Flower classification

1. Introduction

Earth is rich in biodiversity. Flowers, the most attractive part of a plant are found everywhere in this environment that may be in the park, garden, roadside, lakes, etc. there are around 300,000 flower species found in the world. Flower identification is often performed manually by a skilled botanist or taxonomist. It is difficult to identify different species manually by human eyes because of their similar appearances such as shape, size, and colour. Therefore, this is considered to be a complicated and time-consuming process. Generally, it is difficult for common people to know all the flowers accurately.

Artificial Intelligence is changing the technological era in a splendid manner. Machine Learning (ML) in a subset of Artificial Intelligence. It helps the computer system to automatically learn with a set of algorithms. Without being explicitly programmed, it helps the system to improve gradually. ML is used for various applications like spam detection, computer vision, healthcare, biometric recognition, and many more. The development of new technology has helped people to more conveniently using mobile phones to identify and know the flower species instantaneously. This kind of automatic flower detection using images has received ample attention.

In this paper, a machine learning model is developed which uses around 42,000 images of flowers for training which include 56 classes of flower species. ML algorithms can recognize the flower automatically with a high rate of accuracy. These ML models are programmed using python programming language as it is simple and consists of various libraries. This model identifies different flower species using a mobile phone camera. This helps the common people to identify the variety of flower species efficiently and accurately.

2. Data Collection

2.1 Data Cleansing

We had collected 50,000 images of flowers from Google images and other sources for training which include 56 classes of flower species. There were many duplicates, irrelevant, flower clusters, a mixture of other flower and low quality (clearly not visible) flowers images in it. we are filtering and removing these images by visually. In the end our flower data images came 42,000 quantities.

2.2 Data augmentation

For better generalization of the processed dataset, we look into this problem by executing some augmentation steps for the flower images. Originally we have images with noticeable black regions, and some text included images, In order to whittle away such regions we at the beginning resized them to 224x224 pixels. And In data augmentation part we flipped images horizontally and vertically, and rotated the flower image into 90 degrees, our dataset have 56 types of different flowers, so to perform augmentation part we have used basic python libraries like numpy, pandas etc. We have reported these augmentation steps in Figure 1.

(**Fig-1:** Orchid Flower Dataset)

Our augmentation technique improves the existing contour to transform method on flower images in terms of processing the imbalanced data and training technique on the state of-the-art vgg16 models

2.3 Data Preprocessing

Within the field of image analysis, image preprocessing techniques are frequently used in combination with a variety of image classification algorithms. We use 58 types of different flowers in our dataset. To process our dataset we use different image preprocessing techniques aimed to either reduce noise or enhance desired features. We discussed a variety of image preprocessing methods for training our dataset in image classification. Within the various preprocessing techniques discussed, the researchers highlighted the importance of thresholding in image processing. We use VGG16 model for feature extraction and pre-trained our dataset. The other image preprocessing techniques included eliminations of unwanted features and extraction of key features.

As described with the prior models, the data preparation involved standardizing the shape of the input images to small squares and subtracting the per-channel pixel mean calculated on the training dataset. “During training, the input to our ConvNets is a fixed-size 224 × 224 RGB image. The only preprocessing we do is subtracting the mean RGB value, computed on the training set, from each pixel [1].”

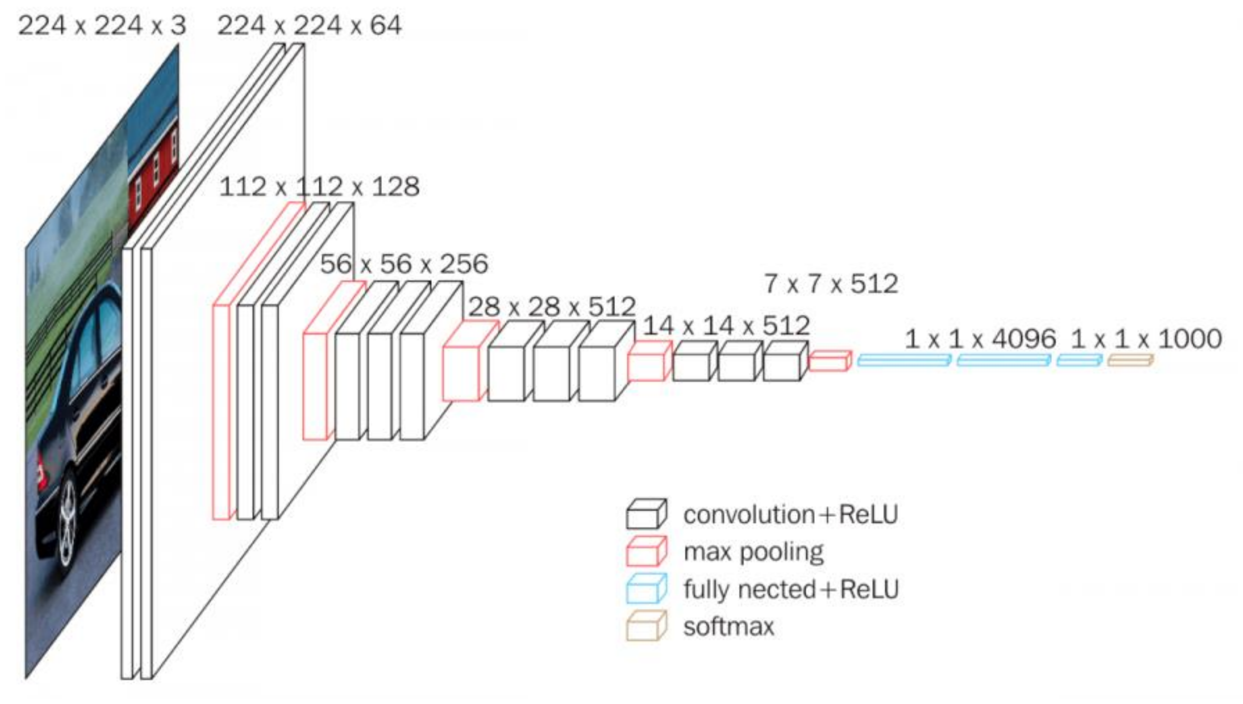
References

[1]— Very Deep Convolution Networks for Large-Scale Image Recognition, 2015.

3. <u>MODEL</u>

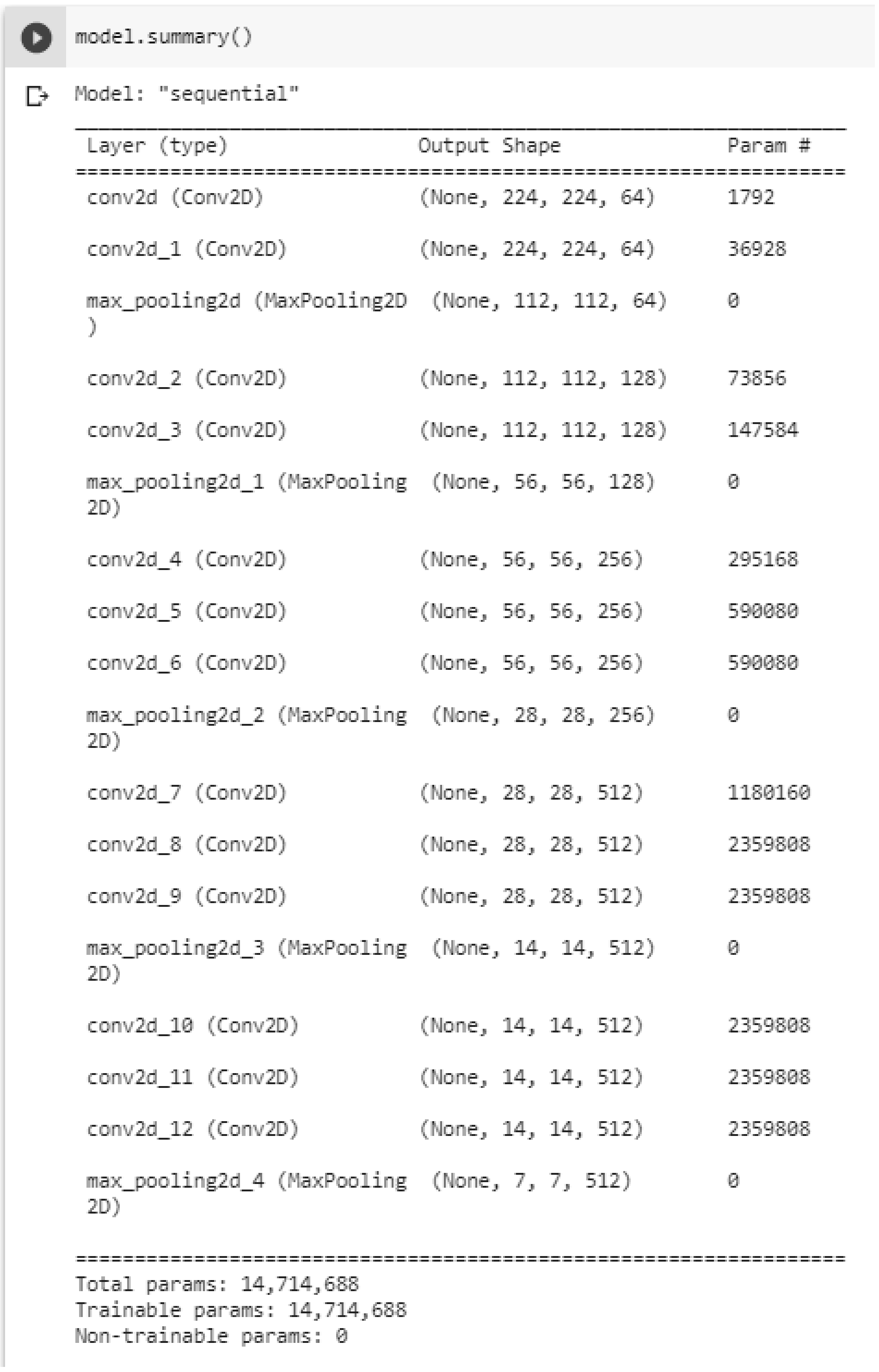
3.1 VGG16

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”[1]. This model achieved 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.



In this paper[1] they build a CNN as above. The architecture contains 8 convolutional layers followed by fully connected dense layers. While working on flower classification we removed those fully connected dense layer. Hence we used VGG16 for feature Extraction and then we tried feeding the extracted features to other machine learning models like Random Forest.

Here we have started with initializing the model by specifying that the model is a sequential model. After initializing the model we added
→ 2 x convolution layer of 64 channel of 3x3 kernal and same padding
→ 1 x maxpool layer of 2x2 pool size and stride 2x2
→ 2 x convolution layer of 128 channel of 3x3 kernal and same padding
→ 1 x maxpool layer of 2x2 pool size and stride 2x2
→ 3 x convolution layer of 256 channel of 3x3 kernal and same padding
→ 1 x maxpool layer of 2x2 pool size and stride 2x2
→ 3 x convolution layer of 512 channel of 3x3 kernal and same padding
→ 1 x maxpool layer of 2x2 pool size and stride 2x2
→ 3 x convolution layer of 512 channel of 3x3 kernal and same padding
→ 1 x maxpool layer of 2x2 pool size and stride 2x2
we also added Relu(Rectified Linear Unit) activation to each layers so that all the negative values are not passed to the next layer.



```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 64)      1792      
                                                                 
 conv2d_1 (Conv2D)           (None, 224, 224, 64)      36928     
                                                                 
 max_pooling2d (MaxPooling2D  (None, 112, 112, 64)     0         
 )                                                               
                                                                 
 conv2d_2 (Conv2D)           (None, 112, 112, 128)     73856     
                                                                 
 conv2d_3 (Conv2D)           (None, 112, 112, 128)     147584    
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 56, 56, 128)      0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 56, 56, 256)       295168    
                                                                 
 conv2d_5 (Conv2D)           (None, 56, 56, 256)       590080    
                                                                 
 conv2d_6 (Conv2D)           (None, 56, 56, 256)       590080    
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 28, 28, 256)      0         
 2D)                                                             
                                                                 
 conv2d_7 (Conv2D)           (None, 28, 28, 512)       1180160   
                                                                 
 conv2d_8 (Conv2D)           (None, 28, 28, 512)       2359808   
                                                                 
 conv2d_9 (Conv2D)           (None, 28, 28, 512)       2359808   
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 14, 14, 512)      0         
 2D)                                                             
                                                                 
 conv2d_10 (Conv2D)          (None, 14, 14, 512)       2359808   
                                                                 
 conv2d_11 (Conv2D)          (None, 14, 14, 512)       2359808   
                                                                 
 conv2d_12 (Conv2D)          (None, 14, 14, 512)       2359808   
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 7, 7, 512)        0         
 2D)                                                             
                                                                 
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
```

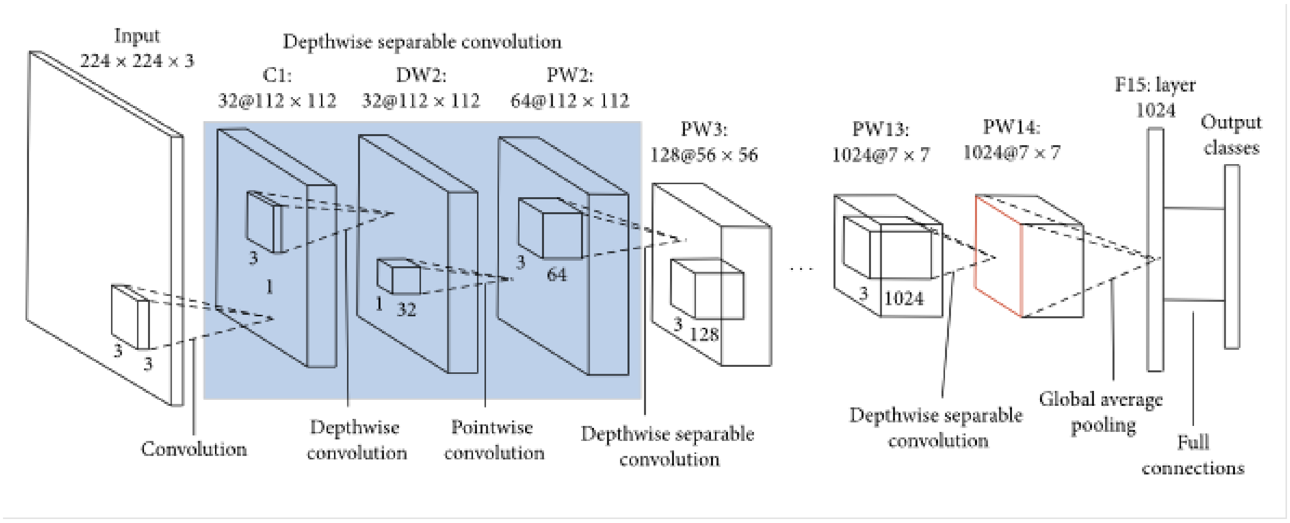
This model gives features of shape 7*7*512 we then reshaped these features to single dimention vector of size 25088(7x7x512=25088) then we fed these features to our Machine Learning Model.

References:-

[1] Simonyan, Karen & Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556.

3.2 MOBILENET

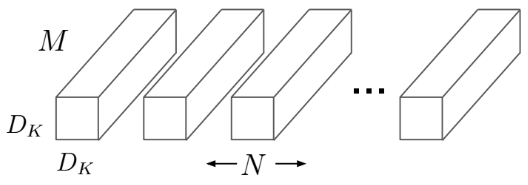
MobileNet is a CNN architecture model for Image Classification and Mobile Vision. It is a streamlined architecture that uses depth-wise separable convolutions to construct lightweight deep convolutional neural networks and provides an efficient model for mobile and embedded vision applications.
There are other models as well but the reason we opted for MobileNet is that it consumes less computation power to run or apply transfer learning to, thus making it a perfect fit for our project as well for the Embedded systems and computers without GPU or low computational efficiency without compromising significantly with the accuracy of the results.



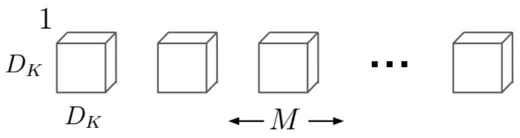
The structure of MobileNet is based on depth-wise separable filters, as shown in Figure 1

Fig1: Architecture of Mobilenet.

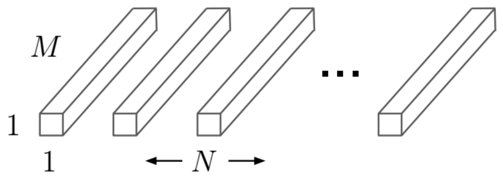
Depth-wise separable convolution filters are composed of depth-wise convolution filters and point convolution filters. The depth-wise convolution filter performs a single convolution on each input channel and the point convolution filter combines the output of depth-wise



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



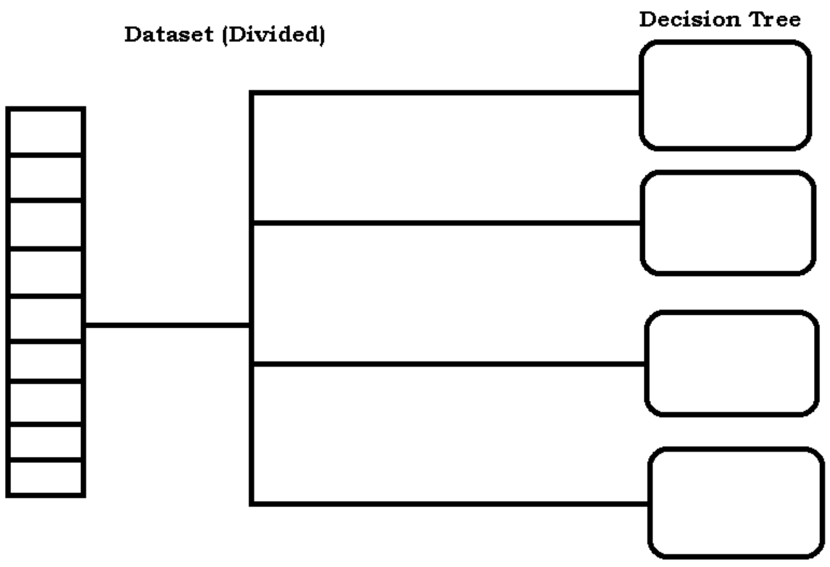
(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

convolution linearly with $1 * 1$ convolutions, as shown in Figure 2.

3.3 Random Forest Classifier

Random Forest is an ensemble classifier which makes use of multiple Decision Trees to arrive at a decision. Random Forest is capable of processing large volume of data in less time. Each tree in Random Forest is Binary and follows top-down approach.

Initially the data is divided into n number of datasets then each dataset is fed to a tree. The number of trees to be used is decided by Hyperparameter Optimization.



We fed the data to 100 decision trees.

4. Results:

This section covers all the details of an experiment carried out on the flower dataset. Moreover, it contains the accuracy results of considered algorithms.

The main goal of this work is to correctly classify the flower images from the collected dataset. The proposed algorithms like CNN (Convolutional Neural Networks), RANDOM FOREST are applied to the flower's dataset (collected by us). The entire dataset consists of 56 different classes of flowers found in India region. Each flower image from each class in the dataset is pre-processed by reducing its dimensions to 224*224 which will improve the computational speed of CNN.

After applying the appropriate pre-processing and segmentation techniques, the features are extracted and the dataset is prepared to apply the proposed algorithms.

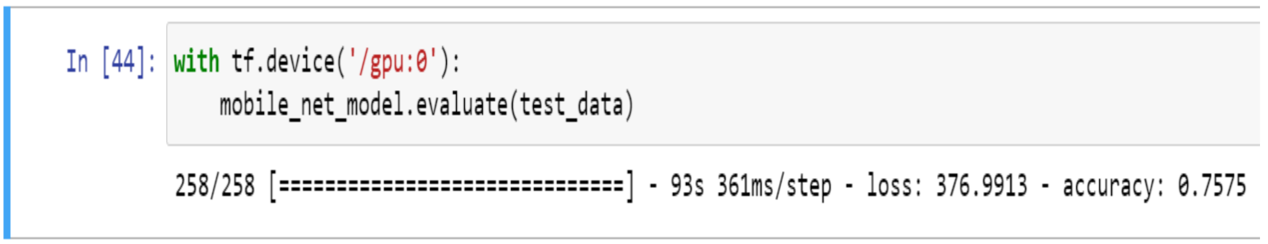
Accuracy Results of Proposed Algorithms:

The following table depicted the performance of CNN, RANDOM FOREST with its training and validation accuracy.

Performance analysis with various classifier

Algorithm	**Testing Accuracy**	**Validation Accuracy**
1. MobileNet	75.75 %	78.85%
2.Mobi-net V2	79.20 %	80.89%
3. RANDOM FOREST	57.12%	-

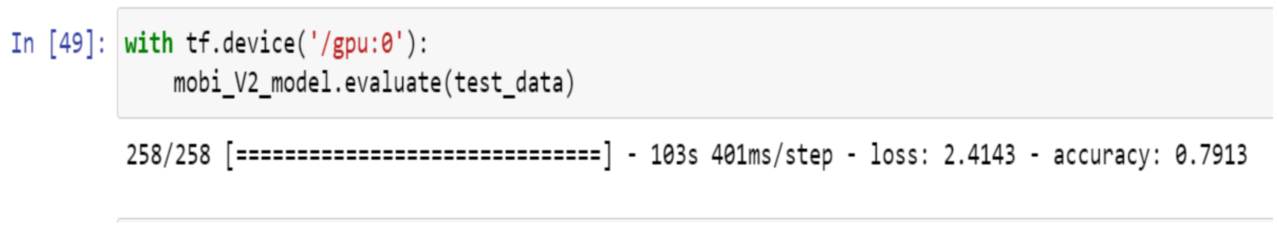
1. The first model build using convolutional neural network (MobileNet) gives validation accuracy of 78.85% and test accuracy of 75.75%. we find that neural network learns from existing feature and with the help of its weights and biases, it can predict the more accurate outcomes.



```
In [44]: with tf.device('/gpu:0'):
             mobile_net_model.evaluate(test_data)

258/258 [==============================] - 93s 361ms/step - loss: 376.9913 - accuracy: 0.7575
```

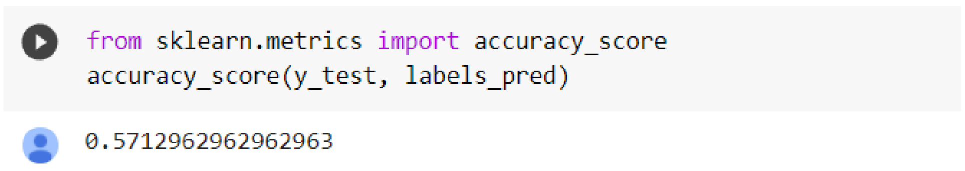
2. The Mobi-NetV2 model achieves gives the validation accuracy of 80.89% and test accuracy of 79.13%. The Mobi-Net v2 architecture is based on an inverted residual structure where the input and output of the residual blocks are thin bottleneck layers opposite to traditional residual models which use expanded representations in the input. Mobile Net v2 uses lightweight depth-wise convolutions to filter features in the intermediate expansion layer.



```
In [49]: with tf.device('/gpu:0'):
             mobi_V2_model.evaluate(test_data)

258/258 [==============================] - 103s 401ms/step - loss: 2.4143 - accuracy: 0.7913
```

3. An accuracy of 57.12% is achieved using Random Forest Classifier. It constitutes decision trees on arbitrarily selected data samples, gets a forecast from each tree and selects the best result by means of voting. It also provides a perfect indicator of feature importance.



```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, labels_pred)

0.5712962962962963
```

5. Conclusion:

MobileNet v1:

Mobile-Nets are **small, low-latency, low-power models parameterized to meet the resource constraints of a variety of use cases**. They can be built upon for classification, detection, embeddings and segmentation similar to how other popular largescale models, such as Inception, are used. Mobile-Net is a **type of convolutional neural network designed for mobile and embedded vision applications**. They are based on a streamlined architecture that uses depth-wise separable convolutions to build lightweight deep neural networks that can have low latency for mobile and embedded devices.

The accuracy results of the models are as follows:

MobileNetV1 Validation Accuracy - 78.85%
MobileNetV1 Testing Accuracy - 75.75%

MobileNet V2:

MobileNet-v2 is **a convolutional neural network that is 53 layers deep**. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. MobileNetV2 is a powerful classification model that is able to reach state of - the - performance through transfer learning. Mobile-Net V2 is a very effective **feature extractor for object detection and segmentation**.

MobileNetV2 Validation Accuracy - 80.89%
MobileNetV2 Testing Accuracy - 79.20%

In V1 the pointwise convolution either kept the number of channels the same or doubled them. In V2 it does the opposite: **it makes the number of channels smaller**. This is also a 1×1 convolution. Its purpose is to expand the number of channels in the data before it goes into the depth-wise convolution.

Random Forest:

Random Forest is the most pliable and easy to use supervised learning technique. A forest is composed of a tree. Forest is more vigorous if it has more tree, A random forest constitutes decision trees on arbitrarily select ted data samples, gets a forecast from each tree and selects the best result by means of voting. It also provides a perfect indicator of feature importance. The random forest has a range of applications, like recommendation engines, image classification, and attributes selection.

Random Forest Testing Accuracy - 57.12%

The Mobile-Net V2 has given very good accuracy of 80.89% on validation and 79.20% on testing than mobile-net v1 which has given accuracy of 78.85% on validation and 75.75% on testing and Random Forest which has given accuracy of 57.12% on Test sample. Mobile Net V2 has given 4-5% better accuracy than Mobile Net v1 and a better accuracy of about 20% than Random Forest.