Assistive Technology: Object Detection

Srinivas Shashank Mulugu Monica Selvam Anh The Nguyen
Nikhillesh Sadassivam
Arizona State University
{smulugu, mselvam, atnguy37,nsadassi}@asu.edu

Abstract

Object detection applications have influenced a large portion of automation in the current era of evolving technology. Image detection and classification in mobile applications have become more prominent over the years and have aided people in a lot of ways, eg. google lens, Camscanner etc. In this paper, we built an object detection application for mobile devices that would aid the visually challenged people by detecting the objects that are situated closest to the user and alerting them about their position and distance of the object from where they are present. Specifically, we try to identify a set of 3 objects (People, Chairs and Cars) in the environment that the smartphone camera detects and identifies, we also estimate the distance of the object from the user's camera and also infer the position of the object from the center of the screen. Given this information, we can alert the person of the presence of such objects next to them, thus enabling the person to be more aware of their surroundings.

1. Introduction

Having accurate recognition of images should be given importance as we try to interpret images better. The development in Machine Learning models have given rise to advancement in object detection. In this paper, we not only try to predict the object's class but also evaluate the distance and relative position of the object from the user. Restricted computational power has always been a challenge when implementing high-level algorithms in a smartphone hence it is necessary that operations are optimized for a smartphone. We have used Tensorflow Lite (TFL) which is a mobile framework to identify three objects namely chairs, people and cars. The user's mobile phone's camera is being used to recognize these objects. For optimization purpose, we are able to use pre-trained models on-device to get better results [3]. By using the COCO dataset, we are able to identify our desired objects for classification. This paper will aid the visually challenged in their everyday lives. To identify the relative position, the bounding box plays an important role. We detect five relative positions – Left, Right, Extreme Left, Extreme Right and Center. To calculate distance measurement of the object from the user, we calculate the focal length and apply triangular similarity. Finally, The Text to Speech feature was implemented to help the visually challenged gain more information about their surroundings as it notifies the user about the object through speech.

2. Related Work

Object detection in mobile devices has started to gain demand as the necessity for object detection applications such as autonomous driving where object detection needs to be done fast and also is restricted to only a certain amount of computational resources. Early works have been done by Zhongjie Li and Rao Zhang [8] where they develop a CNN based object detection algorithm on android devices and other works for object detection models developed such as the MobileNet developed by Andrew G. Howard et al., [6] where the authors use depth wise separable convolutions to build a lightweight deep learning model. Mark Sandler et al., [10] later improved upon the MobileNets and developed the MobileNet V2 where the authors introduce a novel framework for mobile object detection models called SSD Lite. Followed by this, the next generation of MobileNets which are based on certain search techniques and novel architecture designs. MobileNet V3 focuses on the exploration of how automated search algorithms and network design can work together to harness complementary approaches improving the overall state of the art in mobile classification, detection, and segmentation [5]. The goal of this paper is to develop the best possible mobile computer vision architectures optimizing the accuracy-latency trade-off on mobile devices. Through this process, the authors create two new MobileNet models for release: MobileNet V3-Large and MobileNet V3-Small which are targeted for high and low resource use cases. These models are then adapted and applied to the tasks of object detection and semantic segmentation. MobileNet V1 introduced depth-wise separable convolutions as an efficient replacement for standard convolution layers [6]. MobileNet V2 introduced the linear bottleneck and inverted residual structure to make even more efficient layer structures by leveraging the low-rank nature of the problem [10]. For MobileNet V3, the authors use a combination of these layers as building blocks to build the most effective models. Layers are also upgraded with modified swish nonlinearities, which is faster to compute and more quantization-friendly. MobileNet V3-Large is 3.2% more accurate on ImageNet classification while reducing latency by 20% compared to MobileNet V2 [5]. MobileNet V3-Small is 6.6% more accurate compared to a MobileNet V2 model with comparable latency. MobileNetV3-Large detection is over 25% faster at roughly the same accuracy as MobileNetV2 on COCO detection. MobileNetV3-Large LRASPP is 34% faster than MobileNetV2 R-ASPP at similar accuracy for Cityscapes segmentation [5]. To accomplish those, authors introduce the following: (1) Complimentary search techniques, (2) New efficient versions of non-linearities practical for the mobile setting, (3) New efficient network design, (4) New efficient segmentation decoder.

