iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** VI **Month of publication:** June 2021

DOI: https://doi.org/10.22214/ijraset.2021.35091

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: 7.429
Volume 9 Issue VI Jun 2021- Available at www.ijraset.com

Blindfold: A Smartphone based Object Detection Application

Rishabh Sharma[1], Sampada Lamba[2], Khushi Maheshwari[3], Parul Yadav[4]
[1, 2, 3, 4]*Information Technology,Bharati Vidyapeeth College Of Engineering*

Abstract: ***With the advancement of computing power of Smartphones, they seem to be a better option to be used as an Assistive Technology for the visually impaired. In this paper we have discussed an application which allows visually impaired users to detect objects of their choice in their environment. We have made use of the Tensorflow Lite Application Programmable Interface (API), an API by Tensorflow which specifically runs models on an Android Smartphone. We have discussed the architecture of the API and the application itself. We have discussed the performance of various types of models such as MobileNet, ResNet & Inception. We have compared the results of the various Models on their size, accuracy & inference time(ms) and found that the MobileNet has the best performance. We have also explained the working of our application in detail.***

Keywords: Visually Impaired, Object Detection, TensorFlow Lite, SSD, MobileNet, Android, Smartphone Based.

I. INTRODUCTION

Unlike a normal sighted person, the visually impaired face a lot of problems in their day to day life with locating objects being a common one. A visually impaired person has sharper senses than a regular human being. When they are in a familiar environment they know where objects of their use are kept or the general direction. When they are in an unfamiliar environment they have difficulty locating objects. Our aim was to create an application that provides a general sense of direction to the user so that they can find the object they want easily.

Recent research patterns[1] show a rise in development of mobile application aids rather than wearable and handheld devices aids for the visually impaired. Having a single device perform multiple aid operations is prefered over having separate external devices and it is more cost effective. Increased availability and higher computation capabilities of smartphones gives motivation to develop various applications with increased performance to assist the visually impaired in their day to day life.

On comparing various studies and reports on Assistive Technologies [2] we concluded that Sensor based AT were more efficient but were much costlier than Deep Learning AT. Assistive Technology (AT) can be any device or software that reduces the effort of the visually impaired people in understanding their surroundings and makes them more capable, independent and self-sufficient. Assistive technology improves their response to a situation by providing them information.

Our application is based on the Tensorflow Lite API [3] which is capable of running Deep Learning models trained by Tensorflow [4]. Some of these models may run differently from their counterpart as Size and Accuracy changes on an Android Smartphone we have discussed that as well in our results. There are different types of models such as:

1) Quantized Models[5]: Quantized object detection models provide the tiniest model size and quickest performance, at the expense of accuracy.
2) Floating purpose Models[6]: Floating point models offer the most effective accuracy, at the expense of model size and performance. GPU acceleration needs the utilization of floating point models.
3) AutoML Models[7]: AutoML refers to the Cloud AutoML by Google.

II. BACKGROUND.

A. TensorFlow

A normally used package for metric capacity unit tasks is Tensorflow. It is the most commonly used because it provides a precise interface and simple code that can be implemented by beginners . Models created in TensorFlow can be transferred onto and implemented on multiple devices with very little or no changes to the code. Such devices range from android devices to distributed servers.

By adding these identity mappings shortcuts connections are created between layers, inherently creating the network easier to train. The model used throughout this paper is trained on machine servers and then shifted to androids and therefore tensorflow is acceptable. The desired latency for mobile applications is low.

B. TensorFlow Mobile

During design, Tensorflow was developed to be ready to run on heterogeneous systems, as well as mobile devices. Tensorflow Mobile[8] permits developers to store and execute the cubic centimetre models on the device itself, saving surplus computation and communication with the server. As cubic centimetre tasks are resource consuming, models are modified to boost performance. The minimum hardware requirements of TensorFlow Mobile are low,most android devices can run TensorFlow Mobile models. The primary issue which delays the result is the speed of the computations because the desired delay or lag for mobile applications is low.

C. TensorFlow Lite

Tensorflow Lite is the next big step for TensorFlow Mobile, which already provides support preparation on mobile and other devices. The new trend requires the functionality of an application to be available on mobile phones. While using these applications on a mobile devices a user expects the mobile applications to be efficient and quick in terms of camera and voice usability. It is extremely important to optimise TensorFlow Mobile to be efficient on mobile devices. Many of the improvements administered in TensorFlow Lite are hardware accelerations, frameworks similar to the automaton Neural Network[9] API and android-optimised ANN's[10]. TensorFlow -trained models are converted to TensorFlow Lite model format by TensorFlow.

