iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 7 **Issue:** XI **Month of publication:** November 2019

DOI: http://doi.org/10.22214/ijraset.2019.11079

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.177
Volume 7 Issue XI, Nov 2019- Available at www.ijraset.com

Analysis of Pre-Trained Convolutional Neural Networks to Build a Flower Classification System

Simran Gadkari[1], Jenell Mathias[2], Ashwini Pansare[3]
[1, 2, 3]*Fr. Conceicao Rodrigues College of Engineering, Department of Computer Engineering, Bandstand, Bandra (W), Mumbai 400050, India*

Abstract: ***Nature Tourism is an industry that is currently thriving and a lot of tourists visit places abundant in flora and fauna. A lot of times while hiking, tourists come across exotic looking flowers that mesmerize them but they are unable to discern what species the flowers belong to. The system proposed in this paper can help nature enthusiasts identify these flowers correctly. The proposed system examines the various features of the flower and identifies it by retraining models on flower datasets, using transfer learning methods. A review of three convolutional neural network models pre-trained on the ImageNet dataset, using the TensorFlow backend, was conducted to suggest the superior algorithm for flower classification systems in order to identify plant species accurately. The system was then implemented in the form of an Android mobile application that provides relevant information along with the species and family of the flower.***
Keywords: Index Terms-- ImageNet, Keras, TensorFlow, transfer learning

I. INTRODUCTION

Although plants have abundant similarities in them, they possess an equal amount of differences i.e. with respect to their shape, leaf type, and structure, plant shape, soil type, etc. Owing to the wide variety there is a greater need to classify them appropriately[13]. However, the classification of plants using leaves proved to be a complicated process due to the less prominent and extremely difficult to identify variations in the skeleton pattern. But due to the comparatively distinctive patterns and visible similarities, flowers have been proved as more suitable to classify and identify plant species correctly. Classification of Flowers using plants is of crucial importance in the fields of botany as well as other such scientific fields of study. There are innumerable plant species and each plant has a variety of flowers that differ in their color and texture. Therefore, flower classification is a complex task and requires adequate knowledge and research [1,2]. Nature enthusiasts and tourists are attracted to places of natural and cultural diversity. This diversity largely depends on the various kinds of plants, trees, and animals prevalent in those regions. Despite having plenty of knowledge about nature itself, these enthusiasts may not always be able to identify exotic local flowers. Since it is impossible to memorize all species of flowers, manual identification is a redundant process. As flower and plant classification is of utmost importance for various scientific experiments this process demands automation[2]. Although simple classification algorithms can be utilized for the task, the accuracy obtained is not sufficient to serve the purpose of automation. Thus we use the transfer learning models to improve the working of the proposed system. Initially, the basic methods for flower classification were based on their stages of growth, morphology, structure, etc. But recent algorithms input the image of the flower and extract features such as color, texture, shape, size, etc from the image[3]. Thus this paper consists of the exploration of various pre-trained models trained on the flower image dataset and the comparison of their respective workflow, architecture accuracies

II. LITERATURE REVIEW

A. Existing Systems

Existing systems using KNN, MLNN, BPNN, SVM, etc. that classify plants on the basis of their leaf shape, skeleton, color, texture do not give a desirable accuracy for precise classification[1]. Due to similar physical structures, leaves of different species tend to be often confused as the same and leaves from the same species may look different altogether. There are not as many systems for flower classification as there are for leaf classification for the identification of plants. Most of the existing systems are using leaves to identify plants. This can be problematic as discussed previously in this paper, the major reason being the high intraclass variations and inter-class similarities[1]. Another reason for leaves not being the most accurate way of identifying plants is that the climatic conditions and diseases that play a great role in the appearance of leaves. Many leaf classification systems have used SVM, KNN, Naive Bayes method, and CNNs. One of the existing systems for classification of flowers is using Random Forest Classifier method[11], reporting an average accuracy of 80.67% for a database of 10 different flowers. A more superior system[12] has used CNN and acquired an accuracy above 97%. It was done in two steps viz. Segmentation and Classification.