3. Method

3.1. Dataset

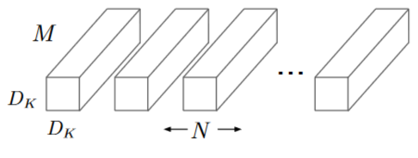
This project uses COCO dataset for object detection and classification. The Microsoft Common Objects in COntext (MS COCO) dataset contains 91 common object categories with 82 of them having more than 5,000 labeled instances. In total, the dataset has 2,500,000 labeled instances in 328,000 images. In contrast to the popular ImageNet dataset, COCO has fewer categories but more instances per category. This can aid in learning detailed object models which are capable of precise 2D localization.

3.2. MobileNet

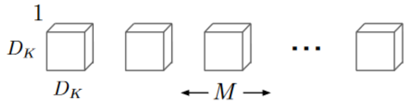
The model utilized for the object detection i.e., to obtain the bounding box and recognizing objects is called MobileNet. MobileNet V1 is highly efficient and works very well on mobile devices which are very constrained in regards to computational power.

3.2.1 Convolution Mechanism

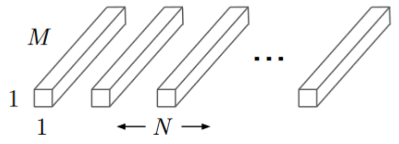
The MobileNets achieve this high efficiency by taking a small hit on the accuracy but the number of parameters used is significantly reduced. This is realized by using depthwise separable convolutions which is a two-step process. First, there is a depthwise convolution with a filter size of D_K x D_K x 1 with M filters applied on the input (D_F x D_F x M) [6]. This is then followed by pointwise convolution of 'N' filters with a size of 1 x 1 x M [6]. This leads to the total cost as shown in equation 1. The cost of a standard convolution would be as shown in equation 2. The number of computations required by depthwise separable convolutions is significantly lesser than standard convolutions. ”So in the case of MobileNets, the number of computations is 8 to 9 times lesser than standard convolutions because 3x3 convolutions are used in the MobileNet architecture” [6]. The contrast between a standard convolution and the depthwise seperable convolutions is shown in Figure 1.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



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

Figure 1. The filters in a standard convolution in (a) are substituted by (b) and (c) in MobileNet. Taken from: Andrew G. Howard et al., [6]

$$D_K * D_K * M * D_F * D_F + M * N * D_F * D_F [6] \quad (1)$$

$$D_K * D_K * M * N * D_F * D_F [6] \quad (2)$$

3.2.2 Network Configuration

The input size for the MobileNet is 224 x 224 x 3 and non-linearity is applied on all of the layers in the form of BatchNorm [7] followed by ReLU except for the final fully connected layer (FC) which is activated using a Softmax layer for the classification output. So essentially there is a 3 x 3 depthwise convolution followed by a 1 x 1 pointwise convolution which have non-linear components introduced by using BatchNorm [13] and ReLU. To reduce the dimensionality of the input, the convolutions are strided in the depthwise convolutions for every layer including the first layer. Average pooling is performed before the data is fed into the final FC layer. The MobileNet has a total of 28 layers. Howard et al.,[6] improved the efficiency in several ways to achieve the performance that is attained in the MobileNets. One of the ways that was done is reduce the computational load of the multiply-accumulate operations by using the more efficient General Matrix Multiply (GEMM) functions since most of the computation is done in the pointwise convolutions which will give dense vectors making the computations much faster compared to sparse vectors. Therefore, ”95% of training time is spent on training 3/4th of the parameters” [6].

3.2.3 Training

The MobileNet is trained in Tensorflow by using RMSProp [11] which is an optimization algorithm that prevents exploding gradients and essentially normalizes them. “Unlike other large models, MobileNet uses uses less regularization and data augmentation techniques because small models have less trouble with overfitting” [6]. Very little to no L2 regularization was used as the number of parameters was very low. The version of MobileNet used for this project was trained on the COCO dataset.

4. Experiments

The Experiment consists of four tasks which are described below:

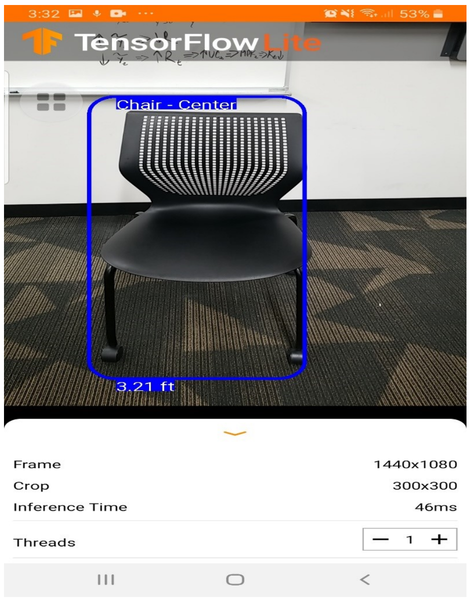
4.1. Object Detection

We reused a pre-trained model of Tensorflow Lite with a quantized MobileNet SSD model trained on the COCO dataset [2]. This application can detect 90 classes but for our case, we constrained the application to detect only three categories cars, people and chairs. The application detects the objects in the frames captured by the user's device's back camera. The application uses the boundary coordinates of the bounding box to calculate the relative position and other distance measurement tasks. The graphic user interface of the application is illustrated in figure 2.

4.2. Distance Measurement

We apply the Triangular Similarity to measure the distance between the object and a user's phone [9]. The Focal Length is different for each phone hence it is necessary to estimate it. After measuring the height of the object, the distance of the object from the user's smartphone camera is calculated. Based on the bounding box, the height of the object on screen is calculated by using equation 3. Android Studio helps in estimating the focal length by providing information on it. The focal length is calculated from the height of the image and Angle of View from the camera.The focal length measurement is described in 3.

$$f = \frac{(P * D)}{H} [4] \tag{3}$$



Frame	1440x1080
Crop	300x300
Inference Time	46ms
Threads	1

Figure 2. Graphic User Interface of the application

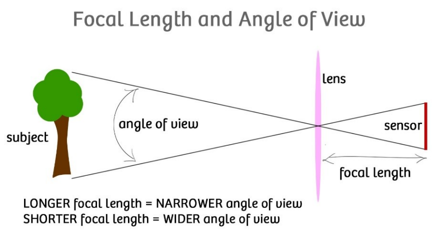

Focal Length and Angle of View
subject
angle of view
lens
sensor
focal length
LONGER focal length = NARROWER angle of view
SHORTER focal length = WIDER angle of view

Figure 3. Focal length measurement. Taken from: [12]

Where, P: Object' s Height on Screen (pixel),
D: distance of object from camera (inch) and
H: Height of the object (Inch).

We can get width and Horizontal View Angle from getPreviewSize and getHorizontalViewAngle function in Android given in equation 4.

$$f = \frac{(W * 0.5)}{tan((HVA * 0.5 * \pi/180)} \tag{4}$$

Where, W = Width of object in screen, HVA = Horizontal View Angle

The Triangular Similarity formula is applied to measure the distance from user's camera to the object given in equation 5.

$$D = \frac{H * F}{P} \tag{5}$$

where, D : Distance from the user's camera to the object
H: Object' s Height (inch)
F: Camera' s Focal Length (pixel)

P: Object' s Height on Screen (pixel)
We tried to measure the Focal Length from three different Android phones: OnePlus A5000, Google Pixel 1, Samsung Galaxy Note 8.

	Object's Height on Screen (pixel)	Real Height of Object (feet)	Real Distance from Object (feet)	Focal Length from Android Library (Pixel)
OnePlus A5000	653	2.9	5.1	1161
Google Pixel 1	597	2.9	5.1	1071
Samsung Note 8	560	2.9	5.1	1130

Table 1. Focal Length of Three Different Android Phones

4.3. Relative position

Relative position is calculated using the boundary coordinates of the bounding box along with the width and height of the phone's screen. For example, in figure 4 we can see the boundary coordinates of the bounding box of object 2 (top is 1500, bottom is 1800, left is 400 and right is 800) We define 5 different positions i.e., left, right, center, extreme left, and extreme right. We divide the screen into four parts evenly as shown in the image below. If an object lies within area covered by 1 and 2, its position should be marked as left. Moreover, if the object is completely inside area 1, its position is extreme left. We apply the same intuition for right and extreme right positions. However, if the object lies in area 2 and 3 (object 1), we should measure the difference between right edge of the box and the center line (blue color). If this difference is less than 1/8th of the screen's width and its left edge is in the area 1, the object 1's position is marked as left, otherwise it is center. Conversely, if the difference between left edge of the box and center line is less than 1/8th of the screen's width and its right edge is in the 4 area, the object 2's position will be marked as right, otherwise is center.

4.4. Text-to-speech

This application will help the visually challenged recognize three objects near them. We implemented the Text to Speech function in order to make the smart phone alert loudly for the user to be aware of the obstacles and avoid them. For this purpose, we used the Text-to-Speech class in Android Studio [1]. This library converts the strings to speech hence alerting the user about the object situated next to them. When multiple objects are recognized, the closest object will be identified.

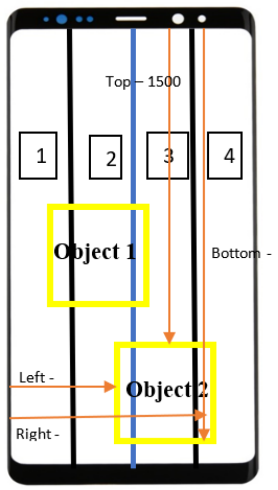


Figure 4. Object detected in the smartphone

5. Division of Work

All members of our team have put in equal effort into completing this project successfully within the stipulated time. The following Table 1 shows my peer evaluation for their contribution.
Anh The Nguyen worked on the Android programming in the TF-Lite file and text to speech functionality.
Monica Selvam worked on calculation of relative position of the objects.
Nikhillesh Sadassivam used COCO dataset to train the model and modify it to identify and classify the 3 objects of interest.
Srinivas Shashank Mulugu worked on calculation of the distance of object from the user.

6. Self-Peer Evaluation Table

Name	Score(out of 20)
Myself	20
Monica Selvam	20
Nikhillesh Sadassivam	20
Srinivas Shashank Mulugu	20

Table 2. Self-Peer Evaluation Table.

7. Conclusion

In this project, we used the Tensorflow Lite, an object detection model for mobile devices to build our object detection application for the visually challenged. We were able to use and modify the model to identify the three objects of interest (chairs, cars and people) using the camera of the mobile device. We were also able to calculate the relative position of the objects to the user into five distinct categories such as extreme left, left, center, right and extreme right. We also estimated the distance of those objects from the user and convey all this calculated information using text to speech functionality. We completed the application and tested it in the real world in different environments. Our application was able to clearly detect the objects of interest, its relative position and the distance from the user and alert the user with those details using generated voice.

References

[1] Handling focal length conversion to pixels for distance calculation— android. 4

[2] Object detection : Tensorflow lite. 3

[3] O. Alsing. Mobile object detection using tensorflow lite and transfer learning, 2018. 1

[4] D. Biga. Focal length conversion to pixels for distance calculation- android, Apr 2019. 3

[5] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al. Searching for mobilenetv3. *arXiv preprint arXiv:1905.02244*, 2019. 1, 2

[6] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 2, 3

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 2

[8] Z. Li and Z. Rao. Object detection and its implementation on android devices. *May 2017*, 2014. 1

[9] A. Rosebrock. Find distance from camera to object using python and opencv, Dec 2018. 3

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 1, 2

[11] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012. 3

[12] D. Tools. How to manual adjust the focus distance?, Jan 1970. 3