D. SSD

SSD[11] makes use of a set of default bounding bins and the usage of convolutional filters carried out to characteristic maps. Single Shot detectors like YOLO[12] take only one shot to detect multiple objects present in a frame by implementing a multibox. It is significantly faster and has a high precision object detection algorithm as compared to other models.
SSD has two components: a backbone model and SSD head.

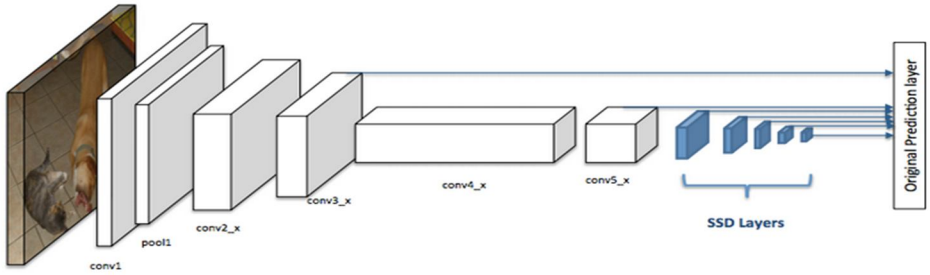


Fig. 1 Architecture of a convolutional neural network with a SSD detector

SSD splits the image using a grid and each of the grid cells is responsible for detecting and recognizing objects in that area of the image. Detecting objects refers to prediction of the class and position of an object within an area.

E. Mobile Net

MobileNet[13] is established on a modernized architecture that uses depth-wise separable contortions to build the CNN[14]. These models allow for the optimisation of hyperparameters to adjust the accuracy commutation depending on the conflict.
MobileNet has turned out to be efficient for a wide range of applications like object detection,facial attributes recognition[15], classification, and large scale geolocation. It factorises the standard convolution and converts it into a 1x1 pointwise convolution and depthwise convolution The pointwise convolution is used to combine the output of the depthwise convolution that applies a filter to each input.

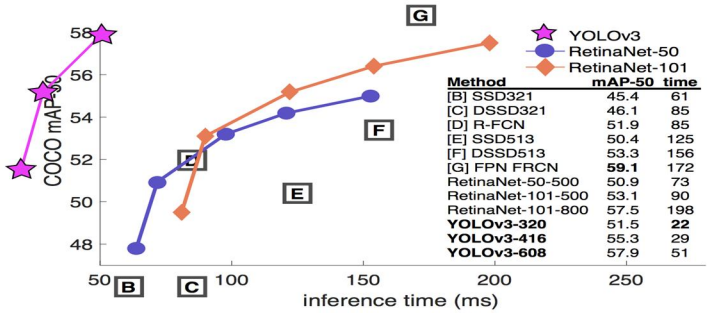
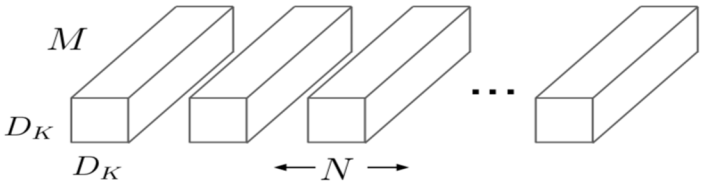
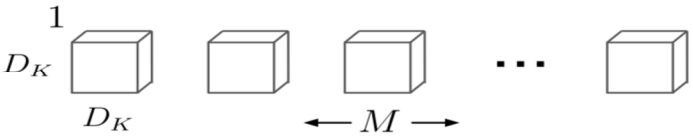


Fig. 2 Example of an image with acceptable resolution

The usual contortion produces new output by merging the input and administering the filters in one step, whereas the output for depthwise convolution is obtained by dividing the layers into a filtering layer and a combining layer, which deducts size and computations. Using 3x3 depthwise convolutions, MobileNet computations are lessened to 1/8 - 1/9 of the usual contortion.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters

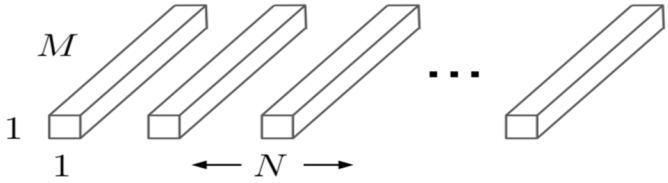


Fig.3 Example of an image with acceptable resolution

MobileNet has proven to be a useful base network for object detection. The results on the COCO dataset are propitious. This architecture gives similar mAP performance as compared to Faster-RCNN[16] and SSD while giving the least computation cost as compared to VGG and Inception.