B. Drawbacks of Existing Systems

The various drawbacks of existing leaf-based systems are :

1) High inter-class similarities and intraclass variations.

2) The basic classification models depend largely on pre-processing images to eliminate the undesired background and suffer from a diminishing gradient.

3) Leaf recognition is limited to factors like climate and plant health because the external surface of the leaf can have a change in color due to diseases or climate changes.

III.PROBLEM DEFINITION

This paper proposes a review of three convolutional neural networks, pre-trained on the ImageNet dataset, on the basis of accuracy measure obtained by training each model on a custom dataset, generated by using web scraping tools like the Google Chrome extension Imageye and Image Downloader, keeping all the other parameters uniform and consistent. This paper also presents the implementation for the superior model out of the three specified models in the form of an Android mobile application.

IV.DATASET

As the pre-trained models are based on transfer learning technologies that are trained on various datasets such as the ImageNet dataset. The system uses a varied image dataset consisting of images of flowers of 17 different categories, to name a few, Beach moonflower, Frangipani, Angel's trumpet and Hawaiian Hibiscus [4].

These are flower images scraped from various websites using web scraping tools and Google Chrome extensions, Image Downloader (version 1.4.9) and Imageye (version 1.4.9). The Image Data belonging to 17 classes with approximately 350 images in each class, thus obtained is visible in the Figure. 1 below. This was further divided into the training dataset and validation dataset in the 80:20 ratio in order to proceed with the task of training the model and predicting the accuracy of the pre-trained models.

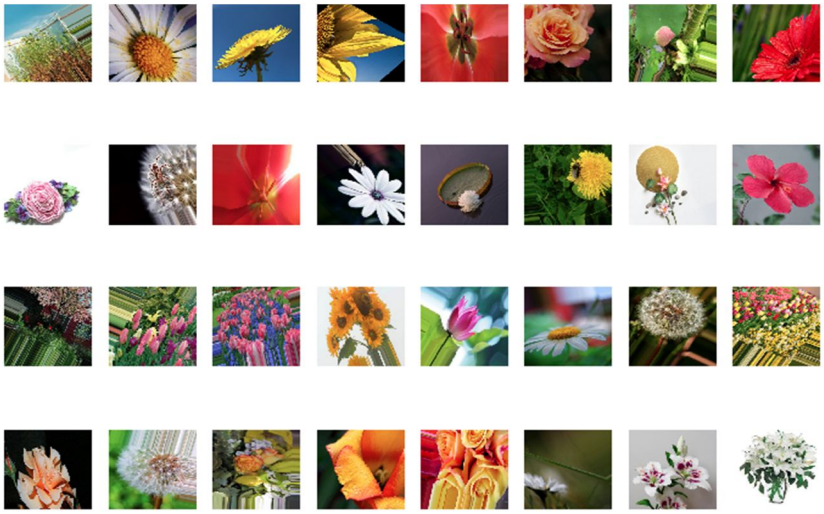


Figure 1: Flower Dataset

As a result, the final image dataset has a total of 4726 images belonging to 17 distinct classes in the training dataset and around 1000 such images belonging to 17 classes in the validation dataset.

V. DATA PRE-PROCESSING

The first step after generating the dataset and installing required dependencies was pre-processing the data, formally known as Image data augmentation. For this, the ImageDataGenerator class from the Keras library was used, for each of the three models. For uniformity, all three models used the same quantification and same parameters.

The rotation_range gives the range of degrees for randomly rotating the image. The width_shift_range and height_shift_range [9] parameters to the ImageDataGenerator constructor specify horizontal and vertical shift to the image. The shear_range randomly displaces each point in some fixed direction. The zoom_range argument gives specifies the zooming factor applied to the image. In the fill_mode argument, outliers are filled according to the given mode ('constant', 'nearest', 'reflect' or 'wrap').

The parameters used in the implementation are given in Table 1.

