Dog Breed Identifier Android Application

Report submitted in fulfilment of the requirements for the Exploratory Project of

Second Year B.Tech.

by

- ***Kshitij Sharma (18075030)***
- ***Kundan Kumar (18075033) &***
- ***Madhur Sahu (18075038)***

Under the guidance of

Prof. Sanjay Kumar Singh

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI
Varanasi 221005, India

June 2020

Dedicated to

My parents, teachers...

Declaration

I certify that

1. The work contained in this report is original and has been done by myself and the general supervision of my supervisor.
2. The work has not been submitted for any project.
3. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.
4. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT (BHU) Varanasi

Date: ***10 June 2020***

Kshitij Sharma (18075030)

Kundan Kumar (18075033) **&**

Madhur Sahu (18075038)

B.Tech Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi,

Varanasi, INDIA 221005

Certificate

*This is to certify that the work contained in this report entitled "**Dog Breed Identifier android app**" being submitted by **Kshitij Sharma (18075030)** , **Kundan Kumar (18075033)** and **Madhur Sahu (18075038)** , carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bonafide work of our supervision.*

Prof. Sanjay Kumar Singh
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Place: IIT (BHU) Varanasi
Date: ***10 June 2020***

Acknowledgment

We would like to express our sincere gratitude to the people who helped us throughout the project. We would like to express our special gratitude and thanks to **Dr. Sanjay Kumar Singh** for providing us an opportunity to work on such a great project entitled "**Dog Breed Identifier Android App**". A special thanks to **Abhinav** sir for providing necessary guidance and making the project possible. We wish to thank our parents for their support and attention. At last but not the least we would like to thank our friends who encouraged us and helped us out in finalizing the project.

Place: IIT (BHU) Varanasi
Date: ***10 June***

Kshitij Sharma (18075030)
Kundan Kumar (18075033)
Madhur Sahu (18075038)

Abstract

This Project deals with an application of **Deep Learning (Especially Convolutional Neural Network)** and **Image processing** to predict the breed of a dog out of more than 133 known breeds with more than 93% accuracy. We have developed a real time android app which collects image from camera of the user and shows probability percentages of top 3 breeds predicted by our trained model. The app uses a tensor flow model which was trained on more than 35000 images of dogs tagged with their breeds using **Transfer Learning**. The model consists of a deep web of several **Convolution and Pooling** layers. We have used **TensorFlow lite** module of android which interprets the input and output dimensions of the model and converts the image into the required dimensions. Our application takes input from camera or gallery and converts it to 224×224×3 **Byte Array** and calculates the prediction function using the trained model which is nothing but the probability distribution (an array of probabilities) for all breeds. The app shows you the breed names and probabilities of the three most likely breeds.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Our Project deals with development of an android application that can identify the breed of a dog if the image of the dog is provided as input. It is an application of **Deep Learning (Especially Convolutional Neural Network)** and **Image Classification.** The app predicts the breed of a dog out of more than 133 known breeds with more than 93% accuracy. We have developed a real time android app which collects image from camera of the user and shows probability percentages of top 3 breeds predicted by the trained model. The app uses a tensor flow model which was trained on more than 35000 images of dogs tagged with their breeds using **Transfer Learning.** The model consists of a deep web of several **Convolution and Pooling** layers.The project includes Two major parts each of which consists of several steps (to be discussed in following chapters):

1. **Designing and Training the CNN model using Transfer Learning** **&**
2. **App Development using Android Studio**

- **Designing and Training the CNN model using Transfer Learning:**

    1. Designing a TensorFlow/keras model.
    2. Setting up Working environment and directories
    3. Downloading the Dataset
    4. Setting partitions of the dataset for training, testing and validation.
    5. Populating the training Data.
    6. Storing the List of Labels in a directory
    7. Create a CNN to Classify Dog Breeds
    8. Training, testing and validation
    9. Downloading the ".tflite" file of the trained model

- **App Development using Android Studio:**

    1. Setting up working environment and the name of the app.
    2. Designing the basic structure of the app.
    3. Adding the activities and buttons accordingly
    4. Setting up the Assets folder.
    5. Asking permissions for access of camera and file manager
    6. Defining the functions of buttons and activities.
    7. TensorFlow interpreter interprets the input/output configuration of the model.
    8. Converting the input into required format/pre-processing of image
    9. Run the interpreter and show the results.

