Flower Species Detection System Using Deep Convolutional Neural Networks

Arun Solanki and Tarana Singh

Abstract A system that correctly identifies the name of a flower species may be beneficial for botanists, camping enthusiasts, and researchers. Previously, classification was only done based on a flower's shape, geometry, and texture, which is not enough for an efficient system. Some significant challenges in this classification task include inter-species similarity, intra-class variation, and the same objects such as leaves or grass around a flower, making this task a research topic. This research has developed an efficient and robust deep learning flower classifier to overcome these problems and limitations based on the current state of the art convolutional neural networks and transfer learning. This research has utilized the Oxford-102 flower dataset having 8189 images of 102 flower species. The proposed method is divided into two different steps. Firstly, the flower images are segmented, and secondly, these segmented images are fed as an input to a convolutional neural network for classifying the species of the flowers. This work has used the PyTorch library for recognition purposes. The flower's dataset uses various pre-trained models on the ImageNet dataset such as AlexNet, VGG, DenseNet, Inception v3, and GoogLeNet. Out of these, DenseNet achieved the highest classification accuracy of 97.92% when trained on GPU provided by Google Collaboratory. This classifier can be integrated with a mobile application to provide an accurate real-time flower species prediction.

Keywords Convolution neural network (CNN) · Segmentation · Cropping · Augmentation · Transfer learning · ReLU activation function

1 Introduction

Plant species recognition based on flower recognition remains a challenge in the field of image processing and computer vision, primarily due to their widespread presence, complex structures, and unpredictable species in nature. Due to this natural complexity, it is highly undesirable to segment or extract regular features or combine

A. Solanki (✉) · T. Singh
Department of Computer Science and Engineering, Gautam Buddha University, Greater Noida, U.P., India

P. K. Singh et al. (eds.), *Emerging Technologies for Computing, Communication and Smart Cities*, Lecture Notes in Electrical Engineering 875,
https://doi.org/10.1007/978-981-19-0284-0_17

shape, texture, and color features, resulting in moderate accuracy in benchmark datasets increase. Several feature extraction techniques that combine global and local feature descriptors have achieved the highest accuracy in flower classification, but automatically identify and recognize large flower species in complex environments [1]. Still needs a powerful and efficient system. This paper also uses mechanism of transfer learning to save our time and resources. For this, we have utilized Oxford-102 dataset of images having 8189 flower images belonging to 102 kinds of different flower species. The proposed method is divided into two major steps [2]. Firstly, the flower images are segmented, and secondly, these segmented images which act as input afterward go into a convolutional neural network for classifying the species belonging to different flower categories. We have also pre-processed the flower images which we will discuss in the later section of this paper. To facilitate our proposed work, we have employed PyTorch library for the recognition purposes and various pre-trained models on ImageNet dataset such as AlexNet, VGG, DenseNet, Inception v3, and GoogLeNet were used on the flower's dataset. Out of these, DenseNet achieved the highest classification accuracy when trained on GPU provided by Google Collaboratory [3]. This classifier can be built into a mobile application so that it can provide an accurate flower species prediction in real-time environment. For the implementation of the flower species recognition system, we have used python 3.6.6 version. We have employed PyTorch library for the development of the code based on transfer learning. The whole training of the model is done on Google Collaboratory which is free GPU provided by Google. We have obtained the dataset which is used in our proposed methodology is from Science and Engineering department of University of Oxford. This dataset is known as 102 flower category datasets [4]. This flower's dataset is having 102 types of different flower species. Each category of flower contains the images ranging from 40 to 258 images.

CNN comes under the picture of an artificial neural network that has wide recognition in image classification. Several layers in a CNN includes convolutional layers, pooling layers, and fully connected layers [5]. Operation in the series format is applied to the data, which acts as input to the CNN network to find a particular pattern in the image. This network does the processing of the data with a grid-like topology [6]. A CNN model uses the image pixel in the form of an array as input. The input data is processed through a hidden layer, and the final output is shown or given by the output layer. The primary function of the hidden layer is feature extraction that deals with the calculation and manipulation of the data [7]. This work is based on the convolution layer, which filters the matrix and performs the convolution operations to help pattern finding in the image. Hidden layers can be many depending upon the architecture of the network like we can have a ReLU layer or a pooling layer or convolution layer [8]. At the end of each process, we get feature maps convolved, rectified, and pooled. The modified pixel values have to get passed through the fully connected layer where the real object detection occurs [9]. Figure 1 illustrates the overall architecture of a convolutional neural network. This also has the depiction of all the layers contained in the network [10].