Image Data Generator Parameters	Values
preprocessing_function	preprocess_input
rotation_range	40
width_shift_range	0.2
height_shift_range	0.2
shear_range	0.2
zoom_range	0.2
horizontal_flip	True
fill_mode	'nearest'

Table 1 : Data augmentation parameters and values

VI. IMPLEMENTATION

The workflow of the system is presented with the help of the above diagram in Figure 2. The dataset was taken from the Kaggle flower dataset [4] and merged with custom generated image data scraped from a variety of websites using a Web Scraping tool i.e. Google Chrome's extension Imageye and Image downloader in order to prepare an image data of 17 classes. This image data were pre-processed using Image Data Generator a class of Keras.

After adequate rescaling, resizing, flipping, rotating, the dataset was split into a training dataset and a validation dataset. Earlier, classification algorithms such as SVM, Random Forest were used but since these algorithms are suitable for small scale datasets and capable of managing fewer outliers[15]. Since the system has to work with a large dataset and deal with more outliers, deep learning models are considered most capable of giving good performance and accuracy. The three models the system is based on are the VGG-16, VGG-19 and the InceptionV3. Each model is trained for 30 epochs with 35 steps under each epoch.

A. VGG-16

VGG 16 is a deep convolutional neural network that consists of 16 weight layers including thirteen convolutional layers with a filter size of 3 X 3, and fully-connected layers with a filter size of 3 X 3, and fully connected layers. The VGG Network architecture was introduced by Simonyan and Zisserman in their 2014 paper[5].

These were the findings of the ImageNet Challenge 2014. The number 16 stands for the number of layers in the neural network. The VGG 16 architecture consists of 12 convolutional layers along with a few maximum pooling layers the weights of which are frozen, i.e. the same weights of the model when trained on the ImageNet Dataset are used for other datasets[14,7]. The last four layers are trainable.

For the proposed system, a flattening layer, dense layer, dropout layer, softmax layer with an activation function ReLU is added at the end. The ReLU is non-linear and so backpropagation of errors can be easily done.

After the appropriate generation and the pre-processing of the image dataset, the VGG-16 model thus prepared is fit on the flower dataset thus giving rise to a training accuracy of 87.95% and validation accuracy of 85.73%. It was trained for 30 epochs and 35 steps per epoch. The approximate time it took to train the model was 5415 seconds.

```
35/35 [==============================] - 45s 1s/step - loss: 0.4834 - acc: 0.8545 - val_loss: 0.6702 - val_acc: 0.7766
Epoch 19/30
35/35 [==============================] - 44s 1s/step - loss: 0.4187 - acc: 0.8562 - val_loss: 0.6552 - val_acc: 0.8067
Epoch 20/30
35/35 [==============================] - 45s 1s/step - loss: 0.4486 - acc: 0.8628 - val_loss: 0.5550 - val_acc: 0.8356
Epoch 21/30
35/35 [==============================] - 44s 1s/step - loss: 0.3769 - acc: 0.8874 - val_loss: 0.5451 - val_acc: 0.8340
Epoch 22/30
35/35 [==============================] - 45s 1s/step - loss: 0.4530 - acc: 0.8571 - val_loss: 0.6374 - val_acc: 0.8397
Epoch 23/30
35/35 [==============================] - 44s 1s/step - loss: 0.4741 - acc: 0.8616 - val_loss: 0.4770 - val_acc: 0.8517
Epoch 24/30
35/35 [==============================] - 45s 1s/step - loss: 0.3809 - acc: 0.8741 - val_loss: 0.5039 - val_acc: 0.8345
Epoch 25/30
35/35 [==============================] - 44s 1s/step - loss: 0.3761 - acc: 0.8911 - val_loss: 0.5680 - val_acc: 0.8320
Epoch 26/30
35/35 [==============================] - 45s 1s/step - loss: 0.3441 - acc: 0.8911 - val_loss: 0.5694 - val_acc: 0.8366
Epoch 27/30
35/35 [==============================] - 44s 1s/step - loss: 0.3767 - acc: 0.8857 - val_loss: 0.8326 - val_acc: 0.8074
Epoch 28/30
35/35 [==============================] - 43s 1s/step - loss: 0.4321 - acc: 0.8589 - val_loss: 0.5406 - val_acc: 0.8411
Epoch 29/30
35/35 [==============================] - 44s 1s/step - loss: 0.3251 - acc: 0.8991 - val_loss: 0.5871 - val_acc: 0.8519
Epoch 30/30
35/35 [==============================] - 43s 1s/step - loss: 0.3520 - acc: 0.8795 - val_loss: 0.6144 - val_acc: 0.8573
```

