Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 46 (2025), No. 1, pp. 1-14
DOI : 10.47974/JIOS-1846

Hyperparameter tuning of CNN based potato plant disease detection model integrated with GridSearchCV

Kajal Kaul *
Anuradha Chug †
Amit Prakash Singh §
University School of Information, Communication and Technology
Guru Gobind Singh Indraprastha University
Dwarka 110078
New Delhi
India

Ravinder Kumar ‡
V. Shanmugam @
Division of Plant Pathology
Indian Council of Agricultural Research - Indian Agricultural Research Institute
Pusa 110008
New Delhi
India

Abstract

There is an increasing need for enhancing plant health in the potato crop since Potato is the staple diet for majority of population living in Asia. Potato crop yield is of prime importance for farmers and agriculturalist. The Deep Convolution Neural Network model has been optimized on the PV dataset for classification, detection and prediction of various Potato Plant Diseases like PEB, PLB and distinguishing it from healthy potato leaves which served as the class labels. Images were then preprocessed using various Data augmentation techniques for enhancing the computational efficacy of the models. This paper aims at comparing various hyperparameter tuned cases for hyperparameters namely optimizer, learning rate, epochs, keeping batch_size, filter_size, activation function and the number of channels fixed for CNN as the base model integrated with GridSearchCV technique. Amongst all the hyperparameters, it was found that optimizer Adam along with 0.0001

* *E-mail:* `kajalkaulphd@gmail.com (Corresponding Author)`
† *E-mail:* `anuradha@ipu.ac.in`
§ *E-mail:* `amit@ipu.ac.in`
‡ *E-mail:* `ravinder.kumar2@icar.gov.in`
@ *E-mail:* `shanpatho@yahoo.com`

learning rate, 3*3 filter size, 32 batch size, 100 epochs and relu activation function at dense layer and softmax function at fully connected layer performed the most optimal. The CNN base model was trained and tested to predict the output class. The efficiency of the model was measured by accuracies achieved for the model including train_accuracy, validation accuracy and test accuracy as 0.959, 0.960 and 0.988 respectively. The losses for the model such as train_loss, validation loss and test loss were 0.102, 0.100 and 0.039 respectively. After comparing all the model cases, Precision was found out to be 0.988, F1 score was found to be 0.988, Recall was 0.988, Roc-auc value was 0.563. To further comprehend the performance and select the most optimal model GridSearchCV technique was then applied for measuring performance with regards to achieving Generalization, Regularization and Optimization in the model. With a high cross-validation score of 0.78 indicates that the model can function well on unknown data, regularization as indicated by high value of C i.e. 31.622, is more tolerant of misclassifications, potentially leading to higher accuracy but potentially also lower sensitivity to detecting positive cases and optimization as the model's parameters were successfully optimized by the 'newton-cg' solver, resulting in strong performance.

Subject Classification: *68T05, 68T07.*

Keywords: *Deep convolution neural networks, GridSearchCV, Plant disease detection, Potato early blight, Potato late blight, Potato healthy leaves.*

1. Introduction

Plant Diseases are of varying types including fungal disease such as Alternaria Brown spot, Early Blight, Powdery Mildew, Fusarium Dry Rot, Verticillium Wilt aka Potato Early Die, White Mold, Cercospora Leaf Blotch, Common Rust, Late Blight. Bacterial diseases include common scab, Ring Rot, Soft Rot, zebra chip, Bacterial wilt. Also, viral diseases include Potato virus S, mild mosaic, potato virus Y, Potato leafroll virus, Potato virus X, Alfalfa virus. Amongst all the above-mentioned diseases, the Plant Village dataset uploaded on Kaggle by **Arjun Tejaswi** has been taken. The dataset comprised of 20600 image files in total of which Potato crop comprised of 2152 files. All the files are divided into three repositories namely PEB having 1000 image files, PLB having 1000 image files and PH having 152 image files.