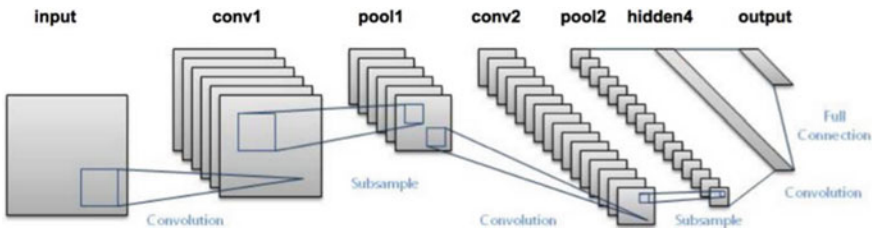


Fig. 1 Architecture of convolutional neural network [11]

When reviewing previous studies, several flower identification methods have been suggested [1, 6, 12]. These methods usually consist of four steps: pre-processing, segmentation, manual design feature extraction, and classification [13]. Due to the complex background of flower images, this task can be very time consuming, and for many types, in particular, the accuracy obtained is still low. Recently, learning feature representations using convolutional neural networks (CNNs) has been very successful in various areas of computer vision, including object detection, segmentation, and visual image classification [9]. Feature learning methods provide a natural way to capture clues using many codewords (sparse coding) or neurons (deep networks) [14, 15]. All of these are useful clues because you can capture the natural features of the object. Therefore, this article examines and presents the efficiency of deep convolutional neural networks, which may more effectively identify plant species based on flowers [16, 17].

This whole paper is organized into eight sections. In section one, introduction and the motivation of the work are presented. In the second section of this paper, a literature survey of the related domain is presented. In section three, the proposed architecture of the system is discussed. The process flow chart of the proposed system is discussed in section four. Then, in section five of this chapter, the pseudo-code of the proposed system is presented, followed by the step-by-step discussion of the implementation of the proposed algorithm in section six. Section seven discusses the results of the proposed system, followed by the comparison of the proposed work with the existing systems. At the end of the paper, a conclusion is given, followed by future work.

2 Literature Survey

Krizhevsky [18] brings out the phenomenal results on the ILSVRC2012 through developing a deep convolutional neural network. The top-1 error rate was 37.5% and the top-5 error rate was 17%. This method was certainly better than other methods in the past for the same domain. A system based on a convolutional neural network was a build-up of deep layers ensemble with the structural network containing eight layers. To avoid overfitting problem [19], there is the incorporation of the essential

features. These features are pooling layers along with normalizing layers with the functionality of dropout. According to Sermanet [20], using CNN for object location and object detection in images will boost classification accuracy. It will also increase the accuracy of detection and location tasks. This method is the winner of the localization task on the challenge of ILSVRC2013 through the developed integrated approach used for detection, localization, and recognition [21, 22]. This algorithm gave brilliant results through classification accuracy. Szegedy [10] developed and designed the architecture of a deep convolutional neural network which is called inception and there is seen great classification and detection results for the challenge ILSVRC2014 [23]. The author in [20] states that for the benefit of classification depth representation is essential. With the substantial increase in the intensity, good results can be achieved on the ImageNet dataset using a conventional CNN.

We can use a convolutional neural network for the segmentation of the images and can be employed to detect the objects in the images. Segmentation through CNN has been achieved through the paper's fully convolutional networks (FCN) concept [20]. Several methods extend the concept of CNN to allow object detection tasks with good accuracy on benchmark datasets. These methods are R-CNN [24] which is region proposals with CNN. Another advancement is fast R-CNN explained in [25]. Later on, there is the development of the architecture of Faster R-CNN [26] and YOLO [27]. The results are similar if we compare FCN with these methods when using CNN's architectures, including AlexNet [28] and VGG-16 [10].

3 The Architecture of the Proposed System

Figure 2 shows the framework, which is the designed architecture for our proposed method employed to deal with flower species recognition.

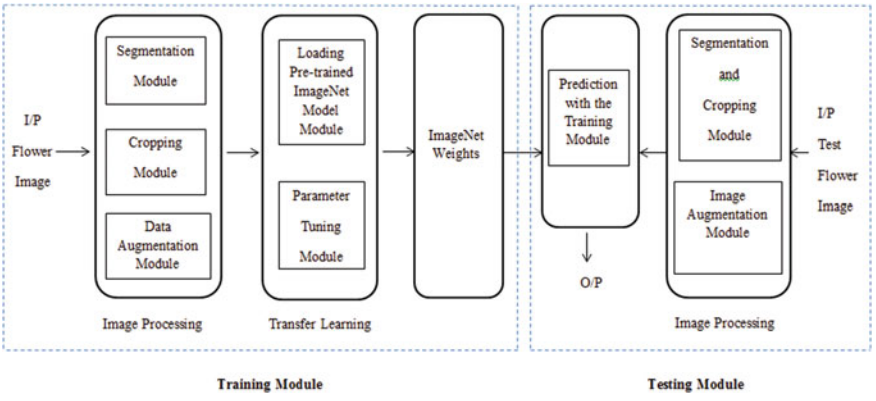


Fig. 2 Architecture of the proposed work

The efficient and robust system we have developed to classify different types of flower species is depicted in Fig. 2. This figure shows the overall framework, which is also the architecture of the proposed method. Architecture is composed of modules, blocks, and sub-modules [29]. Architecture describes the flow of the code right from data collection to the prediction of an unknown set of flower examples by the trained classifier. Here, we have utilized the architecture type of convolutional neural network, DenseNet, a pre-trained model on the ImageNet dataset [30]. This is called a transfer learning mechanism. There are two significant modules in the architecture are:

- **Training Module**: The training module of the architecture proposed contains three blocks and five modules which are described below. The input to this module is the raw flower images one by one. This module has three blocks: Image Processing Block, which is mainly responsible for the data preparation for the training and contains three modules: segmentation module, cropping module, and data augmentation module. The second is the Transfer Learning Block, which focuses on the transfer learning mechanism and comprises two major modules: loading pre-trained model module and parameter tuning module. And the third one is the **ImageNet Weight Block, which has the weights of the ImageNet dataset** used in our flower classification problem.
- **Testing Module**: The testing module of the architecture proposed contains two blocks and three modules which are described below. The output from this module is the class label, the predicted species of an unknown flower image, which is the input to this testing module. This module has three further modules: Predictions with the Training Module, Segmentation and Cropping Module, and Image Augmentation Module.

4 Process Flow Chart of the Proposed System

The process flow carried out in the proposed work is systematically explained by the below flowchart, which contains all the steps of execution to accomplish the research (Fig. 3).

4.1 Oxford 102 flower's Dataset

We have the oxford 102 flower's dataset at our disposal, which has to go into our classification model for flower species prediction [31]. Figures 4 and 5 are the depiction of variability between flower species and variability within flower types.

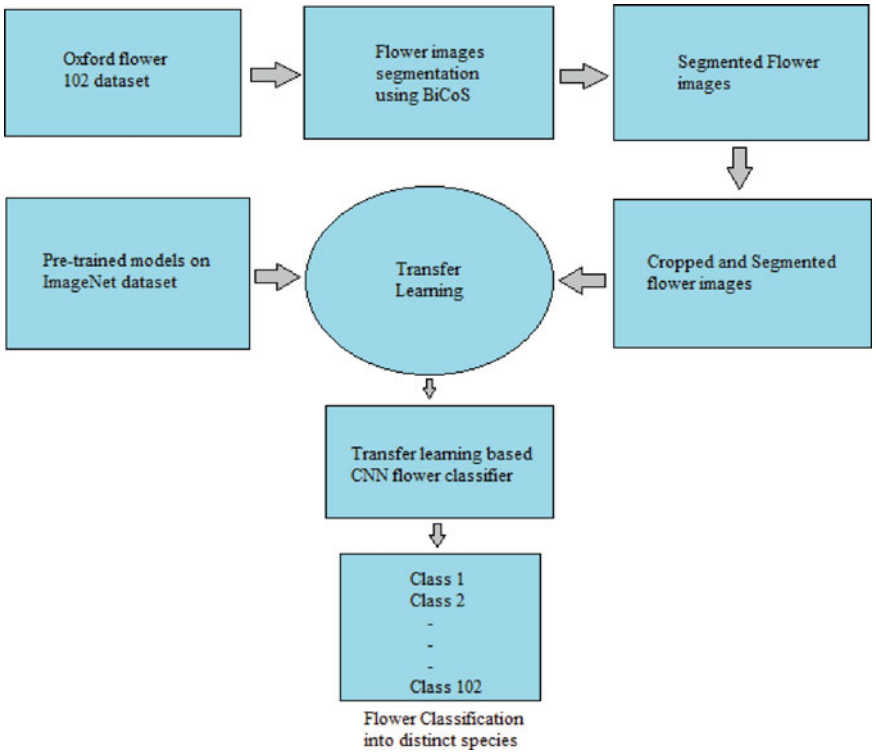


Fig. 3 Process flow of the proposed method

Fig. 4 Variability between classes

Fig. 5 Variability within classes

4.2 Flower Images Segmentation Using BiCoS

To improve classification accuracy, there is a need to separate foreground (the flower structure) from the complex background due to leaves and grass, which is useless for the classification task. So, we do the flower images segmentation to obtain the desired segmented images of the flowers.

4.3 Cropped and Segmented Flower Images

After segmentation, the background becomes black. We need to remove that, so we crop the segmented images using a python script to obtain the cropped and segmented images of flowers.

4.4 Pre-trained Models on ImageNet Dataset

There are many pre-trained models like DenseNet, VGG-16, or AlexNet that can be loaded from the Torchvision module of PyTorch.

4.5 Transfer Learning

This mechanism of transfer learning is gaining huge popularity in deep learning. We load the pre-trained models on the ImageNet dataset into our code. This mechanism is called transfer learning which is reusing things that have very high standards. Pre-trained networks generally contain two things. One is feature detectors, and the other is a classifier. Feature detectors extract from each image the information [32, 33]. Our classifier will learn the input given by feature layers, and therefore we will freeze the feature layers to avoid any modification. If we talk about the most commonly used

pre-trained models on the ImageNet dataset, these are AlexNet, DenseNet, VGG-16 and many more that have gained popularity over recent years.

5 Pseudo-Code of the Proposed System

Step 1: Loading Oxford 102 Dataset.
Step 2: Essential Libraries Loading.
Step 3: Data Segmentation and Cropping of Flower Images.
Step 4: Data Augmentation.
Step 5: Loading pre-trained model.
Step 6: Classifier Building.
Step 7: Model Training.
Step 8: Model Testing.
Step 9: Save Model Checkpoint.
Step 10: Load Model Checkpoint.
Step 11: Processing Images.
Step 12: Class Prediction.
Step 13: Sanity Check.

6 Implementation of the Algorithm

Step 1: Loading Oxford 102 Dataset—In this step, we load the Oxford 102 dataset of flowers images into our code to apply our model for the prediction of flower species. This dataset contains 102 species of flower images and is divided into training and test sets.

Step 2: Essential Libraries Loading—This process involves the loading of essential libraries and packages to make use of the functions in the modules of these packages.

Step 3: Data Segmentation and Cropping—We have to remove the complex background, which contains leaves and grass and these things create significant confusion for the flower classification task. Therefore, we have segmented the flower images using a technique called BiCoS segmentation for image classification. Then, this segmented image is cropped with the help of a python script to improve the accuracy of the network. Figure 6 shows the conversion of the original image of the flower picked up from the Oxford 102 species to the segmented image and the conversion from segmented image to cropped image.

Step 4: Data Augmentation—As our dataset is not very large, we need to augment the dataset of flower images. This is because we want our program to learn as much as it can. So, we must apply some random transformations to build a robust and efficient flower classifier. For this, we have to train our model on various variations of the

a) Original Image b) Segmented Image c) Cropped Image

Fig. 6 Cropped image obtained from the segmented image

original image. These variations include rotation, translation, scaling, cropping, or flipping of the original image.

The transformations are done on each image and these images are passed through a neural network in each epoch. This will allow the network to train on a more significant number of images. Therefore, we have increased the variety of our training data which will further reduce overfitting. This also improves the capability of generalization by the classifier, so there is a sure shot increase in the model's accuracy. To normalize the image values before entering them into the network, we have to provide the network with mean and standard deviation. If we look at the dimensions of image tensors, we can have the values of mean and standard deviation. PyTorch library allows us to do the data transformation through its torchvision package. Inside torchvision package, we have the module named transform, which has several functions helpful in transforming the images.

Step 5: Loading pre-trained model—We load the pre-trained models on the ImageNet dataset into our code. This mechanism is called transfer learning which is reusing things that have very high standards. This mechanism of transfer learning is gaining huge popularity in the field of deep learning. Pre-trained networks generally contain two things. One is a feature detector and the other is a classifier. Feature detectors extract from each image the information. Our classifier will learn the input given by feature layers and therefore, we will freeze the feature layers to avoid any modification. If we talk about the most commonly used pre-trained models on the ImageNet dataset, these are AlexNet, DenseNet, VGG-16, and many more that have gained popularity over recent years. All we save through the power of these trained feature models on huge datasets is our precious time and the computer's resources. These models provide cutting edge results on the smaller datasets when reused.