Figure 2: Training and Validation Accuracy for VGG-16

B. VGG-19

The VGG-19 is very similar to VGG-16, it only differs in the number of weight layers. In VGG-19 the number 19 stands for the number of layers in the neural network.

The first 15 layers of the pre-trained VGG-19 model's weights are frozen. The last four layers are trainable with custom weights. A flattening layer, dense layer, dropout layer, softmax layer is added at the end.

```
35/35 [==============================] - 89s 3s/step - loss: 0.5109 - acc: 0.8455 - val_loss: 0.8690 - val_acc: 0.7541
Epoch 14/30
35/35 [==============================] - 87s 2s/step - loss: 0.5372 - acc: 0.8223 - val_loss: 0.6284 - val_acc: 0.8279
Epoch 15/30
35/35 [==============================] - 89s 3s/step - loss: 0.5110 - acc: 0.8500 - val_loss: 0.5936 - val_acc: 0.8099
Epoch 16/30
35/35 [==============================] - 88s 3s/step - loss: 0.5221 - acc: 0.8324 - val_loss: 0.9829 - val_acc: 0.7601
Epoch 17/30
35/35 [==============================] - 90s 3s/step - loss: 0.4920 - acc: 0.8491 - val_loss: 0.7179 - val_acc: 0.7695
Epoch 18/30
35/35 [==============================] - 88s 3s/step - loss: 0.4068 - acc: 0.8759 - val_loss: 0.5864 - val_acc: 0.8233
Epoch 19/30
35/35 [==============================] - 87s 2s/step - loss: 0.4905 - acc: 0.8518 - val_loss: 0.5902 - val_acc: 0.8401
Epoch 20/30
35/35 [==============================] - 88s 3s/step - loss: 0.4688 - acc: 0.8462 - val_loss: 0.6491 - val_acc: 0.7879
Epoch 21/30
35/35 [==============================] - 86s 2s/step - loss: 0.4526 - acc: 0.8607 - val_loss: 0.5860 - val_acc: 0.8370
Epoch 22/30
35/35 [==============================] - 89s 3s/step - loss: 0.4711 - acc: 0.8527 - val_loss: 0.7968 - val_acc: 0.7961
Epoch 23/30
35/35 [==============================] - 90s 3s/step - loss: 0.3571 - acc: 0.8894 - val_loss: 0.5760 - val_acc: 0.8451
Epoch 24/30
35/35 [==============================] - 92s 3s/step - loss: 0.4018 - acc: 0.8768 - val_loss: 0.6664 - val_acc: 0.8417
Epoch 25/30
35/35 [==============================] - 91s 3s/step - loss: 0.3908 - acc: 0.8911 - val_loss: 0.5102 - val_acc: 0.8618
Epoch 26/30
35/35 [==============================] - 93s 3s/step - loss: 0.4177 - acc: 0.8824 - val_loss: 0.5593 - val_acc: 0.8555
Epoch 27/30
35/35 [==============================] - 90s 3s/step - loss: 0.4369 - acc: 0.8625 - val_loss: 0.5890 - val_acc: 0.8248
Epoch 28/30
35/35 [==============================] - 88s 3s/step - loss: 0.4364 - acc: 0.8661 - val_loss: 0.6739 - val_acc: 0.8036
Epoch 29/30
35/35 [==============================] - 91s 3s/step - loss: 0.3921 - acc: 0.8848 - val_loss: 0.8409 - val_acc: 0.8140
Epoch 30/30
35/35 [==============================] - 87s 2s/step - loss: 0.3786 - acc: 0.8857 - val_loss: 0.5549 - val_acc: 0.8289
```

Figure 3: Training and validation accuracy for VGG-19

Thus the training accuracy for VGG-19 is 88.57% whereas the validation accuracy is 82.89%. It was trained for 30 epochs and 35 steps per epoch. The time taken to train the model is 8150 seconds.

