International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-9, Issue-5, September 2021
https://doi.org/10.21276/ijircst.2021.9.5.6
Article ID IRPD1153, Pages 38-40
www.ijircst.org

Real-Time Image Processing using Flutter and Tflite Packages

Akash Yadav, Deepanshu Thakran, and Dr. Rashmi Gupta

Student, Department of Computer Science & Engineering, ASET, Amity University Haryana, Gurugram, India
Student, Department of Computer Science & Engineering, ASET, Amity University Haryana, Gurugram, India
Assistant Professor, Department of Computer Science & Engineering ASET, Amity University Haryana, Gurugram, India

Correspondence should be addressed to Akash Yadav; akash21091999@gmail.com

ABSTRACT-In today's world, the application available in the field of real time image processing are slow, provide results about a limited set of objects in a frame, and is not easily usable by people of all age groups. The nature of visual based knowledge is far better than anything else, if a person wishes to know about an object, or to remember an object, the best way is to first see how the object looks, and vice-versa, if you know how the object looks and feels, and you know what it is called, it would be best describable. Applications like Google lens are already prevalent in the market, but only available for android, which leaves rest of the OS users in vain, creating a market space for the untouched. Features like pose estimation can be inculcated through different software to yield AI based solutions to serve different purposes. Object detection and image classification can not only provide you information about the object, but in today's connected world, can lead you to different sources of knowledge for the same, be it a link to buy the object, or a thesis about it or some facts about it.

KEYWORDS – Object detection, Image classification, Pose estimation, Real-time image processing, Image processing, tflite, tensorflow lite, Flutter;

I. INTRODUCTION

There are so many models present for real-time image processing for devices with high-end GPUs and fast CPUs, but not all of have that. Even running one single model on the low-end devices is not less than a hustle. Now coming to mobile devices such as our smartphones, tablets and SOCs (eg. Raspberry Pi), they don't have the most ideal processors for running those models. A computer can run several heavy tasks parallelly in the without any issue, but the same is not possible for most of the smartphones or tablets. There are several models that all of want to experience as a user. These models include image classification, object detection, pose estimation, etc. Image classification extract different important components of the image and provide the information to the user. [1] Objects detection can detect and locate the object present in an image. [2] Pose estimation detects the key points of the human body and projects them on the image. [3] These models are very heavy models. Moreover, for real-time processing, they require much more processing power and memory.

We cannot use these heavy models for the real- worlds devices. They need to be modified and compresses in order to be compatible with these devices. For that, we have used TensorFlow lite. TensorFlow lite module help you to compress large into mobile device compatible packages. The compression is so significant and impressive, if your model is about 800 MB in size, it can be easily compressed to around 1.5 to 2 MB without compromising with the performance. To run those models on mobile devices, we created an app using Flutter SDK provided by Google. It provides user with an easy and flexible framework to work with.

II. METHODS

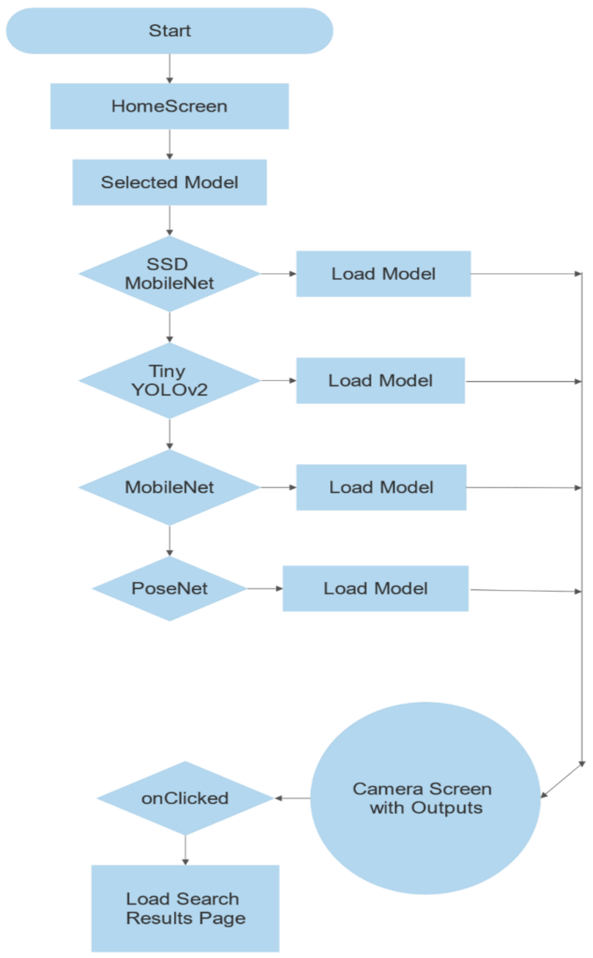

Fig.1: Flow chart of Implementation of the application

This was the working of the entire project, first the app processes the images in real-time, then provide the output in a responsive UI. When the object label is clicked, it sends get request to the Google Custom Search API for the search results of the object, then provides with the list of items from the search results.

Now know about the development and compression of the models.

Here are the basic steps of training and testing a model:

Step 1. Import Packages And Libraries
Step 2. Set Train And Test Directories
Step 3. Define Label, Train And Functions
Step 4. Train Data
Step 5. Create Matrix To Store Pixels And Label Of The Images
Step 6. Train Test Split Test On The Basis Of Trained Data

Now The Main Part, Compressing The Models Using Tensorflow Lite Module:

Step 1. After The Model Is Trained, Convert It Into A Compressed Flat Buffer Using The Tflite Converter.
Step 2. Take The Compresses Flat Buffer File (.Tflite) And Add It To The Assets Folder Of Your App
Step 3. Quanitze By Converting 32-Bit Floats To More Efficient 8-Bit Integers

III. RESULTS

After the successful implementation of the model, the results are generated as shown in figure 2. We can clearly see that the captions generated for the images are pretty much relevant and are close to how a human being will define these images. The app provides the user with real time outputs, all the models work in real time and with Flutter's native installation, there is no performance difference to be found out between this app and other native apps. On a testing performed over 100 objects the amount of objects that were predicted accurate were inculcated in the accuracy

Here are some observations on the accuracy of the models:

- SSD MobileNet: 64.4 to 89.9%
  The score is a number between 0 and 1 that indicates confidence that the object was genuinely detected. The closer the number is to 1, the more confident the model is and the model was successfully able to recognize 899 objects if there is only 1 object in the frame, and 644 objects if there were more multiple objects in the frame. The TensorFlow Lite quantized MobileNet model's accuracy ranges from 30% to 92% in confidence to the object visible in the frame. The size of a model on-disk varies with its performance and accuracy.
- Tiny YOLOv2
  YOLO v2 Tiny is a real-time object detection model implemented with Keras* from this repository and converted to TensorFlow* framework. This model was pretrained on COCO* dataset with 80 classes.
  Accuracy metrics obtained on COCO* validation dataset for converted model.

METRIC	VALUE
mAP	27.34%
COCO* mAP	29.11%

- MobileNet: 64.3%
  Accuracy is measured in terms of how often the model correctly classifies an image. A model with a stated accuracy of 60% can be expected to classify an image correctly an average of 60% of the time. On a dataset of 1000 objects the module was able to classify 643 images.
- PoseNet
  The pose estimation models takes a processed camera image as the input and outputs information about keypoints. The keypoints detected are indexed by a part ID which indicates the probability that a keypoint exists or may exist in that position even if we have an incomplete image of a person inside the frame.

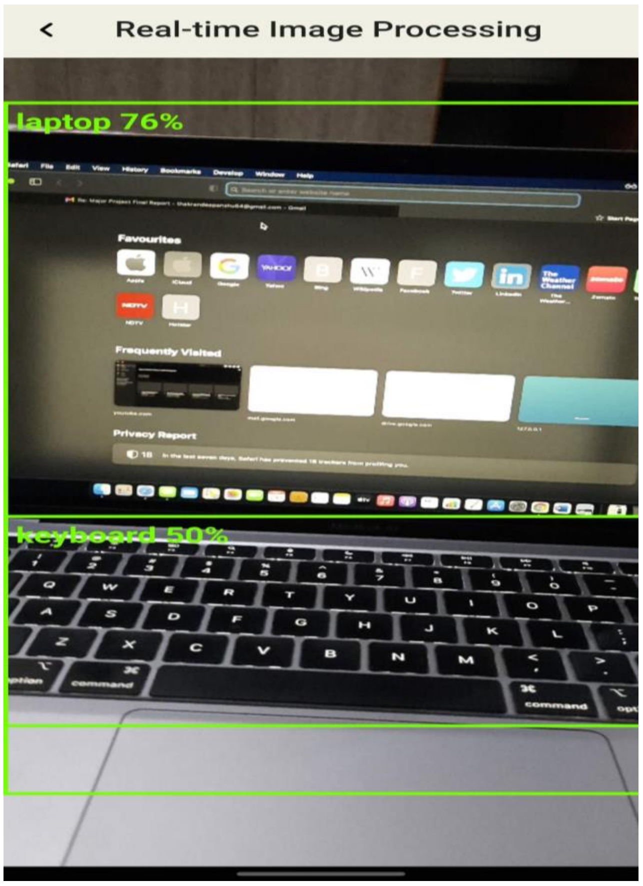

Fig. 2: Real time usage of the application detecting a laptop and a keyboard with 76% and 50% accuracy respectively

IV. DISCUSSION

The app was evaluated on many different properties, be it performance, speed, accuracy or user interface. We also conducted a survey to review this app among a small group on people. Here are some of the results we obtained from the survey.

V. CONCLUSION

This app will allow the user to explore the world with a new perspective. Instead of just looking, they can also learn about different objects and that too in real-time. The models have scope to improve and will be improved further. The user feedbacks will not be unheard and they will get what they need. This product has so much real world application that it can easily attract a wide number of users.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

ACKNOWLEDGMENT

Acknowledgment is not a mere obligation but the epitome of humility and ineptness to all those who have helped in the completion of this project. We are thankful to Dr. Rashmi Gupta, Department of CSE, ASET for their constant guidance and encouragement provided in this endeavor. I also thank my parents for their continued support, understanding and patience without whose support and understanding this endeavor would never be fruitful. We also thank all our friends for helping us out in completing this project and helping me in solving various problems encountered during the progress of this project.

REFERENCES

[1] Wang, M. X. (2019, February 11). Research on image classification model based on deep convolution neural network | EURASIP Journal on Image and Video Processing | Full text. Retrieved from springeropen.com: https://jivp-eurasipjournals.springeropen.com/arti cles/10.1186/s13640-019-0417-8
[2] Zhao, Z.-Q., Zheng, P., Xu, S.-T., & Wu, X. (2019, January 28). Object Detection With Deep Learning: A Review | IEEE Journals & Magazine | IEEE Xplore. Retrieved from ieeexplore.ieee.org: https://ieeexplore.ieee.org/document /8627998
[3] Raj, B., & Osin, Y. (2019, June). An Overview of Human Pose Estimation with Deep Learning | KDnuggets. Retrieved from KDnuggets.com: https://www.kdnuggets.com/2019/06 /human-pose-estimation-deep- learning.html
[4] https://www.tensorflow.org/lite
[5] https://flutter.dev/

ABOUT THE AUTHORS

Akash Yadav, Final year student of Bachelor in Technology in Department of Computer Science & Engineering, ASET, Amity University Haryana, Gurugram, India.

Deepanshu Thakran, Final year student of Bachelor in Technology in Department of Computer Science & Engineering, ASET, Amity University Haryana, Gurugram, India.

Dr. Rashmi Gupta, Assistant Professor, CSE, Amity School of Engineering and Technology, Amity University Haryana. She has 9.5 plus years of teaching and research experience and has published more than 20 papers in various international journals as well as in conferences of good repute. Her research area is Artificial intelligence, Software Engineering, and algorithms.