillage dataset experienced an accuracy of 87% and higher loss values compared to our model's performance.

4.5. Experiment Analysis on Related Approach

During our investigation, we extensively studied the dataset consisting of potato leaves and employed it for training and testing. Additionally, we explored the effectiveness of various models, including VGG-16, Inception V3, ResNet50, and Efficient Net, by training them for twenty epochs.

Table 7. The Surveyed Dataset with Pre-Trained Models.

Pre-Trained Models	Dataset	Epoch	Loss	Accuracy	Validation Loss	Validation Accuracy
VGG-16	Own data	20	0.2508	0.8980	0.8135	0.6939
Inceptionv3	Own data	20	1.2156	0.7834	0.8139	0.6291
ResNet50	Own data	20	0.1246	0.9784	0.7129	0.6480
EfficientNet	Own data	20	0.0567	0.9854	1.1020	0.5000

All the models performed relatively poorly compared to our CNN model validation accuracy. The visual comparisons are shown in Figure 10.

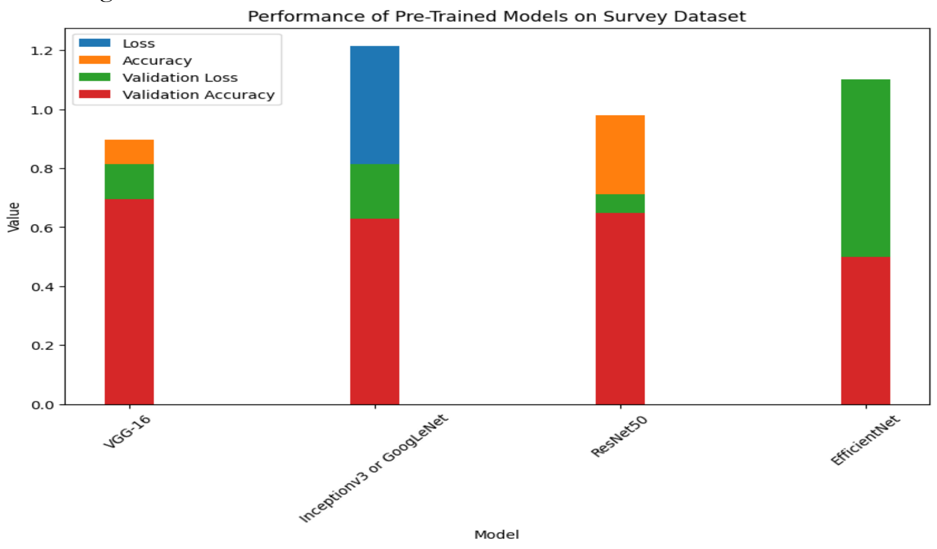


Fig. 10. Complete Model Visualization in a Bar graph for Plant Village Dataset.

Figure 10 illustrates the behavior of various metrics using different colors, such as loss, accuracy, validation, and validation loss. The loss is represented in blue, accuracy in orange, validation loss in green, and validation accuracy in red. The left vertical margin denotes the values in percentages, while the bottom vertical axis focuses specifically on the last five epochs out of the total 20 epochs. These metrics offer valuable insights into the performance of the pre-trained models on the surveyed dataset, showcasing their loss values, accuracies, and generalization capabilities on a separate validation dataset.

5. Conclusion

Developing a Convolutional Neural Network (CNN) model for real-time detection of early and late blight in potatoes has shown promising results. By leveraging smartphones and recent advancements in deep learning, this study addresses the challenges of crop diseases and offers a potential solution to improve potato production. Through the dataset preprocessing and optimization of the network layers, the model achieved an impressive accuracy of 96.09% on the survey dataset. These experimental findings emphasize the potential of this approach in automatically detecting both early and late blight, as well as healthy potatoes, thereby significantly enhancing the accuracy of disease diagnosis. The application of this technology can contribute to food security and economic value in countries where potato cultivation is crucial. The possibility of creating advanced models that can diagnose and treat plant diseases directly on smart devices exists in future endeavors. This endeavor may encompass exploring and experimenting with novel architectures to improve the model's performance when trained on specific datasets.

References

[1] M. A.-M. Molla. “Potato glut badly hurts growers.” The Daily Star. Accessed: June 10, 2023]. [Online]. Available: https://www.thedailystar.net/backpage/potato-production-in-bangladesh-glut-badly-hurts-growers-1748572

[2] S. Murmu, *Early Blight Disease of Potato and Its Management*. LAP Lambert Academic Publishing, 2019.

[3] Y. Lu, S. Yi, N. Zeng, Y. Liu, and Y. Zhang, “Identification of rice diseases using deep convolutional neural networks,” *Neurocomputing*, vol. 267, pp. 378–384, 2017.

[4] J. Amara, B. Bouaziz, and A. Algergawy, “A deep learning-based approach for banana leaf diseases classification,” in *Datenbanksysteme für Business, Technologie und Web (BTW 2017) - Workshopband. Bonn: Gesellschaft für Informatik e.V. (S. 79-88)*, B. Mitschang, D. Nicklas, F. Leymann, H. Schöning, M. Herschel, J. Teubner, T. Härder, O. Kopp, and M. Wieland, (Hrsg.), Eds, 2017.

[5] D. P. Hughes and M. Salathe, “An open access repository of images on plant health to enable the development of mobile disease diagnostics,” 2015, *arXiv:1511.0806*.

[6] U. Mokhtar et al., “SVM-based detection of tomato leaves diseases,” in *Advances in Intelligent Systems and Computing*. Cham: Springer International Publishing, 2015, pp. 641–652.

[7] S. D. Khirade and A. B. Patil, “Plant disease detection using image processing,” in *2015 International Conference on Computing Communication Control and Automation*, 2015.

[8] S. Bashir, “Remote area plant disease detection using image processing,” *IOSR J. Electron. Commun. Eng.*, vol. 2, no. 6, pp. 31–34, 2012.

[9] X.-J. Zhang, Y.-F. Lu, and S.-H. Zhang, “Multi-task learning for food identification and analysis with deep convolutional neural networks,” *J. Comput. Sci. Technol.*, vol. 31, no. 3, pp. 489–500, 2016.

[10] T. Shi et al., “Recent advances in plant disease severity assessment using convolutional neural networks,” *Sci. Rep.*, vol. 13, no. 1, p. 2336, 2023.

[11] S. P. Mohanty, D. P. Hughes, and M. Salathé, “Using deep learning for image-based plant disease detection,” *Front. Plant Sci.*, vol. 7, p. 1419, 2016.

[12] J. S. Lim, *Two-Dimensional Signal and Image Processing*. Englewood Cliffs N.J: Prentice Hall; 1990.

[13] M. Vazan, “Deep learning: From basics to building deep neural networks with Python,” 2022, *arXiv: 2205.01069*.

[14] J. Y. Lee and F. Dernoncourt, “Sequential short-text classification with recurrent and convolutional neural networks,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

[15] K. Wongsuphasawat et al., “Visualizing dataflow graphs of deep learning models in TensorFlow,” IEEE Trans. Vis. Comput. Graph., vol. 24, no. 1, pp. 1–12, 2018.

[16] C. Szegedy et al., “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[17] A. Picon, A. Alvarez-Gila, M. Seitz, A. Ortiz-Barredo, J. Echazarra, and A. Johannes, "Deep convolutional neural networks for mobile capture device-based crop disease classification in the wild," *Comput. Electron. Agric.*, vol. 161, pp. 280–290, 2019.

[18] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, *arXiv:1409.1556*.

[19] J. Akther, M. Harun-Or-Roshid, A.-A. Nayan, and M. G. Kibria, “Transfer learning on VGG16 for the classification of potato leaves infected by blight diseases,” in *2021 Emerging Technology in Computing, Communication and Electronics (ETCCE)*, 2021.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[21] M. Shafiq and Z. Gu, “Deep residual learning for image recognition: A survey,” *Appl. Sci. (Basel)*, vol. 12, no. 18, p. 8972, 2022.

[22] P. Huilgol. “Top 4 pre-trained models for image classification with Python code.” Analytics Vidhya. Accessed: June 10, 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-classification-with-python-code

[23] M. Tan et al., “MnasNet: Platform-aware neural architecture search for mobile,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[24] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for Convolutional Neural Networks," in *36th International Conference on Machine Learning, ICML 2019*, June 2019, pp. 10691–10700.

[25] M. Sandler et al., “Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation,” CVF, 2018.

[26] J. Chen et al., “An efficient memristor-based circuit implementation of squeeze-and-excitation fully convolutional neural networks,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 4, pp. 1779–1790, 2022.

Jesmin Akther has been a Lecturer at the Department of Computer Science and Engineering (CSE), European University of Bangladesh (EUB) since December 2020; in addition, she was a lecturer at Uttara Institute of Business & Technology from January to December 2020, Dhaka, Bangladesh. She has completed his graduation and post-graduation from the Department of Information and Communication Engineering (ICE), Noakhali Science and Technology University (NSTU), Bangladesh. She was the University Rank Holder (Merit Position: 3rd) at the Postgraduate level. She has multiple publications in international journals, including reputed publishers like IEEE, Springer, and HBRP Publication. Her Area of interest includes deep learning, machine learning, android component-based software engineering, Tools and environments of Application Development, Internet of Things (IoT), Software Defined Networking (SDN), Fog Computing, and Edge Computing.

Al-Akhir Nayan received his Bachelor of Science in Computer Science and Engineering (CSE) from the University of Liberal Arts Bangladesh in 2019 and a Master of Engineering degree in Computer Engineering from Chulalongkorn University, Thailand, in 2023. He joined the European University of Bangladesh (EUB) in August 2019 and worked as a Lecturer with the CSE department. His research interests include computer vision, deep learning, machine learning, medical imaging, and the IoT.

Muhammad Harun-Or-Roshid has been working as a Senior Software Engineer at Softrobotics Bangladesh Limited in Dhaka, Bangladesh since July 1, 2022. He has extensive experience as a Software Developer in Android and game development, dating back to 2017. Muhammad completed his graduation from the Department of Computer Science and Engineering (CSE) at Daffodil International University (DIU) in Bangladesh. His expertise has led to multiple publications in reputable international journals, including IEEE and HBRP Publication. His areas of interest encompass game development, software-defined networking (SDN), fog computing, edge computing, deep learning, machine learning, Android component-based software engineering, tools and environments for application development, and the Internet of Things (IoT).