2. Motivation

The research aims to assist farmers and agriculturists in developing plant-healthy practices for nutrient-rich harvests, as farmers face numerous challenges in growing crops and rely on their yield for income. Researchers have a responsibility to help farmers grow crops efficiently and increase output, contributing to India's GDP. The focus is on detecting potato plant diseases via the Deep CNN model integrated with GridSearchCV approach. In Deep CNN model, identifying the most optimal and fine-tuned hyperparameter model cases for maximum efficacy is the goal. This can aid agriculturists potato crop yield in many ways. The GridSearchCV algorithm is introduced to increase the model's efficacy in terms of metrices: the best parameters ('C' and' solver'), and the best Cross-Validation Score. These performance parameters help determine the model's generalizability, regularization, and optimization capabilities.

3. Literature Review

Table 1

Literature Review

Reference no.	Author	Dataset Used & Dataset size	Types of Potato Disease	Algorithm Used	Experimental Setup & Performance measures & Accuracy	Hyper parameters	Novelty added
[1]	Anuradha Chug & Amit Prakash Singh et al	PV dataset; Potato leaf images: 3000; Downscaled images: 256X256 to 32X32	PEB, PLB, PH	Capsule networks	GPU: Python 3.7 GTX1060 6 GB; Original CapsNet Accuracy: 91.83% Implementation: Keras.	Training process: Block-wise DCT Scaling; Converting colored images into MNIST data form;	Capsule networks; Compared the performance with pretrained CNN models executed via transfer learning namely: ResNet18; VGG16; GoogleNet
[2]	Al-Adhaileh et al.	PV dataset; Tot. images: 2152 images; 2652 images after Data balancing; Each image size: 256 × 256 pixels	PEB, PLB, PH	Data Augmentation; Data Balancing algorithm; Deep CNN	Accuracy: 99%	Epochs: 45; Data augmentation done. Optimizer: Adam; Trainable Parameters 839,203; LR: 0.001; ReLu used. Size of batch:32	Data balancing by: Increasing PH leaf images;

Contd...

[3]	Chen et al.	PV dataset; Tot. images: 3246; **Practical fields:** 812; PH samples: 363; EB samples:207; LB samples:109; Potato virus disease samples:133.	PEB fungus general; PLB fungus serious; PLB fungus general; PH	MobileNet-V2 with Atrous Spatial Pyramid Pooling (ASPP)	Recall rate: 91.99%; Accuracy: 97.33%; Specificity: 98.39%.	Learning rate: 1× 10–3a; Mini-batch size: 64; Epochs: 30; Optimizer: SGD.	MobileNet-V2 architecture; Altercated by implementing transfer learning; Feature extraction: Atrous convolution & SPP module; Hybrid channel attention & submodules: spatial attention.
[4]	Hassan et al.	PV dataset; Rice Dataset; Cassava Dataset	PEB, PLB, PH	Inception layer and Residual connection	Recall- 99.19; Precision- 99.17; F1Score- 99.18; **Accuracy:** Rice - 99.6%; Cassava -76.5%.	Train_accuracy: 99.81%; Val_accuracy: 99.39%; Train_loss: 0.0015; Val_loss: 0.0549 Epochs: 50;	Feature Extraction: Inception V3 and ResNet50 are good; Depth-wise separable convolution
[5]	Sharma et al.	Potato- 1500 images - 80:20 Train test split Training set: 1200 Testing set:300 Classes:3 Preprocessing stage: Resized images: 128 X 128 pixels	PEB, PLB, PH	CNN; 1500 potato leaf images	Accuracy:97.66 Precision: 98 Recall: 98 F1 Score:98	Optimizer: Adam; Accuracy: 97.66; Learning rate: 0.0001; Batch size: 32; Activation Function: Relu & Softmax; Loss: Sparse categorical cross entropy;	A CNN model with Batch normalisation layer is used in this paper.
[6]	D. Tiwari et al.	PV dataset; Total images:2152; Train:1700;Test:45; **EB:** Total:1000; Train:787; Test:213; **LB:** Total:1000; Train:791; Test:209; **PH:** Total:152; Train:122; Test:30	PEB, PLB	Transfer learning: VGG19 i.e. Feature extraction & Logistic Regression	Accuracy: 97.80%; Precision:97.8; AUC:99.99; CA:97.8; F1 Score:97.8; Recall:97.8	Activation Function: Relu, Softmax	Feature extraction: VGG16, Training: VGG19; Classifying Algo: SVM; KNN; NN & Logistic regression (accuracy: 97.8%.)

Contd...

[7]	Sinshaw et al.	spMohanty, PV Dataset; Healthy: 152 images, LB:1000; EB:1000; RGB format Resized image size: 256*256;	PEB, PLB, PH	MobileNet, and Efficient Net	Train Accuracy(EB): 98%; Train Accuracy (LB): 99%; Train Accuracy (Healthy): 96%; Tot. images: 2152; Training: 85% = 1829 images per each class, Testing: 15% = 323 images. Accuracy:98%	Batch size: 16; Learning rate: 0.00002 & Epoch: 50;	Model combined with Android Application & Tested with unknown data.
[8]	Polder et al.	The PVY infected potato batches of Hyperspectral images captured using Computer vision via the Dutch General Inspection Service (NAK) was picked for a significant level of infection.	Viral disease	FCN	**Experimental Setup:** Hyperspectral line-scan camera; Precision:0.78; Recall:0.88	Dropout p = 0.5;	Image augmentation techniques: Rotation; Image brightness; Random mirroring;
[9]	Mathur et al.	MangoleafDB. Image source: Mango trees in Ban gladesh; Total images: 4000; Classes: 8.	8 diseases	Inception V3, MobileNet V3 Small, MobileNet V3 Large, and ResNet50	Inceeption V3: 100% accuracy	Optimizer: ADAM; Batch size 16; Maximum Epochs: 30; Learning rate 0.0001; Weight decay: 1e-08	Preprocessing, optimal hyperparameter tuning
[10]	Bhoyar et al.	Dataset source: Indian government	22 crops were taken	Logistic Regression; Decision tree; Random Forest	Random Forest accurcy 97.87	Rainfall, temperature, N, P, K, and humidity	Performance metric: Confusion matrix & classification report

4. Proposed System Requirements

4.1 *Computational Requirements*

Google Colab with faster NVIDIA T4 Tensor Core GPU Python Google Compute Engine. Computation cost is 0.000228. System RAM is 10.7 / 51.0 GB, GPU.

5. Implementation

The workflow of CNN model integrated with GridSearchCV is shown in Figure 1:

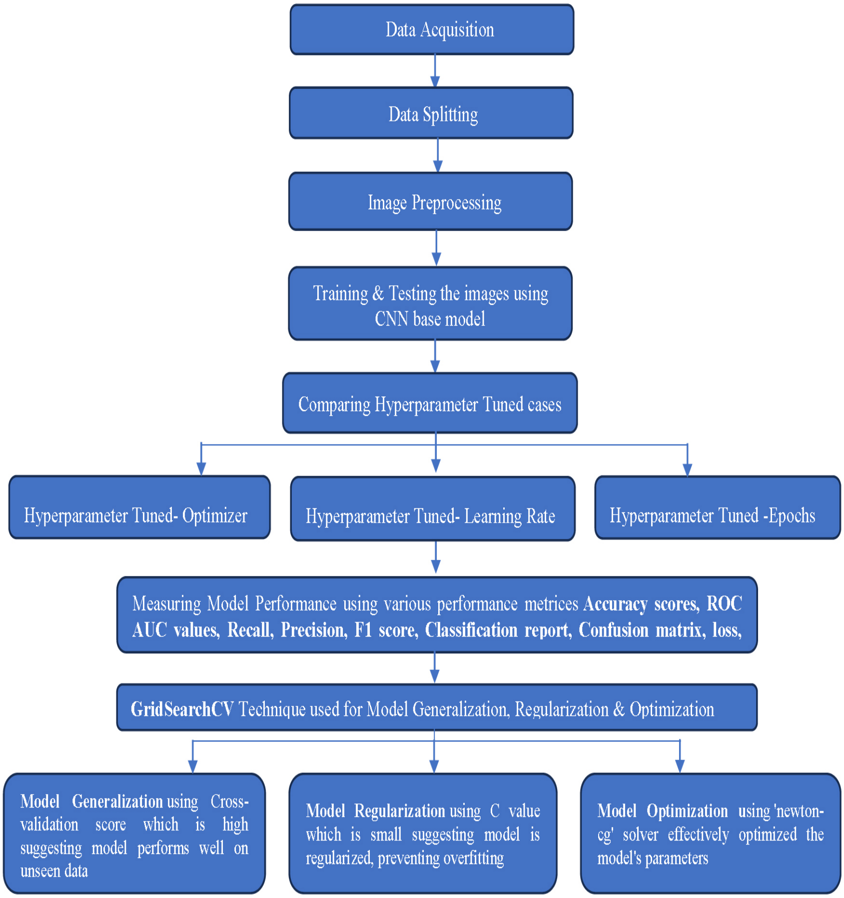

Figure 1

Representation of workflow of events

5.1 *Dataset Acquisition and Description*

This paper uses Deep CNN model architecture to predict potato plant disease using a PV dataset which is also one of the most widely used dataset as can be seen in Table 1. The dataset, uploaded by Arjun Tejaswi, includes 20600 images of plant potato, including 1000 PLB, 1000 PEB, and 152 PH plants. The dataset is plotted for the first 12 images in Figure 2.

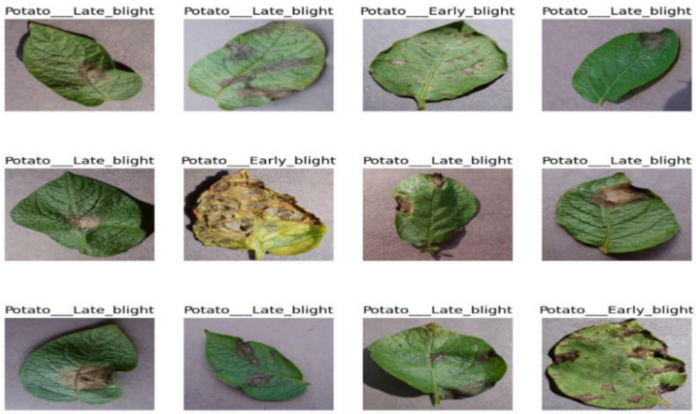

Figure 2

Representation of all three class labels PEB, PLB and PH leaves

5.2 *Image Preprocessing*

The original image size is 256*256, but the dataset is resized to 224x224 for the Sequential layer and data augmentation layer. The pixels are rescaled to 0-1, and the images are randomly transformed to improve computational efficiency and model generalization capacity.

5.3 *Data Splitting*

The data is then divided into testing, validation, and training. Having 80% of the data in train_ds, 10% in val_ds, and 10% in test_ds, the split ratio is maintained. With shuffle:True, train_split, val_split, test_split, shuffle_size: 10000 as the function parameters, the splitting function utilized was def get_dataset_partitions_tf().

5.4 *Deep CNN model*

The training image dataset is passed to sequential layers, 4 convolution layers, each with a max-pooling layer between them. They extract features using kernel size of 3*3, ReLU activation function and larger filter sizes (32, 64, 64, 64). Max-pooling layers minimize spatial dimensions. Then flattening of vectorized feature maps which is fed into 2 connected dense layers with 64 units. ReLu activation function is applied in the premier dense layer, and softmax with {n_classes} units is used in the last dense layer to produce the output class for every input picture across all classes. The architecture of Deep CNN model is represented in Figure 3.

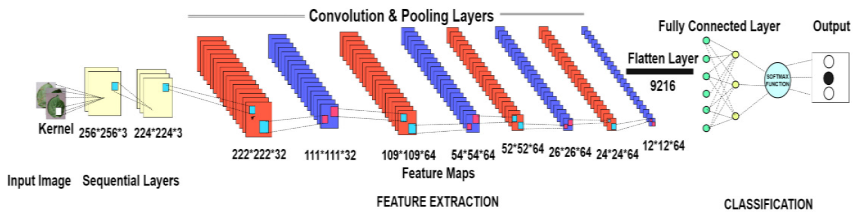

Figure 3

Representation of the Deep CNN model

6. Observations

The observations of all the comparisons made for all the cases based on the following parameters like the choice of optimizers, learning rate, epochs, batch_size, filter_size, activation function, etc. are given in Table 2.

Table 2

Cases with all different Optimizers

Case	**1**	**2**	**3**	**4**	**5**
Optimizer	**Adam**	**SGD**	**RMS Prop**	**AdaGrad**	**Nadam**
Learning Rate	0.0001	0.0001	0.0001	0.0001	0.0001
Epochs	50	50	50	50	50
Batch_size	32	32	32	32	32
Filter_size	3*3	3*3	3*3	3*3	3*3
Activation function	ReLu	ReLu	ReLu	ReLu	ReLu
Channels	3	3	3	3	3

The results are inclusive of performance metrices such as confusion matrix, accuracies, recall, roc-auc values, precision, classification report, f1 score & best parameters like inverse regularization, optimization solver and best cross validation score and loss are taken as follows:

From the above Table 2.1, ADAM optimizer gave us the best outputs in terms of all the performance criterions presented in the table, so further enhancements of the hyperparameters of the Adam based CNN model is preferred. The use of different learning rates to find out which one best suit the DL model is shown in Table 3.

Table 2.1

Performance metrics for all the varied Optimizers

Case	Accuracy scores	Precision	Recall	F1 Score	ROC-AUC values	Classification Report	Confusion Matrix	Best Parameters	Best Cross Validation Score	Loss
1	Train: 0.943 Validation: 0.947 Test:0.945	0.951	0.945	0.945	0.57	Precision Recall F1 Score Support EB 1 0.89 0.94 118 LB 0.89 1 0.94 116 Healthy 1 0.95 0.98 22 Accuracy 0.95 256 Macro avg 0.96 0.95 0.95 256 Weighted avg. 0.96 0.95 0.95 256	[[105 13 0] [0 116 0] [0 1 21]]	'C': 0.0316 'solver': 'newton-cg'	0.771	Train: 0.144 Validation: 0.13 Test:0.163
2	Train: 0.480 Validation: 0.506 Test:0.488	0.578	0.488	0.365	0.40	Precision Recall F1 Score Support EB 0.47 1.00 0.64 113 LB 0.80 0.10 0.18 119 Healthy 0.00 0.00 0.00 24 Accuracy 0.49 256 Macro avg 0.42 0.37 0.27 256 Wt. avg 0.58 0.49 0.37 256	[[113 0 0] [107 12 0] [21 3 0]]	'C': 0.0316 'solver': 'newton-cg'	0.781	Train: 0.94 Validation: 0.92 Test:0.915
3	Train: 0.921 Validation: 0.902 Test: 0.996	0.996	0.996	0.996	0.47	Precision Recall F1 Score Support EB 1.00 0.99 1.00 127 LB 0.99 1.00 1.00 117 Healthy 1.00 1.00 1.00 12 Accuracy 1.00 256 Macro avg. 1.00 1.00 1.00 256 Wt. avg. 1.00 1.00 1.00 256	[[126 1 0] [0 117 0] [0 0 12]]	'C': 0.001, 'solver': 'newton-cg'}	0.78	Train: 0.195 Validation: 0.234 Test: 0.030
4	Train: 0.467 Validation: 0.514 Test: 0.515	0.607	0.515	0.406	0.5	Precision Recall F1 Score Support EB 0.49 1.00 0.66 116 LB 0.80 0.13 0.22 123 Healthy 0.00 0.00 0.00 17 Accuracy 0.52 256 Macro avg. 0.43 0.38 0.29 256 Wt. avg. 0.61 0.52 0.41 256	[[116 0 0] [107 16 0] [13 4 0]]	'C':0.001, 'solver': 'newton-cg'	0.776	Train: 0.896 Validation: 0.9 Test: 0.869
5	Train: 0.936 Validation: 0.934 Test: 0.941	0.983.	0.980	0.980	0.41	Precision Recall F1 Score Support EB 0.99 1.00 1.00 129 LB 1.00 0.95 0.98 107 Healthy 0.83 1.00 0.91 20 Accuracy 0.98 256 Macro avg 0.94 0.98 0.96 256 Wt. avg 0.98 0.98 0.98 256	[[129 0 0] [1 102 4] [0 0 20]]	'C':31.622, 'solver': 'newton-cg'	0.773	Train: 0.154 Validation: 0.168 Test: 0.058

Table 3

Cases with all different Learning rates for Adam Optimizer

Case	**1.1**	**1.2**
Optimizer	Adam	Adam
Learning Rate	0.0001	0.001
Epochs	50	50
Batch_size	32	32
Filter_size	3*3	3*3
Activation function	ReLU	ReLU
Channels	3	3

Table 3.1

Performance metrices for all the varied Learning rates for Adam Optimizer

<table>
<tr><th>Case</th><th>Accuracy scores</th><th>Precision</th><th>Recall</th><th>F1 Score</th><th>ROC-AUC values</th><th>Classification Report</th><th>Confusion Matrix</th><th>Best Parameter</th><th>Best Cross Validation Score</th><th>Loss</th></tr>
<tr><td>1.1</td><td>Train: 0.943
Validation: 0.947
Test:0.945</td><td>0.951</td><td>0.945</td><td>0.945</td><td>0.57</td><td><table><tr><th></th><th>Precision</th><th>Recall</th><th>F1 Score</th><th>Support</th></tr><tr><td>EB</td><td>1.00</td><td>0.89</td><td>0.94</td><td>118</td></tr><tr><td>LB</td><td>0.89</td><td>1.00</td><td>0.94</td><td>116</td></tr><tr><td>Healthy</td><td>1.00</td><td>0.95</td><td>0.98</td><td>22</td></tr><tr><td>Accuracy</td><td></td><td></td><td>0.95</td><td>256</td></tr><tr><td>Macro avg</td><td>0.96</td><td>0.95</td><td>0.95</td><td>256</td></tr><tr><td>Wt. avg</td><td>0.95</td><td>0.95</td><td>0.95</td><td>256</td></tr></table></td><td>[[105 13 0]
[0 116 0]</td><td>'C': 0.0316
'solver':
'newton-cg'</td><td>0.771</td><td>Train: 0.144
Validation: 0.136
Test:0.163</td></tr>
<tr><td>1.2</td><td>Train: 0.957
Validation: 0.971
Test:0.992</td><td>0.992</td><td>0.992</td><td>0.992</td><td>0.49</td><td><table><tr><th></th><th>Precision</th><th>Recall</th><th>F1 Score</th><th>Support</th></tr><tr><td>EB</td><td>0.99</td><td>0.99</td><td>0.99</td><td>129</td></tr><tr><td>LB</td><td>0.99</td><td>0.99</td><td>0.99</td><td>106</td></tr><tr><td>Healthy</td><td>1.00</td><td>1.00</td><td>1.00</td><td>21</td></tr><tr><td>Accuracy</td><td></td><td></td><td>0.99</td><td>256</td></tr><tr><td>Macro avg.</td><td>0.99</td><td>0.99</td><td>0.99</td><td>256</td></tr><tr><td>Wt. avg.</td><td>0.99</td><td>0.99</td><td>0.99</td><td>256</td></tr></table></td><td>[[128 1 0]
[1 105 0]</td><td>'C': 1.0,
'solver':
'newton-cg'</td><td>0.773</td><td>Train: 0.108
Validation: 0.083
Test: 0.020</td></tr>
</table>

From Table 3.1, It was observed that on increasing the learning rate the AUC value did not improve in case 1.2 and hence the classification performance didn't improve. AUC value being higher in Case 1.1 means better classification achieved and lower value of C i.e. Inverse regularization indicates stronger regularization, which can hinder the model's capability to capture complex parameters but reduces overfitting. It can now be experimented with other hyperparameters like the count of epochs. So, taking Case 1.1 forward, also experimenting with the count of Epochs as shown in Table 4.

Table 4

Cases with all different Epochs for learning rate 0.0001 Adam Optimizer

Case	1.1.1	1.1.2	1.1.3
Optimizer	Adam	Adam	Adam
Learning Rate	0.0001	0.0001	0.0001
Epochs	50	100	150
Batch_size	32	32	32
Filter_size	3*3	3*3	3*3
Activation function	ReLU	ReLU	ReLU
Channels	3	3	3

In the above Table 4.1, Case 1.1.2 Test accuracy increased significantly, precision, recall, f1 score values are good, auc score is highest of all which means classification of instances is better. Value of C being high means even though it is susceptible to undergo certain amount of Overfitting of data but its' ability to identify complex features is more. Also, Test loss has significantly reduced which means that it is working fine with any new data under classification.

7. Results

After thorough comparison of all the hyperparameter tuned models which have been represented above as cases in Table 2.1, 3.1 and 4.1 respectively. The most optimal model i.e. the best case comprised of the optimizer adam, with count of epochs equivalent to 100, batch size:32, learning rate:which is 0.0001, filter size: 3*3 and activation function: ReLu on the Convolution layer and activation function on the fully connected layer is Softmax. The model's accuracies achieved comprised of train accuracy, validation accuracy and test accuracy as 0.959, 0.960 and 0.988 respectively. The losses for the model include training loss, validation loss and test loss came out to be 0.102, 0.100 and 0.039 respectively. After comparing all the model cases recall was 0.988, precision was 0.988 and F1 score was 0.988, roc-auc value was 0.563. The novelty being added here is the use of GridSearchCV in finding the robustness of the base CNN model. It adds to the other performance parameters such as a high cross-validation score of 0.78 for generalization, regularization parameter depicted by high value of C i.e. 31.622 and solver: newton-cg was used for Optimisation.

Table 4.1

Performance metrices for all the varied Epochs and Learning rates fixed as 0.0001 for Adam Optimizer

Case	Accuracy scores	Precision	Recall	F1 Score	ROC-AUC values	Classification Report	Confusion Matrix	Best Parameter	Best Cross-Validation Score	Loss
1.1.1	Train: 0.930 Validation: 0.915 Test:0.949	0.952	0.949	0.949	0.504	Precision Recall F1 Score Support EB 1.00 0.91 0.95 115 LB 0.91 0.99 0.95 120 Healthy 0.95 0.90 0.93 21 Accuracy 0.95 256 Macro avg 0.95 0.94 0.94 256 Wt. avg 0.95 0.95 0.95 256	[[105 10 0] [0 119 1] [0 2 19]]	'C': 0.001, 'solver': 'newton-cg'	0.776	Train: 0.176 Validation: 0.205 Test:0.126
1.1.2	Train: 0.959 Validation: 0.960 Test: 0.988	0.988	0.988	0.988	0.563	Precision Recall F1 Score Support EB 1.00 0.97 0.99 108 LB 0.98 1.00 0.99 126 Healthy 1.00 1.00 1.00 22 Accuracy 0.99 256 Macro avg 0.99 0.99 0.99 256 Wt. avg 0.99 0.99 0.99 256	[[105 3 0] [0 126 0] [0 0 22]]	'C': 31.62277660168 3796, 'solver': 'newton-cg'	0.780	Train: 0.102 Validation: 0.100 Test: 0.039
1.1.3	Train: 0.972 Validation: 0.978 Test: 0.996	0.996	0.996	0.996	0.484	Precision Recall F1 Score Support EB 1.00 1.00 1.00 121 LB 1.00 0.99 1.00 117 Healthy 0.95 1.00 0.97 18 Accuracy 1.00 256 Macro avg. 0.98 1.00 0.99 256 Wt. avg. 1.00 1.00 1.00 256	[[121 0 0] [0 116 1] [0 0 18]]	'C': 1.0, 'solver': 'newton-cg'	0.781	Train: 0.070 Validation: 0.056 Test: 0.033

8. Conclusion & Future Scope

Deep Convolution Neural Network has been optimized in this paper by performing hyperparameter tuning of the base model. Hyperparameters like learning rate, epochs, keeping batch size, filter size, activation function and the number of channels fixed for all cases were taken into consideration. Various performance metrices like f1 score, accuracy, recall, support, roc-auc, precision etc. were noted. The novelty added in this paper is the use of GridSearchCV in identifying the robustness of the base CNN model. It provides additional performance metrics such as high cross-validation score: 0.78 indicates that the model works good on unknown data**,** regularization as shown by high value of C i.e. 31.622, is more tolerant of misclassifications which leads to higher accuracy overall but also is lesser sensitive to detecting positive cases. Also, for optimization of parameters 'newton-cg' solver was used, resulting in robust performance. In future, hyperparameter tuning of the model **c**an be performed for optimization of the batch_size, the number of channels and filter_size. Also, various activation functions can be utilized for enhancing the efficiency of the model. The deployed model in this paper could also later be used for some IOT generated datasets or camera clicked images from the agricultural fields.

9. Abbreviations

PV dataset, Plant Village dataset; EB, Early Blight; LB, Late Blight; PH, Potato Healthy; DL model, Deep Learning model; CNN base model, Convolution Neural Network base model.

10. Acknowledgement

Really grateful to the Department of Biotechnology (DBT) for kind support in undertaking this research under project titled "Application of Machine Learning for Hyperspectral Imaging and Remote Sensing aimed at Early Detection of Fungal Foliar Diseases and Bacterial Wilt Diseases in Potato Crop", DBT/Reference. No. BT/PR45388/133/58/2022.

References

[1] N. Sareen, Anuradha Chug, and Amit Prakash Singh, "An image based prediction system for early blight disease in tomato plants

using deep learning algorithm," *Journal of Information and Optimization Sciences*, vol. 43, no. 4, pp. 761-779 (2022).

[2] M. Al-Adhaileh, A. Verma, T. Aldhyani, and D. Ko, "Potato Blight Detection Using Fine-Tuned CNN Architecture," *Mathematics*, vol. 11, no. 6, pp. 1-16 (2023), doi: 10.3390/math11061516.

[3] J. Chen, X. Deng, Y. Wen, Z. Liu, and A. Zhang, "Weakly-supervised learning method for the recognition of potato leaf diseases," *Artificial Intelligence Review*, vol. 56, pp. 7985–8002 (2023).

[4] S. Hassan and A. Maji, "Plant Disease Identification Using a Novel Convolutional Neural Network," *IEEE Access*, pp. 1-13 (2022), doi: 10.1109/ACCESS.2022.3141371.

[5] R. Sharma, A. Singh, K. Kavita, N. Jhanjhi, M. Masud, E. Jaha, and S. Sk, "Plant Disease Diagnosis and Image Classification Using Deep Learning," *Computers, Materials and Continua*, vol. 71, pp. 2125-2140 (2022), doi: 10.32604/cmc.2022.020017.

[6] D. Tiwari, M. Ashish, N. Gangwar, A. Sharma, and P. Kumar, "Potato leaf diseases detection using deep learning," in Proc. 2020 4th Int. Conf. Intelligent Computing and Control Systems (ICICCS), Madurai, India, pp. 461–466 (2020), doi: 10.1109/ICICCS48265.2020.9121067.

[7] N. Sinshaw and B. Assefa, "Artificial Intelligence Assisted Early Blight and Late Blight Potato Disease Detection Using Convolutional Neural Networks," *Ethiop. J. Crop Sci.*, vol. 8, no. 2, pp. 15-28 (2020).

[8] G. Polder, P. M. Blok, H. A. C. de Villiers, J. M. van der Wolf, and J. Kamp, "Potato Virus Y Detection in Seed Potatoes Using Deep Learning on Hyperspectral Images," Front Plant Sci., vol. 10, pp 1-13 (2019), doi: 10.3389/fpls.2019.00209, PMID: 30881366, PMCID: PMC6405642.

[9] P. Mathur, F. Sheth, D. Goyal, and A. Gupta, "Deep insight: Mathematical modeling and statistical analysis for mango leaf disease classification using advanced deep learning models," *Journal of Interdisciplinary Mathematics,* vol. 27, pp. 317-342 (2024), doi: 10.47974/JIM-1830.

[10] D. Bhoyar, S. Jain, J. D. Kene, Y. A. Suryawanshi, A. R. Bhagat, and S. J. Karale, "Enhancing agriculture productivity with machine learning in crop yield optimization," *Journal of Statistics and Management Systems,* vol. 27, no. 2, pp. 213–223 (2024), doi: 10.47974/JSMS-1248.

Received August, 2024