Running Neural Networks On Android Devices

Embedded Systems Course Project

~Bikkavolu Prasanthi (B20CS010),Natha Pavan Sandeep (B20CS036)

INTRODUCTION

Over the last years, the computational power of mobile devices such as smartphones and tablets has grown dramatically,reaching the level of desktop computers available not long ago. While standard smartphone apps are no longer a problem for them, there is still a group of tasks that can easily challenge even high-end devices, namely running artificial intelligence algorithms.In this project,we implemented neural networks on android smartphones using tensorflow lite and other libraries through an API,keeping in mind the hardware specifications of the device.

METHODOLOGY

Dataset

In order to make an Image Classification model, we used the CIFAR10 dataset.
This dataset has 50000 train images and 10000 test images.The images are of airplane, ship, dog, frog, cat, horse, deer, automobile, bird, truck. Thus there are 10 classes in this dataset.

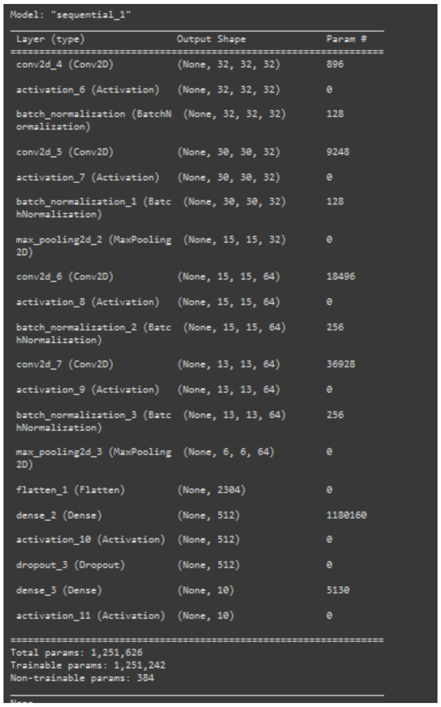
Overview

The model using Neural Networks with the help of tensorflow library and several others is made using Google Colab. The model is then converted into TFlite format so that we can use it in the Android Studio in order to make the API. Then we load the model using tensorflow into the Project and perform several functions in order to use the camera and gallery. All these are done through respective UI design components.

Neural Networks

Model1 ,Model2

Several Conv2D and Max Pooling 2D layers are used as shown in the figure:



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 32, 32, 32)        896

 activation_6 (Activation)   (None, 32, 32, 32)        0

 batch_normalization (BatchN  (None, 32, 32, 32)       128
 ormalization)

 conv2d_5 (Conv2D)           (None, 30, 30, 32)        9248

 activation_7 (Activation)   (None, 30, 30, 32)        0

 batch_normalization_1 (Batc  (None, 30, 30, 32)       128
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_6 (Conv2D)           (None, 15, 15, 64)        18496

 activation_8 (Activation)   (None, 15, 15, 64)        0

 batch_normalization_2 (Batc  (None, 15, 15, 64)       256
 hNormalization)

 conv2d_7 (Conv2D)           (None, 13, 13, 64)        36928

 activation_9 (Activation)   (None, 13, 13, 64)        0

 batch_normalization_3 (Batc  (None, 13, 13, 64)       256
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten_1 (Flatten)         (None, 2304)              0

 dense_2 (Dense)             (None, 512)               1180160

 activation_10 (Activation)  (None, 512)               0

 dropout_3 (Dropout)         (None, 512)               0

 dense_3 (Dense)             (None, 10)                5130

 activation_11 (Activation)  (None, 10)                0

=================================================================
Total params: 1,251,626
Trainable params: 1,251,242
Non-trainable params: 384
_________________________________________________________________
```

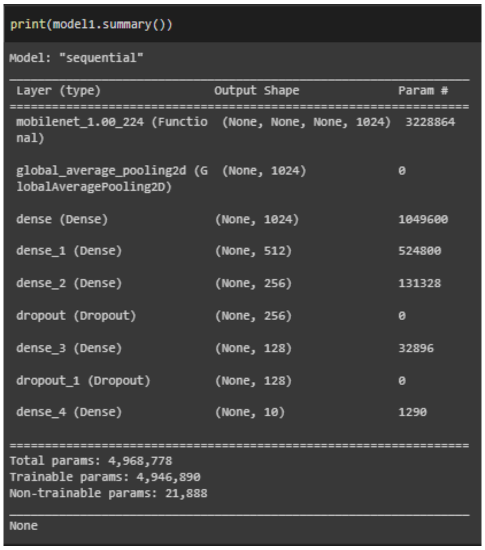
Mobilenet

It is a CPU based model.It is a lightweight and efficient model designed for mobile devices with limited computational resources. It uses depth wise separable convolutions to reduce the number of parameters and computational requirements.

The first layer is a pre-trained MobileNet model with a 1.0 width multiplier and an input size of 224x224. This layer has 3,228,864 parameters.

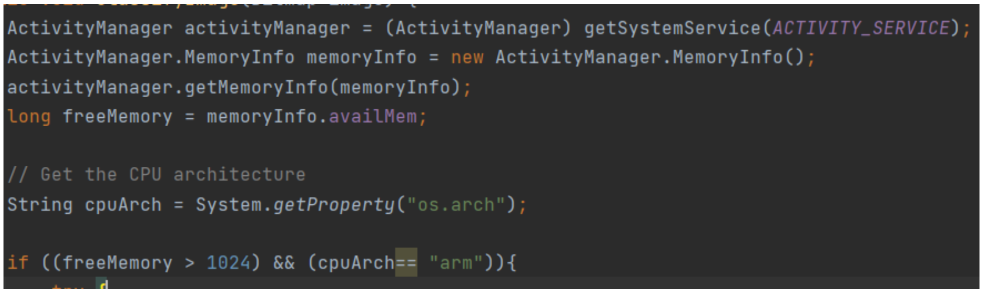
The second layer is a Global Average Pooling 2D layer that averages the values of each feature map in the previous layer, resulting in a fixed-length vector of 1024 values.

The next five layers are Dense layers, which are fully connected layers that take the fixed-length vector as input and output another vector of different size.



```
print(model1.summary())

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenet_1.00_224 (Functio  (None, None, None, 1024)  3228864
 nal)

 global_average_pooling2d (G  (None, 1024)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 1024)              1049600

 dense_1 (Dense)             (None, 512)               524800

 dense_2 (Dense)             (None, 256)               131328

 dropout (Dropout)           (None, 256)               0

 dense_3 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_4 (Dense)             (None, 10)                1290

=================================================================
Total params: 4,968,778
Trainable params: 4,946,890
Non-trainable params: 21,888
_________________________________________________________________
None
```

Model Switching



```
ActivityManager activityManager = (ActivityManager) getSystemService(ACTIVITY_SERVICE);
ActivityManager.MemoryInfo memoryInfo = new ActivityManager.MemoryInfo();
activityManager.getMemoryInfo(memoryInfo);
long freeMemory = memoryInfo.availMem;

// Get the CPU architecture
String cpuArch = System.getProperty("os.arch");

if ((freeMemory > 1024) && (cpuArch== "arm")){
```

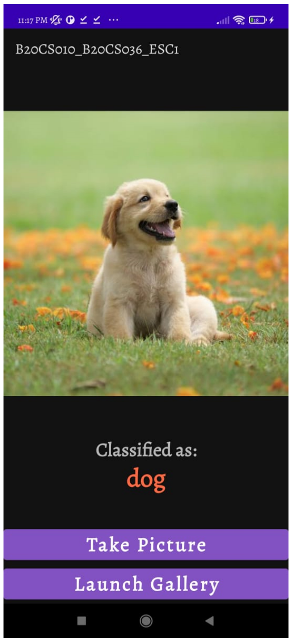
Based on the available free memory, CPU available, model is selected and implemented.

API

We’ve used Android Studio to create this API .

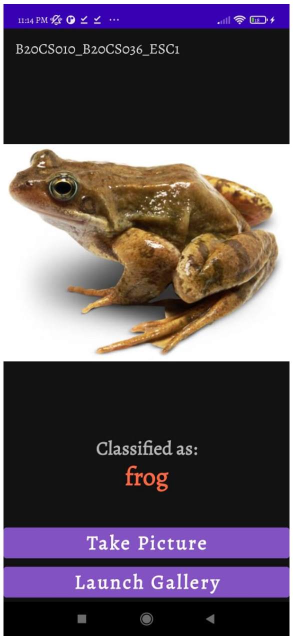
RESULTS

The obtained results after giving images as input:



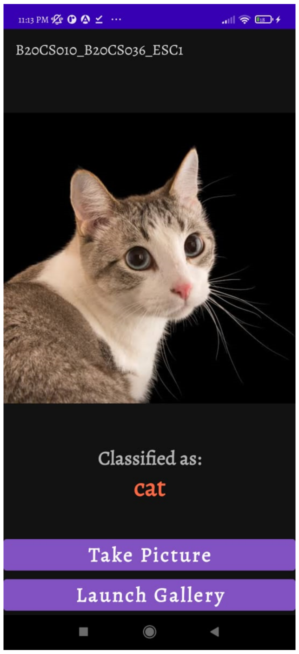
B20CS010_B20CS036_ESC1

Classified as:
dog



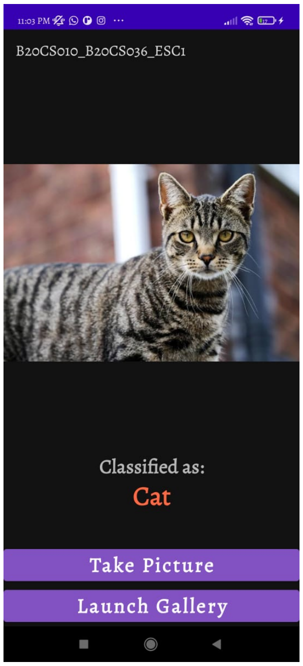
B20CS010_B20CS036_ESC1

Classified as:
frog



B20CS010_B20CS036_ESC1

Classified as:
cat



B20CS010_B20CS036_ESC1

Classified as:
Cat

REFERENCES

- *AI Benchmark*
- *Neural Networks API*
- *Fruit Image Classification*