C. Inception-V3

InceptionV3 is a convolutional neural network based on the original paper [6] "Rethinking the Inception Architecture for Computer Vision". The InceptionV3 model was trained on a dataset called ImageNet [7], with 1000 object classes for eg. keyboard, mouse, etc. and approximately a million images. This model has 48 layers[8].

InceptionV3 uses the transfer learning method. Transfer learning is the process of using a pre-trained model and fitting that model on a different dataset, using the same weights as were used when it was first trained on the original dataset.

After generating the dataset, pre-processing of the images was performed for better accuracy while testing. Then, the InceptionV3 model was compiled and fit to our dataset. The training accuracy of this model on our dataset is 96.39% and the validation accuracy is 91.76%. The model was trained for 30 epochs, 35 steps per epoch. The approximate time for the model to train was 4800 seconds. The accuracy pattern for the model implementation is given in Figure 14.

```
35/35 [==============================] - 82s 2s/step - loss: 0.1044 - acc: 0.9596 - val_loss: 0.2364 - val_acc: 0.9188
Epoch 21/30
35/35 [==============================] - 81s 2s/step - loss: 0.1094 - acc: 0.9592 - val_loss: 0.2306 - val_acc: 0.9203
Epoch 22/30
35/35 [==============================] - 80s 2s/step - loss: 0.0996 - acc: 0.9609 - val_loss: 0.1895 - val_acc: 0.9311
Epoch 23/30
35/35 [==============================] - 80s 2s/step - loss: 0.1025 - acc: 0.9604 - val_loss: 0.2180 - val_acc: 0.9266
Epoch 24/30
35/35 [==============================] - 81s 2s/step - loss: 0.1096 - acc: 0.9564 - val_loss: 0.2460 - val_acc: 0.9200
Epoch 25/30
35/35 [==============================] - 80s 2s/step - loss: 0.0970 - acc: 0.9623 - val_loss: 0.2010 - val_acc: 0.9309
Epoch 26/30
35/35 [==============================] - 83s 2s/step - loss: 0.1097 - acc: 0.9576 - val_loss: 0.2136 - val_acc: 0.9288
Epoch 27/30
35/35 [==============================] - 80s 2s/step - loss: 0.1132 - acc: 0.9583 - val_loss: 0.2429 - val_acc: 0.9188
Epoch 28/30
35/35 [==============================] - 80s 2s/step - loss: 0.1051 - acc: 0.9593 - val_loss: 0.2537 - val_acc: 0.9121
Epoch 29/30
35/35 [==============================] - 79s 2s/step - loss: 0.1104 - acc: 0.9555 - val_loss: 0.2535 - val_acc: 0.9147
Epoch 30/30
35/35 [==============================] - 79s 2s/step - loss: 0.0990 - acc: 0.9639 - val_loss: 0.2398 - val_acc: 0.9176
```

Figure 4: Train and Validation Accuracy for Inception V3

VII. PROPOSED SYSTEM

This model uses the live flow of data using Keras image data generator. The data generator sends the images in batches of 100 for training and 10 for testing, and each image is augmented to make the model more robust. The model was trained on a 1080ti GPU. It took about 80 minutes to train. After training the model with 30 epochs and 35 steps per epoch and different hyperparameters, the model was saved as a tflite file, allowing to simply import the trained model into an android app using TensorFlow mobile kit. The Android app allows users to either click a picture or upload a picture to the app, from the phone gallery. The app then preprocesses the image to fit the model's input criteria and then predicts the appropriate flower class. The softmax output is also displayed, giving the confidence with which the model predicts a flower.

The trained CNN model is then compressed to a tflite file with a size of less than 100 MB. This tflite file is imported into the user App. Now, the user can install the Android Application .apk and use the app to recognize a flower in real-time.