1.2 Motivation of the Project Work

We were really excited while doing the project work because of our specific interest in Machine learning. Learning Android development came out as another motivation for us. Learning new skills, creating something useful on our own and watching it working kept us motivated throughout the project.

Chapter 2

Designing and Training the CNN model using Transfer Learning

2.1 Designing a TensorFlow/keras model

Our project consists of a model based on deep learning. The model contains a deep web of an input layer of size =224*224*3, followed by several **convolutions** and **pooling** layers each of them are further activated with **ReLu** activation function. Finally, we have a **fully connected layer** of size equal to number of trained breeds which is activated by **SoftMax** function. The basic Design of the model is explained in the figure:

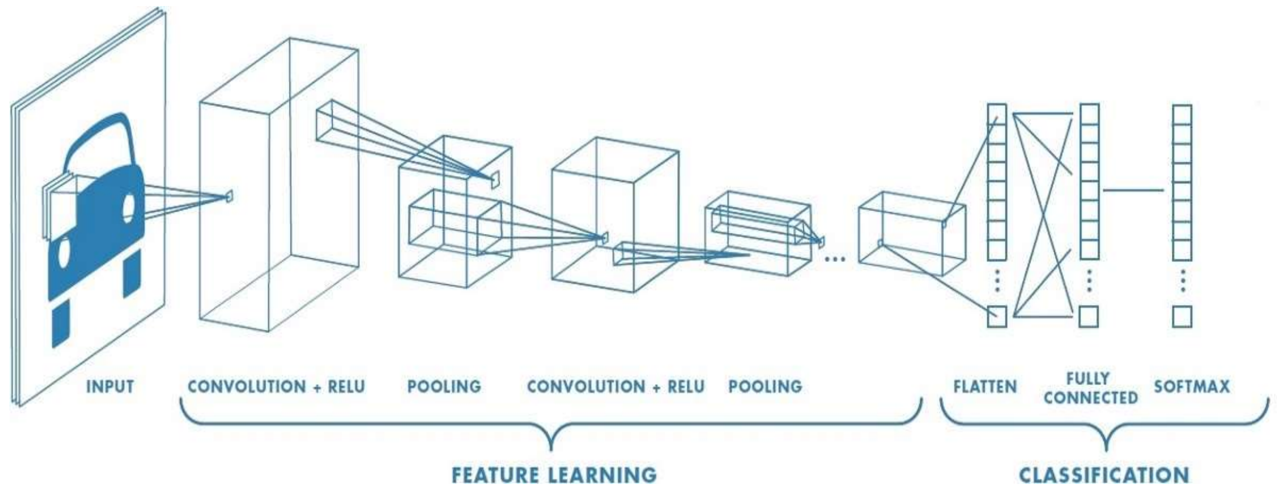


Fig 2.1 Basic Design of the model used in the project.

The Input Layer: A Byte Array of size 224*224*3
The Output Layer: A vector of size equal to number of breeds which stores probabilities of each breed.

2.2 Setting up Working environment and directories

The Whole training and testing work of the model was done on **Google Colab** through **Transfer Learning.** The reason behind this is the limitation of our personal Systems and also the handy nature of the Transfer Learning as we don't need to install the required libraries and modules one by one. We only needed to setup the separate directories for train, test, validation dataset and also for storing *labels.txt* which contains a lexographic list of the breeds.

2.3 Downloading the Dataset

The Dataset we have used consists of more than 8000 images of dogs tagged with their breeds. The dataset was downloaded from cloud storage of kaggle.com.

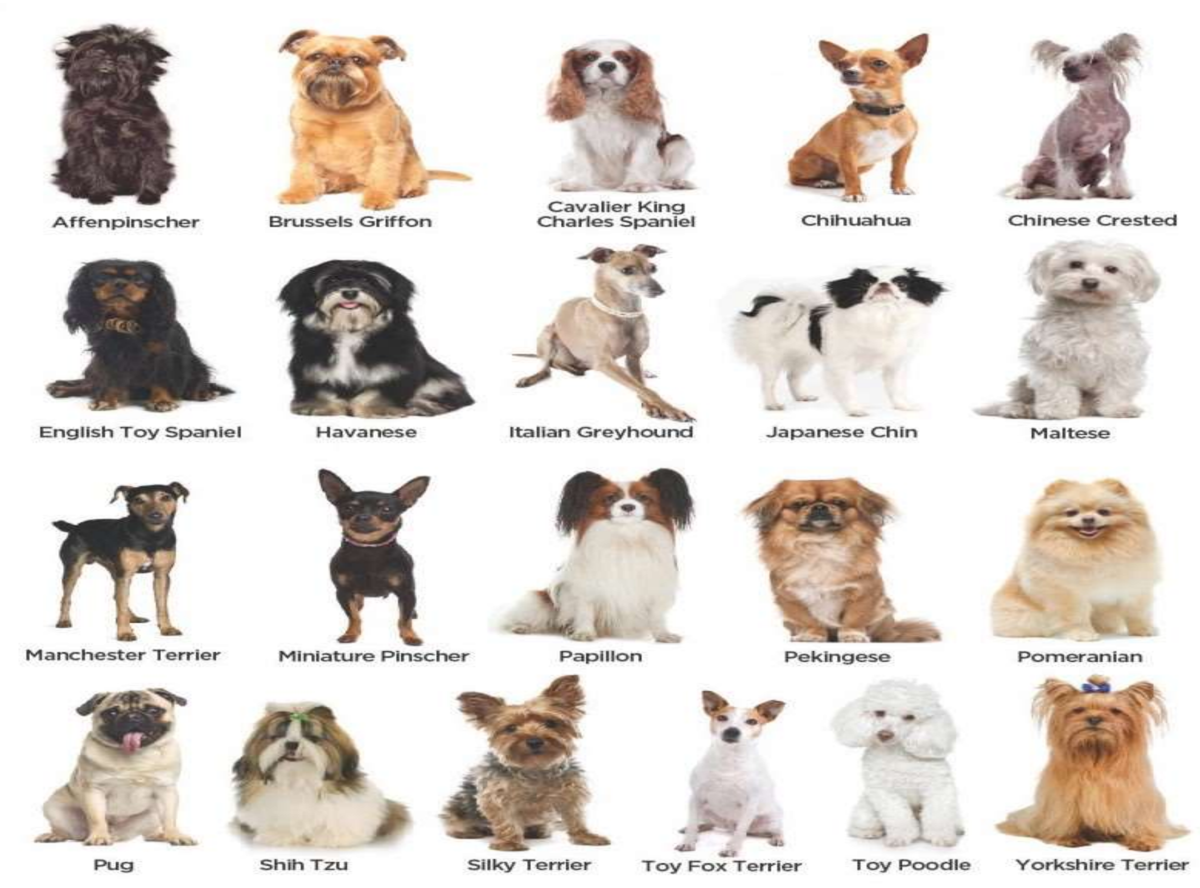


Fig 2.2 Sample Dataset

2.4 Setting partitions of the dataset for training, testing and validation

We have used the following criteria of partition for different purposes:
Training: 80 % of the total dataset was used for training (6400 images)
Testing: 10 % of the total dataset was used for testing (800 images)
Validation: the rest 10 % was used for validation (800 images)

2.5 Populating the training Data

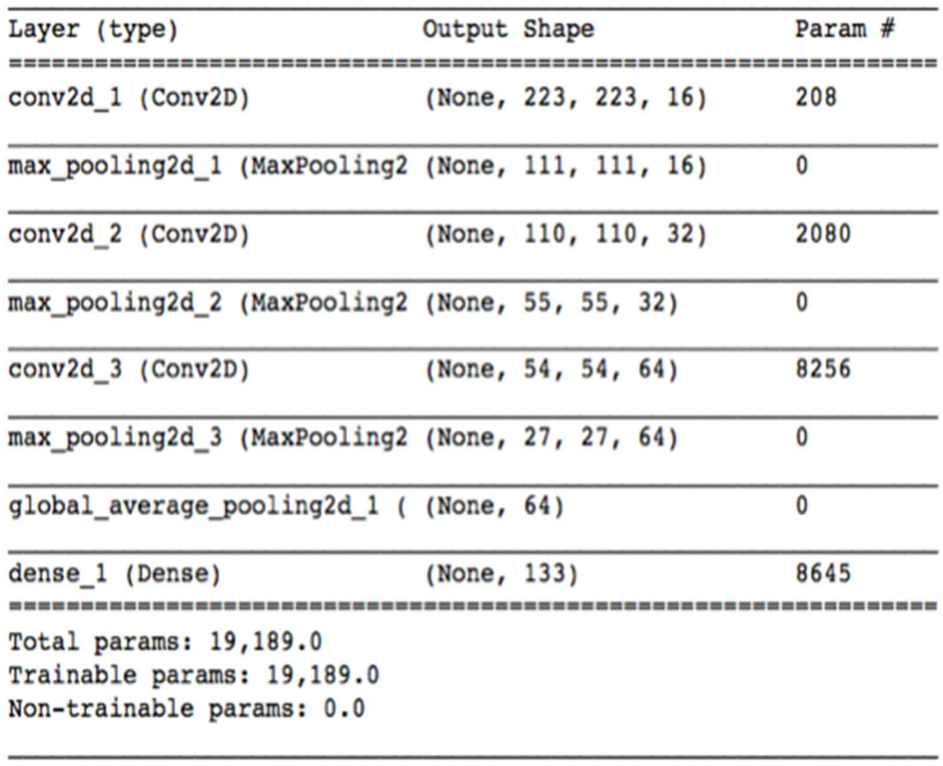
8000 images are not enough for obtaining a decent accuracy of the model. We have to expand the training dataset in order to improve the performance and ability of the model to generalize. Population can be done using **ImageDataGenerator** class of the Keras deep learning library. We can populate the data by Flipping each image vertically and horizontally by changing the brightness of the image.

2.6 Storing the List of Labels in a directory

A file named *labels.txt* stores the list of all breeds in lexographical order.

2.7 Creating a CNN to Classify Dog Breeds

The used CNN architecture has 3 convolutional layers alternating with maxpooling layers, 10% dropout and batch normalization. This is then followed by a global average pooling layer which is then followed by a dense layer (the fully connected layer) to identify 133 breeds. The non-linearity used here is **ReLu** for each convolution and pooling layer and **SoftMax** for the fully connected layer. SoftMax maps each output of the last pooling layer to a value less than 1.0, the sum of all mappings is equal to 1.0 as we want to know the probability of each breed.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 223, 223, 16)      208
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 111, 111, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 110, 110, 32)      2080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 55, 55, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 54, 54, 64)        8256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 27, 27, 64)        0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 133)               8645
=================================================================
Total params: 19,189.0
Trainable params: 19,189.0
Non-trainable params: 0.0
_________________________________________________________________
```

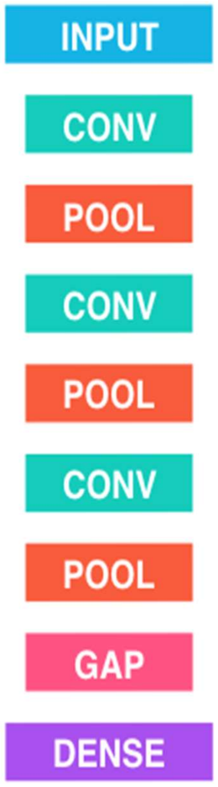


Fig 2.3 Dimension of input, convolution, pooling and output layers

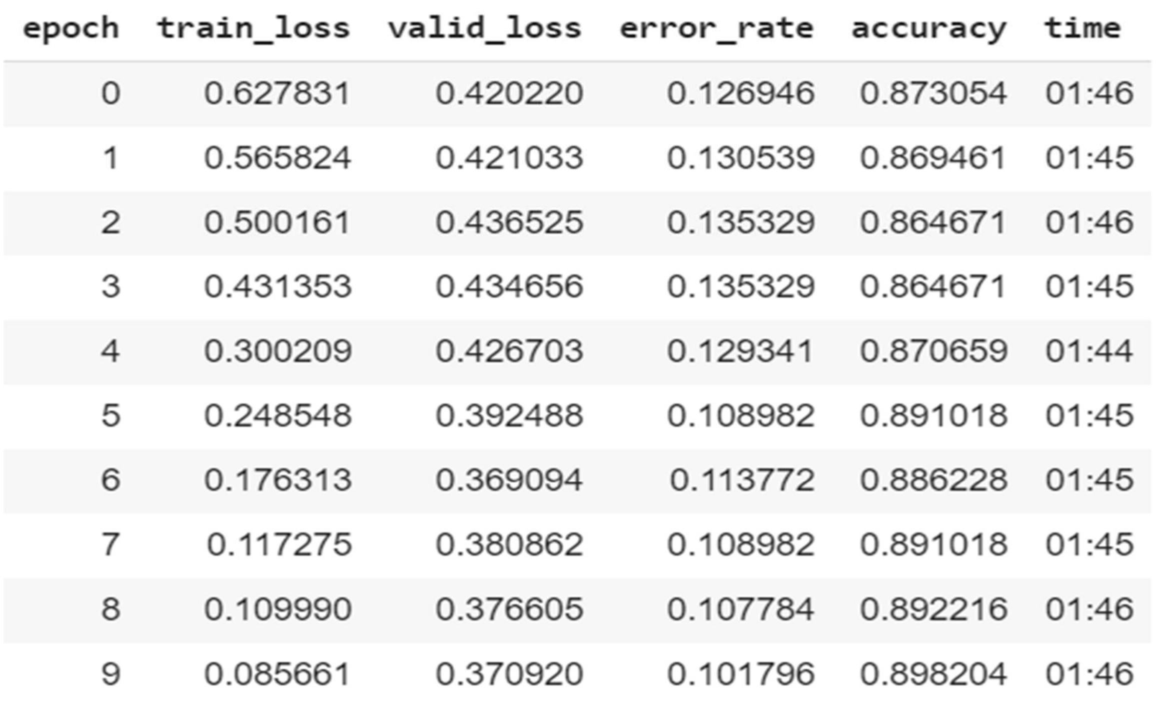
2.8 Training, testing and validation

We need to compile the model using a suitable Loss function and optimizer function. In this project we have used the *rmsprop* optimizer and the *categorical_crossentropy* loss as per suggestions of Abhinav sir.

```
model.compile(loss='categorical_crossentropy', optimizer='rmsprop')

The training was done on the batches of 32
```

Below is a snapshot of training and validation parameters:



epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.627831	0.420220	0.126946	0.873054	01:46
1	0.565824	0.421033	0.130539	0.869461	01:45
2	0.500161	0.436525	0.135329	0.864671	01:46
3	0.431353	0.434656	0.135329	0.864671	01:45
4	0.300209	0.426703	0.129341	0.870659	01:44
5	0.248548	0.392488	0.108982	0.891018	01:45
6	0.176313	0.369094	0.113772	0.886228	01:45
7	0.117275	0.380862	0.108982	0.891018	01:45
8	0.109990	0.376605	0.107784	0.892216	01:46
9	0.085661	0.370920	0.101796	0.898204	01:46

Fig 2.4 Training Metrics

The final accuracy after adjusting the Hyperparameters was close to 93%.

2.9 Saving and Downloading the final model

Finally, we save the best weights obtained by training testing and validation in *model.hdf5* format. But this file cannot be used directly in android studio. We need to convert this file into *model.tflite* format so that TensorflowLite module of android can interpret it. In the next chapter we will use this file to design the android app.

Chapter 3

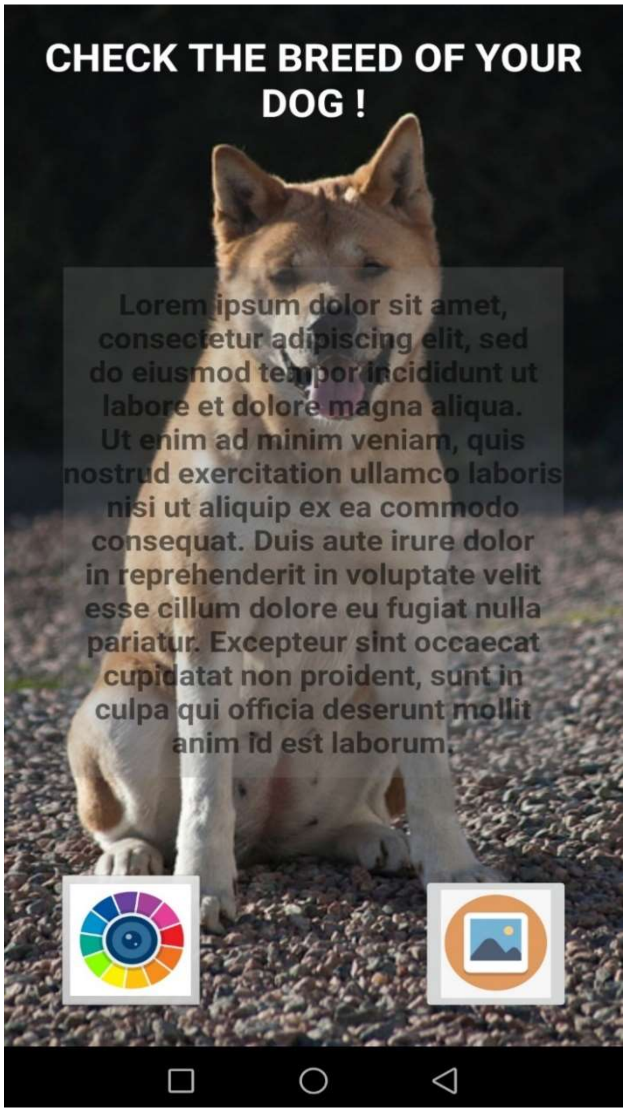
App Development using Android Studio

3.1 Designing the basic structure of the app

We will design a simple two *activity* app: The welcome/main *activity* and the Results activity. The main activity will have a *TextView* containing the Title and short manual of the app. There will be Two buttons: one for camera and another for gallery. The camera button will take you to the camera view where you can see the breed of your dog without clicking the picture. The Gallery button will take you to the File manager where you can select the preclicked image of your dog. On the selection of image, you will be taken to the Results page where a *TextView* will show the results.

3.2 Adding the activities and buttons accordingly

We Create an empty activity and name it *MainActivity* . On the bottom of this activity we add two buttons one for camera and one for gallery .On the Top the activity A text View will show the title of the app and In the centre part of the activity we have a TextView containing info about the app. Below is a snapshot the main activity.



CHECK THE BREED OF YOUR DOG !

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Fig 2.4 Main activity

3.3 Setting up the Assets folder

In the *res* folder of the project we create an *Assets* Folder in which we keep the *model.tflite* and *labels.txt* file downloaded earlier.

3.4 Asking permissions for access of camera and file manager

We need to ask for the permission to access camera and file manager from the user. For this we need to add a few lines of code in *AndroidManifest.xml* file of the app.

```
<uses-permission-sdk-23 android:name="android.permission.CAMERA"></uses-
permission-sdk-23>
<uses-feature android:name="android.hardware.camera" />
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"></uses-permission>
```

3.5 TensorFlow interpreter interprets the input output configuration of the model

Now comes the use of TensorFlow Lite module of android to interpret the model. The interpreter returns the input and output dimensions of the model. In our project input shape is 224*224*3 and output shape is 133*1.

3.6 Converting the input into required format/pre-processing of image

The input image can be of any size depending upon the device configurations, so we need to convert the input image to 224*224*3 byte array. The image taken from camera is of *bitmap* object type and that from gallery is of *Uri* object type. We can do the conversions easily by the help of Converters present in android.

3.7 Run the interpreter and show the results.

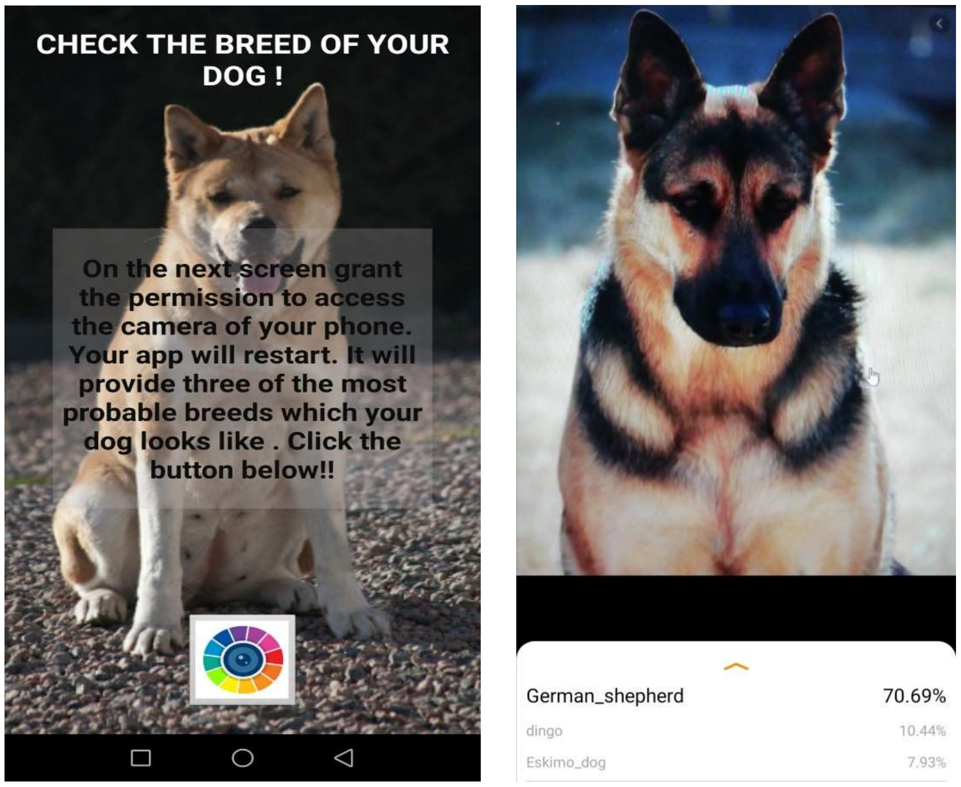
Finally, we have to pass the byte array to the TensorFlow interpreter. The interpret simply calculates the SoftMax values using the saved model (*model.tflite*) and returns the final probability distribution in a vector of shape 133*1. We have to define a function that sorts this array and find the index of the top three probabilities. Using these indices, we can refer to the names of the breeds listed in *labels.txt*. Now the app is ready to run.

Chapter 4

Results and Conclusions

1. The accuracy of the model which we have trained is 93%.
2. The size of the dataset which we use to train our model plays a major role in deciding the accuracy of the model. Larger the dataset, more experienced the model and hence more the accuracy.
3. Activation function such as ReLu function, plays an important role here as it adds the non-linearity to our model. The SoftMax function helps provide our predicted result on the scale of 0 to 1.
4. We can build more complex models and make our machines learn to identify anything and also tell the detail about it.

Here are some snapshots of the working app:



CHECK THE BREED OF YOUR DOG !

On the next screen grant the permission to access the camera of your phone. Your app will restart. It will provide three of the most probable breeds which your dog looks like . Click the button below!!

```
German_shepherd        70.69%
dingo                  10.44%
Eskimo_dog              7.93%
```

Fig 4.1 Snapshots of the app

Bibliography

Figures:

Fig 2.1 Basic Design of the model used in the project
https://blog.floydhub.com/building-your-first-convnet/

Fig 2.2 Sample Dataset
https://medium.com/nanonets/how-to-easily-build-a-dog-breed-image-classification-model-2fd214419cde

Learning material:
https://www.tensorflow.org/lite
https://kaggle.com/
https://towardsdatascience.com/
https://www.coursera.org/