Step 6: Classifier Building—We will build our classifier and this classifier has to be replaced by the model pre-trained on ImageNet. We will freeze the feature layers of the network. This is because we will provide the input according to our dataset. The feature layer will learn these inputs and we don't want these layers to learn the same inputs as of ImageNet dataset. We will set the output size of the classifier as 102 as we have these many species in our dataset. We will achieve this task with the help of defining a precise function.

Step 7: Model Training—In this step, it is time to train our classifier on our flower's 102 category dataset. We will train the final layers of the network. We only

train the classifier parameter while the feature parameters are kept frozen. We can change our optimizer as well as a scheduler in the piece of our code.

Step 8: Model Testing—In this step, our trained model is evaluated to measure the performance of the test images of our dataset. At the end of this step, we obtain the percentage accuracy, which means how many flower test images are correctly classified.

Step 9: Save Model Checkpoint—We will save our model in the directory created. This is done to ensure the backup of our created and trained model. This will come in handy when we have to use this trained model on some unknown images of flowers.

Step 10: Load Model Checkpoint—We load our trained model to use this on the unknown flower images to predict their species name.

Step 11: Processing Images—We will carry out the processing of the images because we will take this image as the unknown image for which we need to predict the class label. So, there is a need of related data transformations.

Step 12: Class Prediction—We will predict the class of flower species of the given image unknown to the model. This whole process is carried out in the probability that a particular flower type belongs to that class.

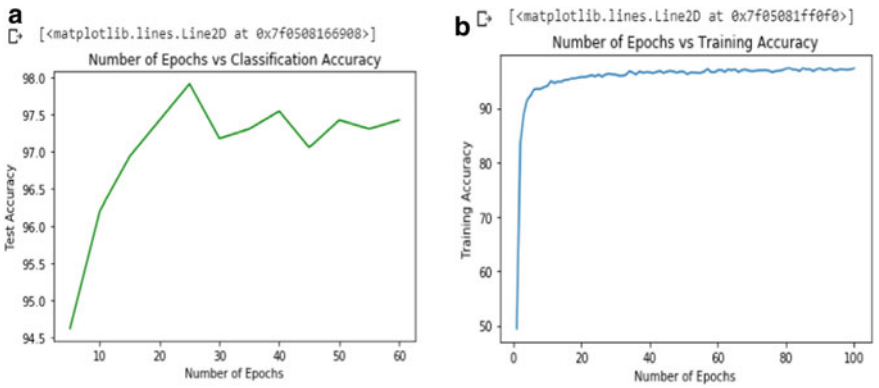
Step 13: Sanity Check—All the earlier parts of this code are combined in a function. This function performs the plotting, or we can say just graphing where the models predict with uncertainty.

7 Discussion of Result of the Proposed System and Comparison of the Results with the Existing Work

7.1 Number of Epochs Versus Classification Accuracy

In Graph 1a, we plotted 60 values of classification accuracies measured at epochs ranging from 1 to 60. This graph depicts the dependency of several epochs on the classification accuracy of our deep learning trained classifier. As the number of epochs increases, the accuracy also increases and reaches the maximum of 97.92 at epoch 25. From then onwards, accuracy declined and then increased again. Then, the graph follows a pattern of increase and decreases until epochs 60. But we observed that the accuracy could not cross the value of the maximum, which is 97.92% at any epochs after 25. So, we conclude that increasing the epochs after 25 do not affect or increase the classification value of the model. Graph 4 shows the results of the Proposed System.

In Graph 1, we have plotted the number of epochs on the x-axis and training accuracy on the y-axis. We plotted 100 values of training accuracies measured at the epochs, ranging from 1 to 100 in the training phase. This graph illustrates that training accuracy shows a sudden increase till epoch 15. Still, from there onwards, the accuracy doesn't show a desirable increase and there is also not drastic decrease in the training accuracy. This means that after epochs 15 the accuracy curve seems to



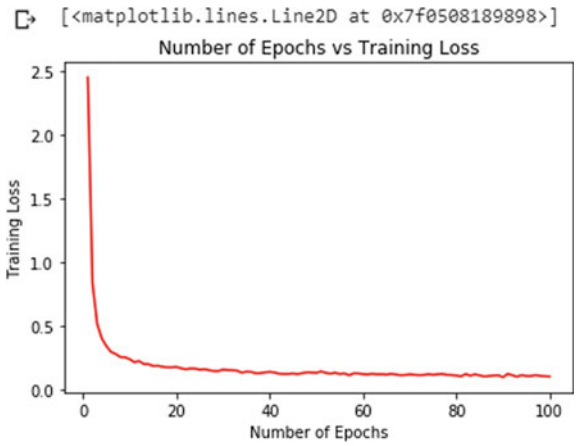
Graph 1 **a** and **b** Number of epochs versus classification accuracy and training accuracy

be constant as it keeps on increasing and decreasing only by a very small or negligible value.

7.2 Number of Epoch Versus Training Loss

In Graph 2, we have plotted the number of epochs on the *x*-axis and training loss on the *y*-axis. We have plotted 100 values of training loss measured at the epochs, which range from 1 to 100 in the training phase. This graph illustrates that training loss shows a sudden decrease till epochs 10, but the loss doesn't show a desirable decrease and there is also no drastic increase in the training loss. This means that after epochs 10 the loss curve seems to be constant as it keeps on decreasing and increasing but only by a very or negligible small value.

Graph 2 Number of epochs versus training loss

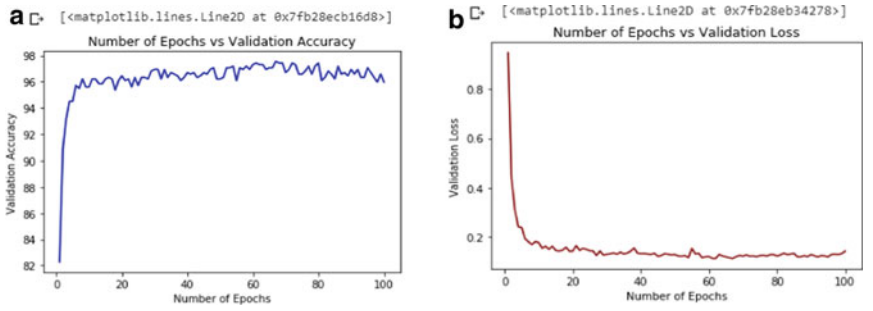


7.3 Validation Curve

Number of Epochs versus validation accuracy.

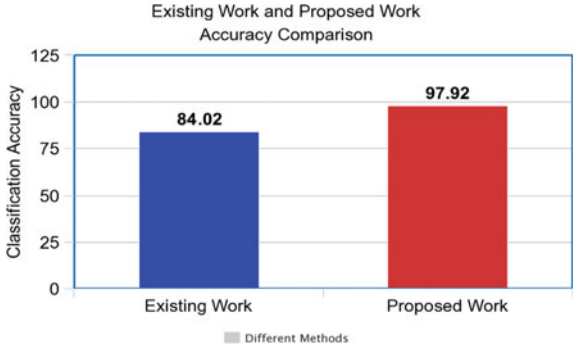
Graph 3a shows the dependency of the number of epochs on the validation accuracy. We have plotted 100 values of epochs on the *x*-axis and 100 values of validation accuracy on the *y*-axis. As the number of epochs increases, the accuracy of the validation points also increases. Still, this accuracy value increases only until the epoch value of 8 at a fast rate. Then, from epoch 8 the validation accuracy fluctuates. That is it keeps on increasing and decreasing till epochs 100. Overall, we can say that after epoch 8 the validation accuracy doesn't show a significant increase or also does not display a drastic decrease.

Graph 3b shows the dependency of the number of epochs on the validation loss. We have plotted 100 values of epochs on the *x*-axis and 100 values of validation loss on the *y*-axis. As the number of epochs increases, the loss of the validation points decreases, but this value of loss decreases only till the epoch value of 8 at a very fast rate. Then from epoch 8 the validation loss fluctuates. That is, it keeps on decreasing and increasing till epochs 100. Overall, we can say that after epoch 8 the validation loss doesn't show a significant decrease or also does not display a drastic increase.



Graph 3 **a** and **b** Number of epochs versus validation accuracy and validation loss

Graph 4 Existing work versus proposed work



7.4 Comparison of the Results of the Proposed System with the Exiting System

Graph 4 compares the existing work [34] in flower classification with the proposed method, which is robust and efficient. Both of the work is performed on the Oxford-102 dataset. The existing work achieved a classification accuracy of 84.02% with the use of five convolutional layers. Our developed deep learning flower classifier system has set the really high standards in this domain by achieving a very high recognition rate of 97.92%.

8 Conclusion and Future Work

This work has developed an efficient and robust flower species recognition classifier based on deep learning. We have used the dataset from the University of Oxford, which is the Oxford-102 flower dataset with a total of 8189 images of different categories of flower species [35, 36]. We have divided our dataset into training sets and validation set for evaluation purposes. We have employed PyTorch library by Facebook to code our research work. DenseNet161, a pre-trained model of the ImageNet dataset, was loaded to use its weights and later applied to our flower's dataset [11, 37]. This all result was achieved through transfer learning mechanism, which is gaining popularity in deep learning. We have developed a four-step novel approach for the classification of the 102 categories of flower species which is below;

1. The data Augmentation for better training of flower classifier.
2. The Flower Image Segmentation using the BiCoS method for removing the complex background.
3. The cropping of segmented flower images using python script.
4. The model training using the pre-trained model—DenseNet.
5. For training purposes, we have used Jupyter Colab Notebook, a free graphics processing unit (GPU) provided by Google Collaboratory. Our proposed method achieved very high accuracy on the flower's dataset, which is 97.92% classification accuracy. This is one of the best results obtained in the domain of flower species classification.

A deep learning-based CNN classifier is being developed in this work, one of the most robust and efficient with 97.92% classification accuracy on the benchmark dataset. But still, there exists some more work in this domain that can be done in future to use the system in the real world with high accuracy. Some future work points are an extension in the dataset having more categories, integration with mobile applications, and increase in the training data.

References

1. Singh SP, Solanki A, Singh T, Tayal A (2021) Internet of intelligent things: injection of intelligence into IoT devices. In: Artificial intelligence to solve pervasive internet of things issues. Academic Press, pp 85–102
2. Issa MB, Daraghmeh M, Jararweh Y, Al-Ayyoub M, Alsmirat M, Benkhelifa E (2017) Using logistic regression to improve virtual machines management in cloud computing systems. In: 2017 IEEE 14th international conference on Mobile Ad Hoc and Sensor Systems (MASS), 22–25 Oct. 2017
3. Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: Fleet DJ, Pajdla T, Schiele B, Tuytelaars T (eds) ECCV, volume 8689 of Lecture Notes in Computer Science. Spring, pp 818–833
4. Redmon J, Divvala S, Girshick R et al (2016) You only look once: unified, real-time object detection. In: Proceeding IEEE conference computer vision and pattern recognition, Las Vegas, NV, June 2016, pp 779–788
5. Singh T, Nayyar A, Solanki A (2020) Multilingual opinion mining movie recommendation system using RNN. In: Proceedings of first international conference on computing, communications, and cyber-security (IC4S 2019). Springer, Singapore, pp 589–605
6. Luong DTA, Chandola V (2017) A K-means approach to clustering disease progressions. In: IEEE Conference on 14 September 2017
7. Solanki A, Singh T (2021) COVID-19 epidemic analysis and prediction using machine learning algorithms. Emerging technologies for battling Covid-19: applications and innovations, pp 57–78
8. Rong F (2017) Audio classification method based on machine learning. In: IEEE Conference on 21 September 2017
9. Singh T, Mishra J (2021) Learning with artificial intelligence systems: application, challenges, and opportunities. In: Impact of AI technologies on teaching, learning, and research in higher education. IGI Global, pp 236–253
10. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Proceeding of international conference learning representations, San Diego, CA, May 2015, arXiv preprint arXiv:1409.1556
11. Hu W, Hu R, Xie N et al (2014) Image classification using multi-scale information fusion based on saliency driven nonlinear diffusion filtering. IEEE Trans Image Process 23(4):1513–1526
12. Nitta T (1993) A back-propagation algorithm for complex numbered neuralnetworks. In: Proceedings of 1993 International Joint Conference on Neural networks, IJCNN' 93-Nagoya. 25–29 Oct. 1993
13. Nilsback M, Zisserman A (2008) Automated flower classification over a large number of classes. In: Proceeding sixth indian conference computer vision, graphics and image processing, Bhubaneswar, India, December 2008, pp 722–729
14. Nilsback M, Zisserman A (2006) A visual vocabulary for flower classification. In: Proceeding IEEE conference computer vision and pattern recognition, New York, NY, June 2006, vol 2, pp 1447–1454
15. Pandey S, Solanki A (2019) Music instrument recognition using deep convolutional neural networks. Int J Inf Technolhttps://doi.org/10.1007/s41870-019-00285-y (SpringerPublication)
16. Singh T, Solanki A, Sharma SK (2021) Role of smart buildings in smart city—components, technology, indicators, challenges, future research opportunities. Digital Cities Roadmap: IoT-Based Architecture and Sustainable Buildings, pp 449–476
17. Bhardwaj N, Solanki A (2016) An efficient algorithm for color image segmentation. Selforganizology 3(3):87–99
18. Krizhevsky A, Sutskever I, Hinton G (2012) ImageNet classification with deep convolutional neural networks. In: Pereira F, Burges C, Bottou L et al (eds) Advances in neural information processing systems (Curran Associates, Inc., Red Hook, NY, USA, 2012), pp 1097–1105
19. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov R (2012) Improving neural networks by preventing co-adaptation of feature detectors, CoRR, abs/1207.0580

20. Szegedy C, Liu W, Jia Y et al (2014) Going deeper with convolutions. arXiv preprint arXiv:1409.4842
21. Yang M, Zhang L, Feng X et al (2014) Sparse representation based Fisher discrimination dictionary learning for image classification. Int J Comput Vis 109(3):209–232
22. Priyadarshni V, Nayyar A, Solanki A, Anuragi A (2019) Human age classification system using K-NN classifier. In: Luhach A, Jat D, Hawari K, Goa XZ, Lingras P (eds) Advanced informatics for computing research. ICAICR 2019. Communications in computer and information science, vol 1075. Springer, Singapore
23. Khan F, van de Weijer J, Vanrell M (2012) Modulating shape features by color attention for object recognition. Int J Comput Vis 98(1):49–64
24. Shelhamer E, Long J, Darrell T (2017) Fully convolutional networks for semantic segmentation. IEEE Trans Pattern Anal Mach Intell 39(4):640–651
25. Girshick R, Donahue J, Darrell T et al (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceeding IEEE conference computer vision and pattern recognition, Columbus, OH, June 2014, pp 580–587
26. Ren S, He K, Girshick R et al (2017) Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39(6):1137–1149
27. Girshick R (2015) Fast R-CNN. In: Proceeding of IEEE international conference computer vision, Santiago, Chile, December 2015, pp 1440–1448
28. Zhou H, Zheng J, Wei L (2013) Texture aware image segmentation using graph cuts and active contours. Pattern Recog-nition 46(6):1719–1733. https://doi.org/10.1016/j.patcog.2012.12.005
29. Xie L, Wang J, Lin W et al (2017) Towards reversal-invariant image representation. Int J Comput Vis 123(2):226–250
30. Hsu T, Lee C, Chen L (2011) An interactive flower image recognition system. Multimedia Tools Appl. 53(1):53–73
31. Mottos A, Feris R (2014) Fusing well-crafted feature descriptors for efficient fine-grained classification. In: Proceeding IEEE international conference image processing, Paris, France, October 2014, pp 5197–5201
32. Chai Y, Rahtu E, Lempitsky V et al (2012) TriCoS: a tri-level class discriminative co-segmentation method for image classification. In: Proceeding of European conference computer vision, Florence, Italy, October 2012, vol I, pp 794– 807
33. Chen Q, Song Z, Hua Y et al (2012) Hierarchical matching with side information for image classification. In: Proceeding IEEE Conference. Computer Vision and Pattern Recognition, Providence, RI, June 2012, pp 3426–3433
34. Liu Y, Tang F, Zhou D et al (2016) Flower classification via convolutional neural network. In: Proceeding of IEEE international conference functional-structural plant growth modeling, Simulation, Visualization and Applications, Qingdao, China, November 2016, pp 110–116
35. Chai Y, Lempitsky V, Zisserman A (2011) BiCoS: a Bi-level co-segmentation method for image classification. In: Proceeding of international conference computer vision, Barcelona, Spain, November 2011, pp 2579–2586
36. Qi X, Xiao R, Li C et al (2014) Pairwise rotation invariant co-occurrence local binary pattern. IEEE Trans Pattern Anal Mach Intell 36(11):2199–2213
37. Ciresan DC, Meier U, Schmidhuber J (2012) Multi-column deep neural networks for image classification. In: CVPR, pp 3642–3649