The workflow can be realized by the Figure. 5.

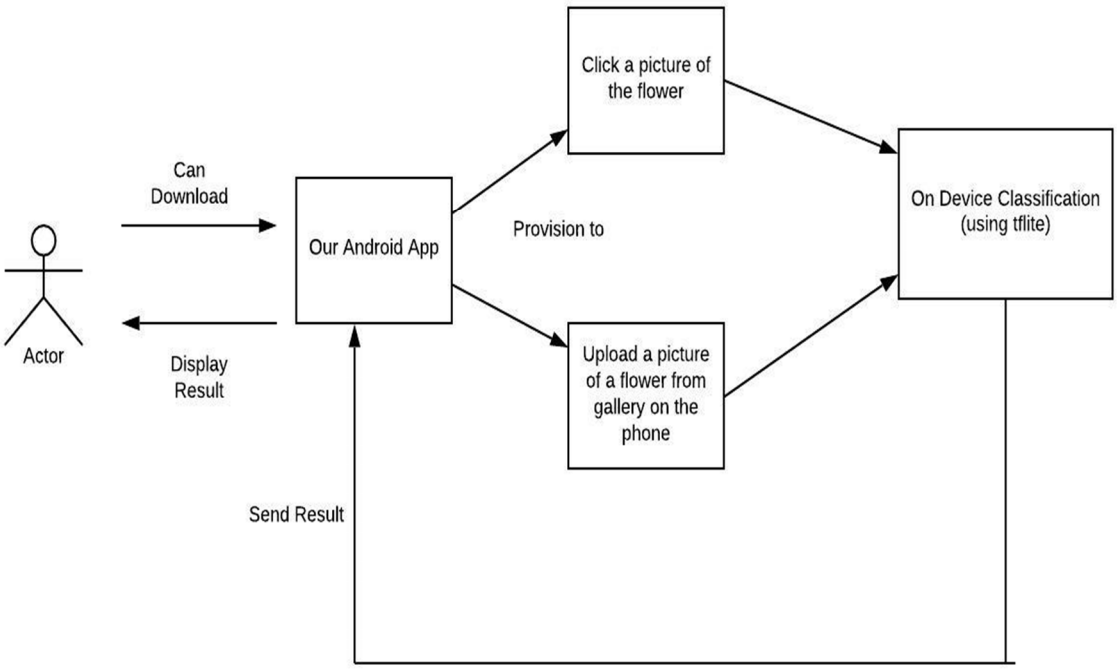


Figure 5: Workflow of the Proposed System

Screenshots of the application can be visualized in the following Figure 6.

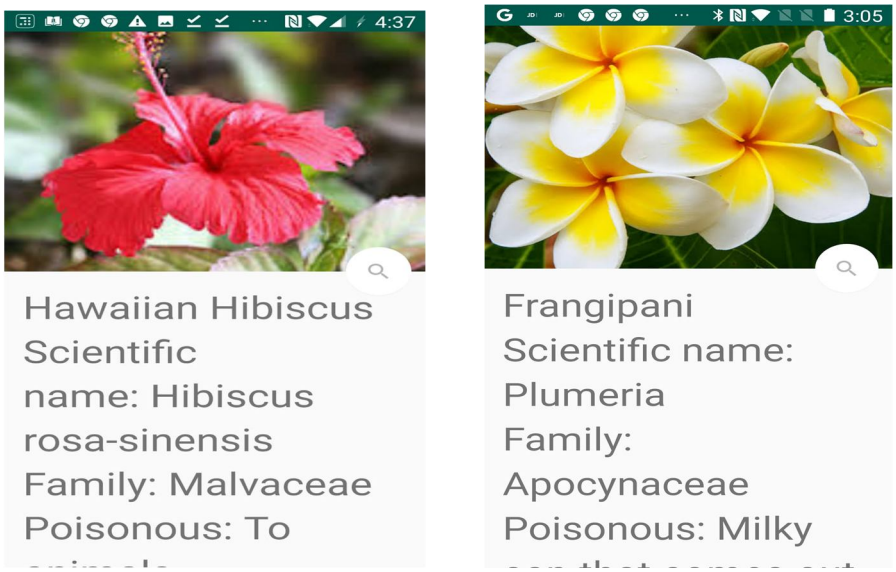


Figure 6: Screenshots of the Android Application

VIII. RESULTS

The comparative analysis in the table below clearly shows the superiority of the InceptionV3 model. The dataset, data augmentation parameters and preprocessing function, all being the same for all three models, the following results show a non-biased, uniform and transparent comparison between the three models.

	VGG-16	VGG-19	InceptionV3
train accuracy	87.95	88.57%	96.39%
validation accuracy	85.73	82.89	91.73%
The time taken to train in seconds	5415s	8150s	4800s

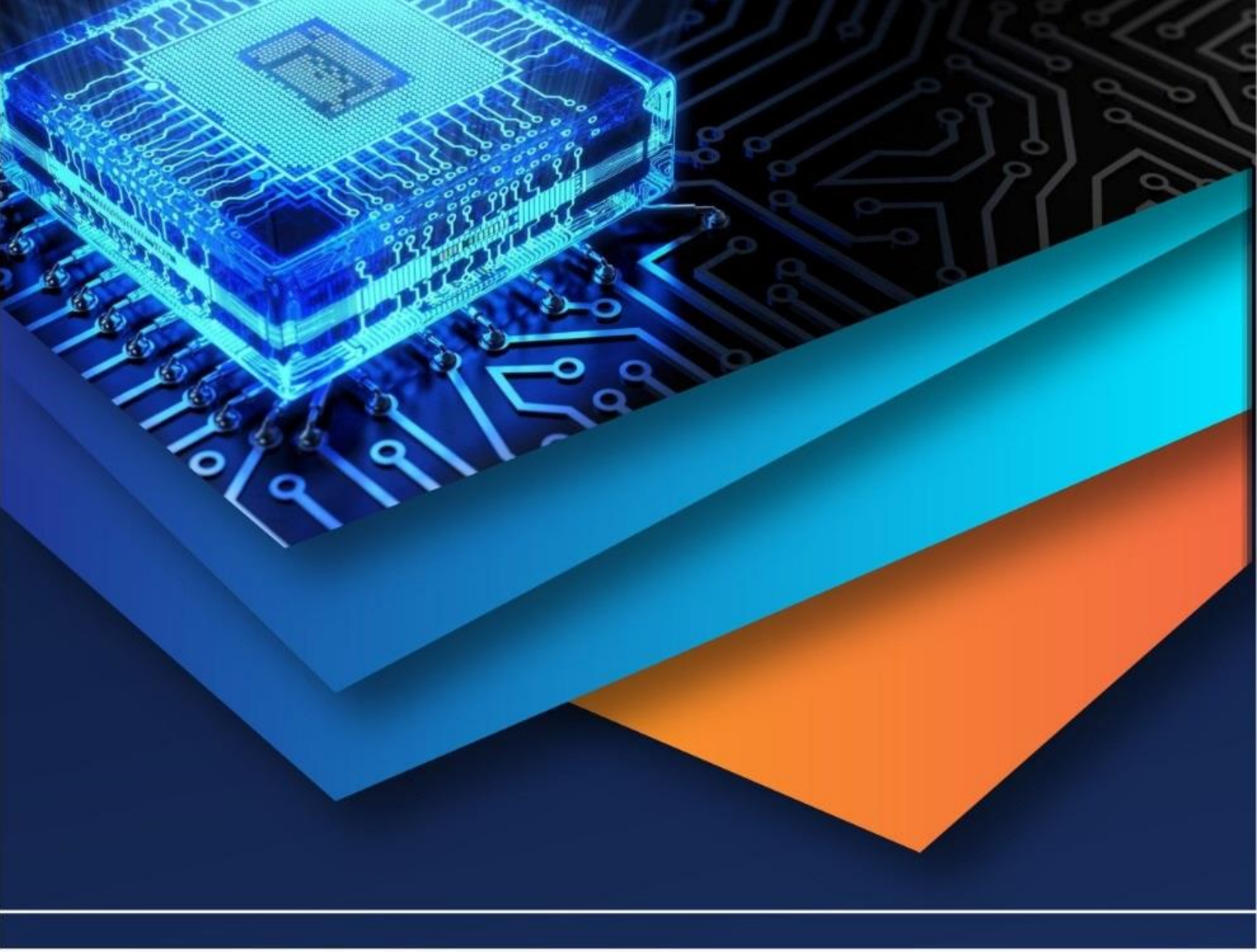
Table 2: Comparative analysis of VGG-16, VGG-19, and InceptionV3

IX. CONCLUSION

This paper presented a review of three convolutional neural networks viz. VGG-16, VGG-19, and InceptionV3 on the basis of train and validation accuracy, with InceptionV3 giving the most desirable results with a training accuracy of 96.39% and a validation accuracy of 91.73%. The number of epochs, steps per epoch, dataset and pre-processing function along with parameters and values was constant for all three models. The Inception V3 model was realized as an Android Application that could detect accurately, a flower from 17 classes of flowers.

REFERENCES

[1] Marco Seeland, Michael Rzanny, David Boho, Jana Wäldchen &Patrick Mäder, "Image-based classification of plant genus and family for trained and untrained plant species", 3 Jan. 2019.
[2] Hulya Yalcin, Salar Razavi "Plant classification using convolutional neural networks - IEEE ", 29 Sep.2016.
[3] Xiaoling Xia, Cui Xu, Bing Nan, "Inception-v3 for flower classification - IEEE Conference", Published in 2017 2nd International Conference on Image, Vision and Computing (ICIVC), Added to IEEE *Xplore* on 20 July 2017
[4] "Flowers Recognition | Kaggle." https://www.kaggle.com/alxmamaev/flowers-recognition. (Accessed 22 Oct. 2019.)
[5] Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", arXiv:1409.1556v6, Submitted on 4 Sep 2014 (v1), last revised 10 Apr 2015 (this version, v6))
[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, "Rethinking the Inception Architecture for Computer Vision ", The Computer Vision Foundation, Submitted on 2 Dec 2015 (v1), last revised 11 Dec 2015 (v3) arXiv:1512.00567v3
[7] "ImageNet." http://www.image-net.org/. (Accessed 21 Oct. 2019.)
[8] "Advanced Guide to Inception v3 on Cloud TPU" 28 Jan. 2019, https://cloud.google.com/tpu/docs/inception-v3-advanced.
[9] "How to Configure Image Data Augmentation in Keras." 12 Apr. 2019, https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/(Accessed 22 Oct. 2019.)
[10] "VGG 16- An Advanced Approach Towards Accurate Large Scale Image Recognition", 6 Sep. 2017, https://www.techleer.com/articles/305-vgg-16-an-advanced-approach-towards-accurate-large-scale-image-recognition/.(Accessed 24 Oct. 2019.)
[11] Sripian, Peeraya & Yusungnern, Prawaran" Flower Identification System by Image Processing.", 2015.
[12] Hazem Hiary, Heba Saadeh, Maha Saadeh, Mohammad Yaqub,“ Flower Classification using Deep Convolutional Neural Networks”, The Institution of Engineering and Technology, 2015
[13] Hossam M. Zawbaa, Mona Abbass, Sameh H. Basha, Maryam Hazman, Abul Ella Hassenian, “An automatic flower classification approach using machine learning algorithms”, 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 24-27 Sept. 2014
[14] "VGG16 - Implementation Using Keras - engMRK." 6 Oct. 2018, https://engmrk.com/vgg16-implementation-using-keras/. (Accessed 24 Oct. 2019.)
[15] "When Does Deep Learning Work Better Than SVMs or Random." https://www.kdnuggets.com/2016/04/deep-learning-vs-svm-random-forest.html (Accessed 24 Oct. 2019.)



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY