International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

# An Optimized Technique for Plant Identification through Deep Residual Networks

**Jaswant Narendra Saxena[1], Ananya Nagraj[2]**

[1]Senior Consultant, Cognizant Technology Solutions, Hyderabad
[2]Post Graduate Student, Stevens Institute of Technology, Master of Science (MS) in Management Information Systems

**Abstract**
Advancing our knowledge and understanding of the plants around us is very significant and crucial in medical, economic, and sustainable agriculture. Plant image recognition has been an interdisciplinary emphasis in the science of computer vision. Convolutional neural networks (CNN) are used to learn feature representation of 185 classes of leaves, under the benign conditions of rapid advancement in computer vision and deep learning algorithms. A 50-layer deep residual learning framework with 5 steps is built for large-scale plant classification in the natural environment. On the leaf snap data set, the proposed model achieves a recognition rate of 93.09 percent as accuracy of testing, demonstrating that deep learning is a highly promising forestry technology.

**Keywords:** Plant Identification, Deep Learning, Residual Networks.

## 1. Introduction

Plants are a crucial resource for human well-being since they provide oxygen and sustenance. As a result, experts and the breeding business are working hard to ensure that agriculture can continue for a long time without interruption. A strong understanding of plants is required to completely recognize new, distinct, or uncommon plant species in order to support the ecosystem, boost the drug business, and increase agricultural sustainability and productivity. As deep learning technology progresses, various new and advanced models for automatic plant identification have been presented. Furthermore, in computer vision, the classification and learning of an item in an image is a very difficult operation. Today, most researchers employ diverse leaf variations as a similar technique for studying novel plants, and some leaf databases, such as Flavia, Swedish, and ICL, have been transmitted, but fewer attempts have been made to extract local aspects of leaf, flower, or fruit. However, great effort has been directed towards identification and prediction in various applications. Furthermore, computational intelligence approaches are critical in identification and prediction applications. Rough computing, for example, is combined with neural networks [1, 2], genetic algorithms [3, 4], and soft sets [5].

Deep convolutional neural networks are essential for learning various visual features utilizing image classification techniques. To improve image quality, we must increase resolution, which necessitates the development of deeper neural networks, and increasing the number of stacked layers of the network becomes critical, as evidenced by recent data [6,7]. The enormous number of hidden layers creates the problem of vanishing gradient descent [8], which classical machine learning algorithms cannot overcome.

To address the aforementioned issues and to capitalize on the deep learning breakthrough in image recognition, a 50-layer deep learning model based on residual networks is constructed for uncontrolled plant identification on the Leafsnap dataset, which contains 185 different tree species. The proposed model obtains an accuracy rate of 93.09 percent with a margin of error of 0.24 percent.

## 2. Literature Survey

Plant detection is crucial in assisting various or uncommon plant species in increasing drug trafficking, maintaining the ecosystem, and increasing agricultural output and property. Neeraj Kumar et al. [9] proposed Leafsnap, an image recognition system for plant species that may be detected automatically. They created a mobile app that helps users identify trees by taking images of their leaves, and their current version includes all species in the specified dataset. They employed an essential metric to determine the many functions of the curvature-based shape of the border leaf and recognized plants based on these characteristics. Their recognition method is divided into four stages: categorizing, segmenting, extracting, and comparing. Furthermore, they used the closest neighbors (NN) technique to identify leaf type.

Sue Han et al. [10] used a well-trained convolutional neural network model to identify plants. Instead of utilizing CNN, they recommended employing deconvolution networks (DN) to recognize the learned features. This method was used to get visual awareness of the features required to recognize a leaf from distinct classes, eliminating the requirement to develop the features manually. They created a new dataset named MalayaKew Leaf Dataset, which contained only 44 classifications. They constructed a new dataset (D2) by manually cropping and rotating the photos in the existing dataset (D1). They randomly chose 34672 leaf patches for training and 8800 for testing, yielding 99.6% accuracy on the D2 dataset and 97.7% accuracy on the D1 dataset.

Jing Hu et al. [11 and 12] suggested an MSF-CNN (Multi-Scale Fusion Convolutional Neural Network) for leaf detection at different plant scales. They used a set of bilinear interpolation techniques to down sample one input image into numerous low-resolution images. The photos were then input into the MSF-CNN architecture, which gradually learned distinct features in multiple layers. By pooling the last layer information, the final characteristic for anticipating the input image plant species is obtained.

They retrained the Deep Plant on the D1 dataset and predicted classes with an accuracy of 98.1% using the Support Vector Machine (SVM) model and 97.7% using the Multi-Layer Perceptron (MLP) technique. They discovered that some of the classes were incorrectly categorized and concluded that detecting plant shapes is not a good way to recognize plants. The model was then trained on the D2 dataset, and it achieved 99.6%, which is greater than the D1 dataset. They determined that the D2 outperforms the D1 because venation of separate orders is a more robust feature for plant recognition.

Jaswant Narendra Saxena et al. [13,14 and 15] improved a trained model to recognize leaves in pictures. They demonstrated how a model trained on a large dataset may be applied on a small training dataset. As a result, established machine learning methods were surpassed by using local binary patterns (LBPs). They did not train their model from scratch, instead using an ImageNet-trained CNN model. They worked on an ImageClef2013 dataset that has photos of both clean and cluttered backgrounds. Because to the scarcity of training data, there was overfitting and significant variability. As a result, they used transfer learning to avoid overfitting and fine-tuned AlexNet using the Caffe framework. They compared AlexNet from scratch with random initialization to fine-tuning versions, yielding an accuracy of 71.17% on the validation dataset and 70.0% on the testing dataset.

## 3. Proposed Research Design

A 50-layer deep residual network is used in the suggested study design. Each layer is made up of an identity block and a convolutional block. The identity block is the standard ResNets block, and it corresponds to the condition where the input and output activations have the same dimensions. We ran a more powerful version (shown in Fig. 1) that skips three hidden layers rather than two, giving our model a distinct advantage in terms of speed and accuracy.

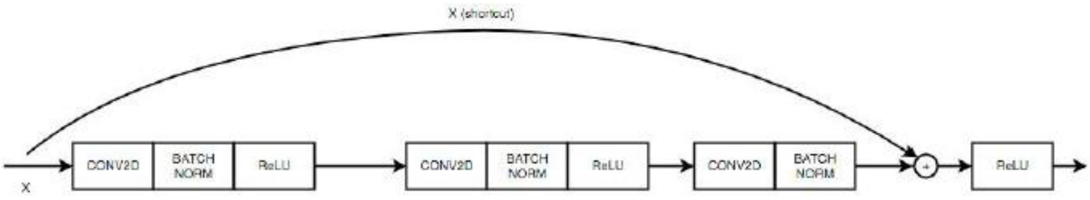



**Figure 1. The identity block**

This block's function is to match input and output dimensions. The difference between this block and the identity block is that there is an additional CONV2D layer in the shortcut path, as illustrated in Fig.2. We utilized the CONV2D layer in the shortcut path to stretch the input to the different dimension so that the dimensions match up in the final addition required to add the shortcut value back to the main path.

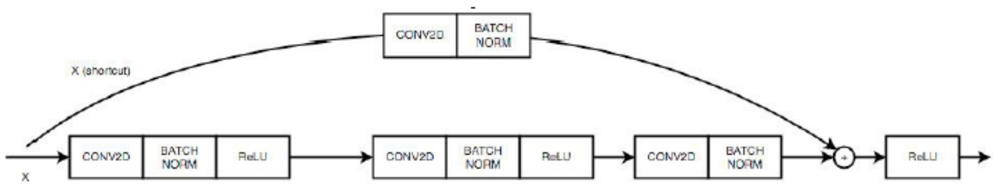



**Figure 2. The convolutional block.**

### 3.1 Implementation

Our model's implementation is based on the open-source deep learning framework keras. All trials were carried out on Google's Cloud platform, which used a Tesla K80 GPU processor. We used the Augmentor library, which is a standalone Python package that allows finer grain control over augmentation and implements the most relevant augmentation techniques in the actual world. It employs a stochastic technique that use building blocks to connect processes in a pipeline. To improve accuracy, the data set size [27] has been extended by rotating, zooming, and rotating the photos. All of the photographs were downsized to 64x64x3 dimensions before being split by 255.

**Algorithm 1: Augmentation of images**
**initialization**:
**N**: no of images.
**width**: width of images we want to change.
**height**: height of images we want to change.
**sample**: after mentioning specific changes, we can sample it which will generate N augmented images based on our specifications.
**path**: path of the dataset where all images are present
**begin**
Initialize an empty pipeline
image←pipeline.Augmentor(path)
image←pipeline.rotate(max_left_rotation, probability, max_right_rotation)
image←pipeline.flip_left_right(probability)
image←pipeline.zoom_random(percentage_area=0.8, probability)
image← pipeline.flip_top_bottom(probability)
image←pipeline.resize(probability, width, height)
sample←pipeline.sample(N←27000)

### 3.2 Dataset distribution and converting into HDF5 format

Our dataset (which was made up of photos) was converted into Hierarchical Data Format 5 (hdf5), which is a collection of file formats designed to store and organize massive amounts of data. We transformed our dataset to hdf5 format because of its hierarchical structure (similar to files/folders), compression rate, and speed of access. Following that, we prepared 150 photographs for each class (out of a total of 180 classes), for a total of 27,000 images. We then divided our dataset into 80% training (21600 images) and 20% testing (5400 photos).

**Algorithm 2: Actual implementation of Residual networks**
**initialization:**
**filters:** the number of filters in the CONV layers of the main path
**image**: input image
**input_shape:** images' shape
**classes**: number of classes, integer
**epochs**: no of epochs
**batch_size**: number of training examples utilised in one iteration
**F**: no of filters
**begin:**
training_dataset, test_dataset← load dataset
training_dataset← normalize training dataset by dividing pixels values with 255.
testing_dataset← convert test dataset using one hot encoding
model← call ResNet50 function with input_shape← 64x64x3 and classes ← 180
model← compile model using 'adam' optimizer and 'categorical_crossentropy' loss value.
fit model with training_dataset, testing_dataset, epochs and batch size as parameters

```
Function ResNet50(input_shape, classes)
    Stage 1:
    image←zero_padding(padding_shape)
    image← 2D_Convolution(F,filters)
    image←Relu_Activation(image)
    image←max_pooling(window_shape)
    Stage 2:
    filter←64x64x256
    stage←2
    image← call convolutional_block function with image, filter, stage and block←'a'asparamenters.
    image← call convolutional_block function with image, filter, stage and block←'b'asparamenters
    image← call convolutional_block function with image, filter, stage and block←'c'asparamenters
    Stage 3:
    filter←128x128x512
    image← call convolutional_block function with image, filter, stage and block←'a'asparamenters
    image← call identity_block function with image, filter, stage and block←'b'asparamenters
    image← call identity _block function with image, filter, stage and block←'c'asparamenters
    image← call identity _block function with image, filter, stage and block←'d'asparamenters
    Stage 4:
    filter←256x256x1024
    stage←4
    image← call convolutional_block function with image, filters, stage and block←'a'asparamenters
    image← call identity_block function with image, filters, stage and block←'b'asparamenters
    image← call identity _block function with image, filters, stage and block←'c'asparamenters
    image← call identity _block function with image, filter, stage and block←'d'asparamenters
    image← call identity _block function with image, filter, stage and block←'e'asparamenters
    image← call identity _block function with image, filters, stage and block←'f'asparamenters

        Stage 5:
        filter← 512x512x20148
        image← call convolutional_block function with image, filters, stage←5 and block←'a'asparamenters
        image← call identity_block function with image, filters, stage←5 and block←'b'asparamenters
        image← call identity _block function with image, filters, stage←5 and block←'c'asparamenters
        image←average_pooling(pool_size, padding_shape)
        image← convert output into categorical values using softmax activation function
        model← compile
    Function identity_block(image, filters)
        prev_image← image
        image← 2D_Convolution(filters)
        image←BatchNormalization(image)
        image←Relu_Activation(image)
        image← Add(prev_image, image)
        image←Relu_Activation(image)
        return image
    Function convolutional_block(image, filters)
        prev_image← image
        image← 2D_Convolution(filters)
        image←BatchNormalization(image)
        image←Relu_Activation(image)
        prev_image← 2D_Convolution(filters)
        prev_image←BatchNormalization(prev_image)
        image← Add(prev_image, image)
        image←Relu_Activation(image)
```

## 4. Result Analysis

In this part, we evaluate the performance of our CNN model using residual networks. We utilized the "Adam" optimizer and the "categorical_crossentropy" loss function since they are suitable for predicting multiple mutually incompatible classes. Table 1 shows the configuration parameters of our model.

Table 1. Configuration of the parameters used during training.

| Parameter | Value |
|---|---|
| Image size | 64x64x3 |
| Optimizer | Adam |
| Learning rate | 0.001 |
| Batch Size | 120 |
| Epochs | 41 |

The training procedure of our ResNet50 model (50 layers) is depicted in Fig.3(a). Training accuracy improves quickly after the first epoch and stabilizes after 32 epochs. As demonstrated in Fig. 3 (b), our model achieved 99.07% training accuracy and the error rate decreased from 1.3133% to 0.0344%. Table 2 shows that the suggested ResNet50 achieves 93.09% testing accuracy with a loss rate of 0.247%.

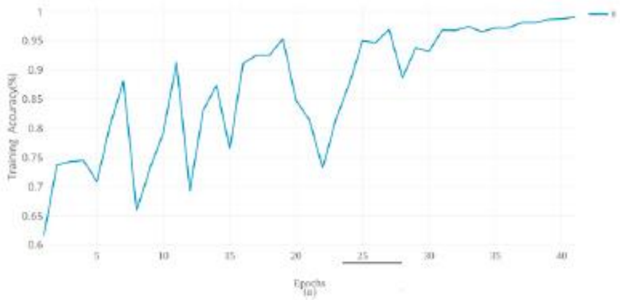
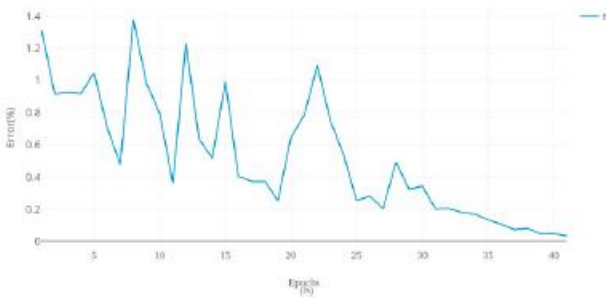



**Figure 3. (a) Evolution of classification accuracy on training set.**




**(b) Training Error Rate**

Table 2. Performance evaluation of our model

| Process | Number of images | Accuracy (%) | Error (%) |
|---|---|---|---|
| Training | 21600 | 99.07 | 0.034 |
| Testing | 5400 | 93.09 | 0.247 |

## 5. Conclusion

This paper investigated a deep learning methodology that used CNN to learn discriminatory traits for plant identification from leaf photos. We demonstrated how skip-connections in residual networks aid in addressing the Vanishing Gradient problem, achieving 93.09% accuracy in the test set, demonstrating that deep learning is a potential method for large-scale plant categorization in the natural world. In the future, we intend to investigate several CNN architectures in order to improve performance. We intend to extend the deep learning model from the classification challenge to include prediction, disease segmentation, insect detection, and other tasks.

## References

1. Anitha and D. P. Acharjya. (2015) "Neural Network and Rough Set Hybrid Scheme for Prediction of Missing Associations." International Journal of Bioinformatics Research and Applications, Vol. 11 (6), pp. 503-524.
2. A. Anitha and D. P. Acharjya, (2018) "Crop Suitability Prediction in Vellore District using Rough Set on Fuzzy Approximation Space and Neural Network." Neural Computing & Applications, Vol. 30 (12), pp. 3633–3650, DOI: 10.1007/s00521-017-2948-1.
3. R. Rathi and D. P. Acharjya, (2018) "A Rule Based Classification for Vegetable Production for Tiruvannamalai District using Rough Set and Genetic Algorithm." International Journal of Fuzzy System Applications, Vol. 7 (1), pp. 74 – 100.
4. R. Rathi and D. P. Acharjya, (2018) "A Framework for Prediction using Rough Set and Real Coded Genetic Algorithm." Arabian Journal for Science and Engineering, Vol. 43 (8), pp. 4215 – 4227.
5. T. K. Das and D. P. Acharjya, (2014) "A Decision-Making Model using Soft Set and Rough Set on Fuzzy Approximation Spaces." International Journal of Intelligent Systems Technologies and Applications, Vol. 13 (3), pp. 170-186.
6. K. Simonyan and A. Zisserman, (2015) "Very deep convolutional networks for large-scale image recognition." International Conference on Learning Representations, arXiv:1409.1556.
7. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, (2015) "Going deeper with convolutions." Computer Vision and Pattern Recognition, pp. 1–9, DOI: 10.1109/CVPR.2015.7298594.
8. X. Glorot and Y. Bengio. (2010) "Understanding the difficulty of training deep feedforward neural networks." Artificial Intelligence and Statistics, Vol. 9, pp. 249-256.
9. Neeraj Kumar, Peter N Belhumeur, Arijit Biswas, David W Jacobs, W John Kress, Ida C Lopez, and João VB Soares, (2012) "Leafsnap: A computer vision system for automatic plant species identification." European Conference on Computer Vision, Vol. 7573, pp. 502–516, Springer.

10. S. H. Lee, C. S. Chan, P. Wilkin, and P. Remagnino, (2015) “Deep- plant: Plant identification with convolutional neural networks.” International Conference on Image Processing (IEEE), pp. 452–456, DOI: 10.1109/ICIP.2015.7350839.
11. Jing Hu, Zhibo Chen∗, Meng Yang, Rongguo Zhang and Yaji Cui, (2018) “A Multi-Scale Fusion Convolutional Neural Network for Plant Leaf Recognition.” IEEE Signal Processing Letters; vol. 25, pp. 853 – 857.
12. Mohamed Abbas Hedjazi, IkramKourbane, YakupGenc, (2017) “On Identifying Leaves: A Comparison of CNN with Classical ML Methods.” Signal Processing and Communications Applications Conference (SIU), IEEE, pp. 1-4, DOI: 10.1109/SIU.2017.7960257.
13. Jaswant Narendra Saxena and Ananya Nagraj, “An Efficient Real-Time Object Recognition Model Using Deep Learning Approach”, International Journal of Multidisciplinary Educational Research ISSN:2277-7881; Volume:12, Issue:7(1), July- 2023.
DOI: http://ijmer.in.doi./2023/12.07.08
14. Jaswant Narendra Saxena and Ananya Nagraj, “Hybrid System for Detection and Classification of Plant Disease Using Qualitative Texture Features Analysis”, International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211, Volume 11, Issue 7, July-2023.
DOI: https://doi.org/10.56025/IJARESM.2023.117232299
15. Jaswant Narendra Saxena and Ananya Nagraj, “An Optimized Technique for Image Classification Using Deep Learning”, IRJCS: International Research Journal of Computer Science ISSN: 2393-9842 Volume 10, Issue 04, May-2023.
DOI: https://doi.org/10.26562/irjcs.2023.v1004.11