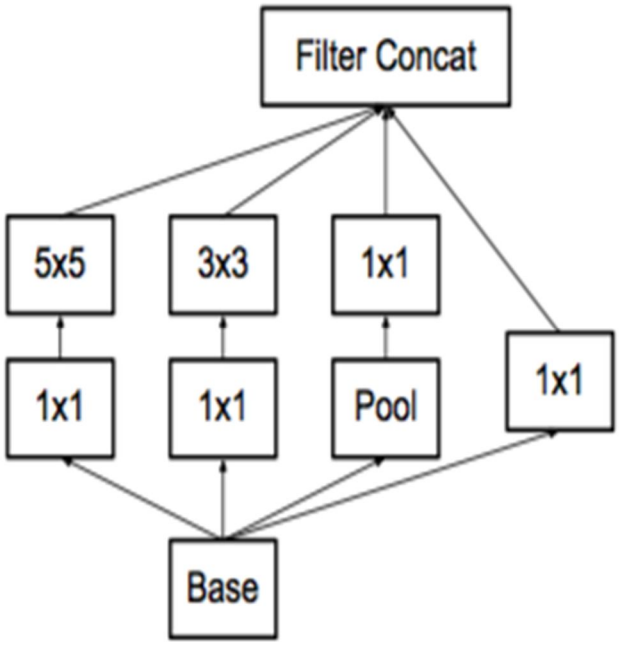


Fig. 4 SSD

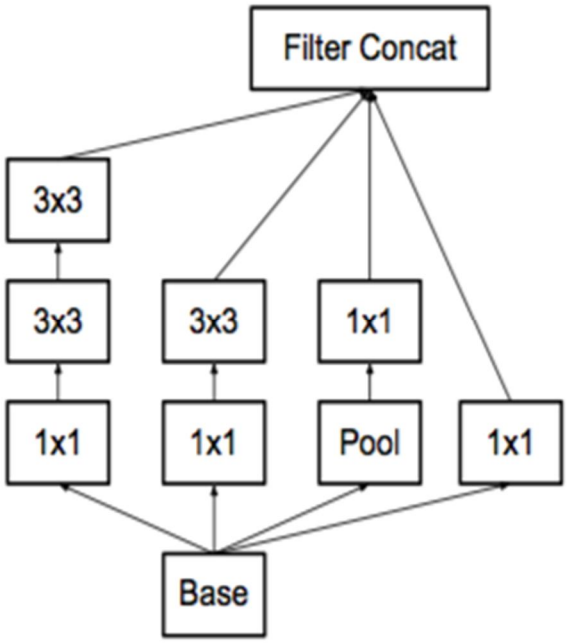


Fig. 5 MobileNet

III. ARCHITECTURE

Architecture of the Tensorflow Lite API[17] is given below:

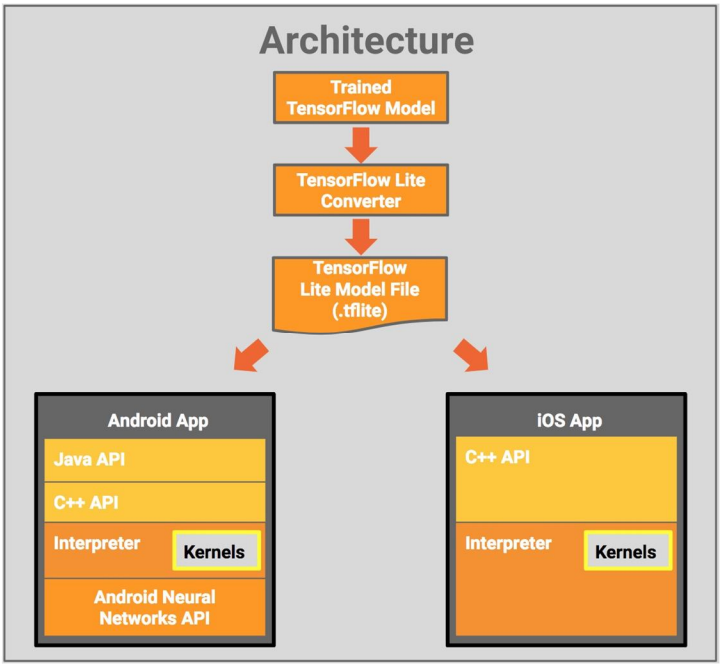


Fig. 6 Architecture of the TensorFlow Lite API

We train a model using the Tensorflow API. We create a frozen graph of the Model.
Which is then converted into the .tflite file format. The output is a .tflite file which is the model itself and a class file which contains all the names of the classes.
The architecture of the Tensorflow Lite Object Detection API is as follows:

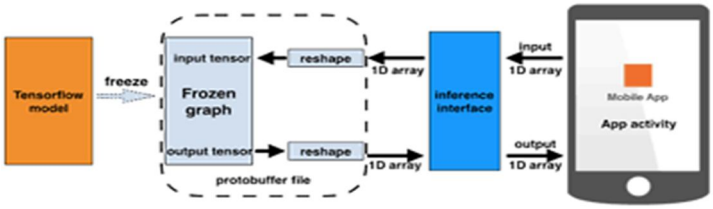


Fig. 7 Architecture of the Tensorflow Lite Object Detection API

The API uses the .tflite file to compare weights and calculate the confidence value which contains the weights. The class file is mapped according to the weights and is used to recognize the object once the confidence value exceeds the threshold requirement.
The most commonly used type of model for the TFlite API is MobileNet. They require low computation power and resources and yield an efficient result.
The technique used is Single-Shot multibox Detection for which the model 'mobilenet-ssd' is used. This model is implemented using the Caffe framework [18].

IV. WORKING

The application uses the Tensorflow Lite Object Detection API as its Skeleton. We have added a new layer to this application which uses the Google Speech Recognition API [19] to take the speech input of the user.
This is an android application that detects objects in the frames as seen by your smartphone's back camera, using a MobileNet SSD [20] model trained using the COCO dataset. A file reader has also been implemented to check if the input of the user exists in the class file of the model or not. If the class file does not contain the input by the user then an Audio message is given to the user that the class file does not contain the input and to try again.

The application only detects the class given as voice input by the user. The Speech Recognition API is local, so it also works in the offline mode.

The classes stored in the class file of the model are accessed by the Buffered File Reader to check whether the user input matches any class in the class file. Hence it is important to limit the size of the class file and the model as that would take more time in loading and checking the existence of the object by the user.

On startup the application screen contains a single button which upon being pressed starts the voice recognition. After it is verified that the input matches an existing class in the labels.txt file the Object Recognition process starts.

Since the Object Detection API of Tensorflow Lite tries to detect all objects in its frame, after adding some constraints of our own, it only detects the object the user wants to detect.

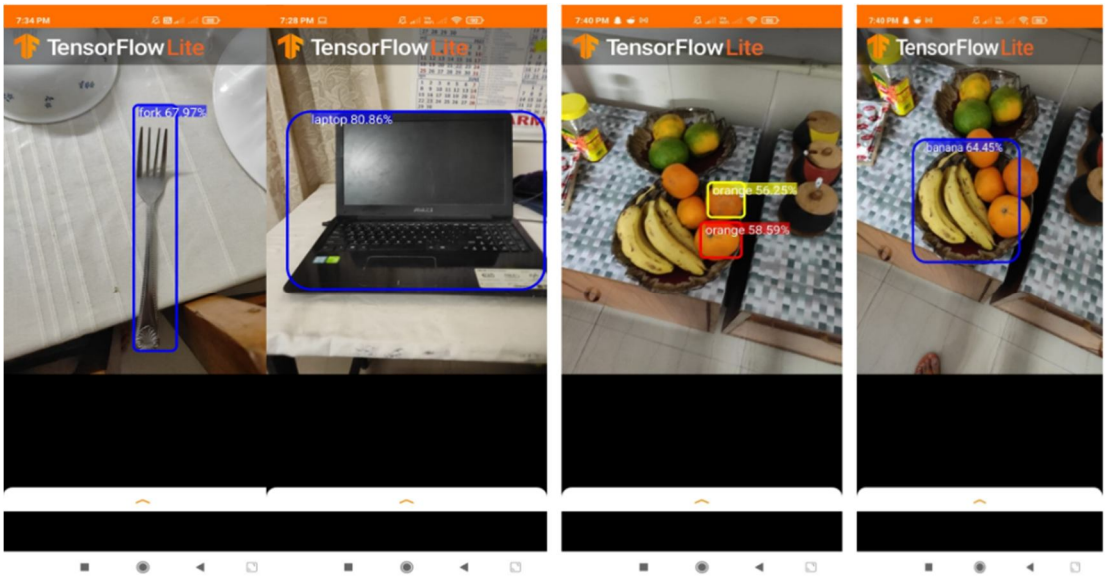


Fig. 8 Real Time working of the BlindFold Application

V. RESULTS

After implementing different types of models such as Quantized Models, Floating Point Models, AutoML Models, etc. The following observations were noticed.

TABLE I
Results Obtained

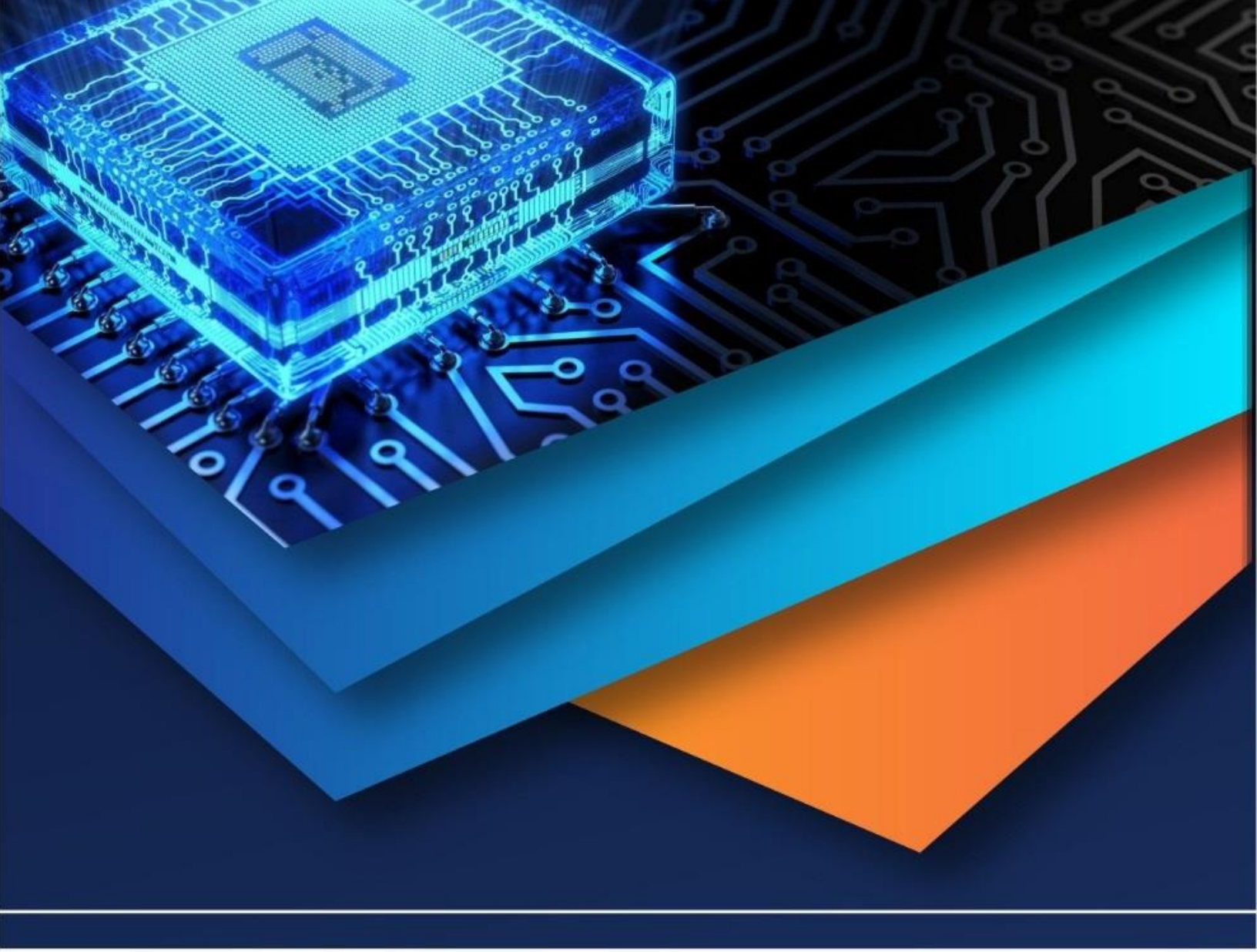
Model name	Model size	Accuracy	CPU, 4 threads	NNAPI
Mobilenet_V1_1.0_192_quant	4.3 Mb	69.1%	9.9ms	5.2ms
Inception_V1_quant	6.4Mb	70.1%	39ms	36ms
ResNet_V2_101	178.3Mb	76.8%	526ms	1572ms

The results of the models may vary on different devices because of the quality of the hardware such as:

1) *Camera:* The application uses the rear camera of the phone and the quality of the lens, sensors may have an impact on the performance of the model.
2) *Central Processing Unit(CPU):* The processing power of the CPU has a big impact on the inference time of the model.
3) *Graphics Processing Unit (GPU):* Models which utilise the GPU run faster than their CPU counterpart[21] because the computing power is a lot higher than the CPU.

REFERENCES

[1] Maheshwari, K., Lamba, S., Sharma, R., & Yadav, P. A Survey on Mobile Applications for the Assistance of the Visually Impaired.(1997).
[2] Bhowmick, Alexy, and Shyamanta M. Hazarika. "An insight into assistive technology for the visually impaired and blind people: state-of-the-art and future trends." Journal on Multimodal User Interfaces 11.2 (2017): 149-172.
[3] David, Robert, et al. "Tensorflow lite micro: Embedded machine learning on tinyml systems." arXiv preprint arXiv:2010.08678 (2020).
[4] Abadi, Martín, et al. "Tensorflow: A system for large-scale machine learning." 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16). 2016.
[5] Jacob, Benoit, et al. "Quantization and training of neural networks for efficient integer-arithmetic-only inference." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.
[6] Yin, Penghang, et al. "Quantization and training of low bit-width convolutional neural networks for object detection." arXiv preprint arXiv:1612.06052 (2016).
[7] He, Xin, Kaiyong Zhao, and Xiaowen Chu. "AutoML: A Survey of the State-of-the-Art." Knowledge-Based Systems 212 (2021): 106622.
[8] Alsing, Oscar. "Mobile object detection using tensorflow lite and transfer learning." (2018).
[9] Pishnamazi, Mahboubeh, et al. "Application of lignin in controlled release: development of predictive model based on artificial neural network for API release." Cellulose 26.10 (2019): 6165-6178.
[10] Agatonovic-Kustrin, S., and Rosemary Beresford. "Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research." Journal of pharmaceutical and biomedical analysis 22.5 (2000): 717-727. .
[11] Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.
[12] Aralikatti, Anish, et al. "Real-time object detection and face recognition system to assist the visually impaired." Journal of Physics: Conference Series. Vol. 1706. No. 1. IOP Publishing, 2020.
[13] Li, Yiting, et al. "Research on a surface defect detection algorithm based on MobileNet-SSD." Applied Sciences 8.9 (2018): 1678.
[14] Shin, Hoo-Chang, et al. "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning." IEEE transactions on medical imaging 35.5 (2016): 1285-1298.
[15] Xia, Xiao-Ling, Cui Xu, and Bing Nan. "Facial expression recognition based on tensorflow platform." ITM Web of Conferences. Vol. 12. EDP Sciences, 2017
[16] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." arXiv preprint arXiv:1506.01497 (2015).
[17] Lee, Juhyun, et al. "On-Device Augmented Reality with Mobile GPUs."
[18] Jia, Yangqing, et al. "Caffe: Convolutional architecture for fast feature embedding." Proceedings of the 22nd ACM international conference on Multimedia. 2014.
[19] Këpuska, Veton, and Gamal Bohouta. "Comparing speech recognition systems (Microsoft API, Google API and CMU Sphinx)." Int. J. Eng. Res. Appl 7.03 (2017): 20-24
[20] Yadav, Nikhil, and Utkarsh Binay. "Comparative study of object detection algorithms." International Research Journal of Engineering and Technology (IRJET) 4.11 (2017): 586-591.
[21] Lawrence, John, et al. "Comparing TensorFlow deep learning performance using CPUs, GPUs, local PCs and cloud." (2017).



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY