Real-time Object Detection using Deep Learning for helping People with Visual Impairments

1st Matteo Terreran
Dept. of Information Engineering
University of Padova
Padova, Italy
matteo.terreran@dei.unipd.it

2nd Andrea G. Tramontano
Dept. of Information Engineering
University of Padova
Padova, Italy
andreagaetano.tramontano@dei.unipd.it

3rd Jacobus C. Lock
School of Computer Science
University of Lincoln
Lincoln, United Kingdom
jlock@lincoln.ac.uk

4th Stefano Ghidoni
Dept. of Information Engineering
University of Padova
Padova, Italy
stefano.ghidoni@unipd.it

5th Nicola Bellotto
School of Computer Science
University of Lincoln
Lincoln, United Kingdom
nbellotto@lincoln.ac.uk

Abstract—**Object detection plays a crucial role in the development of Electronic Travel Aids (ETAs), capable to guide a person with visual impairments towards a target object in an unknown indoor environment. In such a scenario, the object detector runs on a mobile device (e.g. smartphone) and needs to be fast, accurate, and, most importantly, lightweight. Nowadays, Deep Neural Networks (DNN) have become the state-of-the-art solution for object detection tasks, with many works improving speed and accuracy by proposing new architectures or extending existing ones. A common strategy is to use deeper networks to get higher performance, but that leads to a higher computational cost which makes it impractical to integrate them on mobile devices with limited computational power. In this work we compare different object detectors to find a suitable candidate to be implemented on ETAs, focusing on lightweight models capable of working in real-time on mobile devices with a good accuracy. In particular, we select two models: SSD Lite with Mobilenet V2 and Tiny-DSOD. Both models have been tested on the popular OpenImage dataset and a new dataset, named L-CAS Office dataset, collected to further test models' performance and robustness in a real scenario inspired by the actual perception challenges of a user with visual impairments.**

Index Terms—**Object detection, real-time, electronic travel aid**

I. Introduction

As a fundamental part in computer vision and visual understanding, object detection is among the key components for solving more complex or high level vision tasks such as scene understanding, object tracking and activity recognition. Great improvements have been achieved in recent years, thanks to the advent of deep learning which allows very high performance levels when applied to object detection [1] [2] [3]. Object detection has found applications in a large variety of scenarios such as autonomous driving, surveillance, medical and industrial ones. However, there are many fields not yet explored. An interesting application is the use of object detection to help disabled people with their mobility, as proposed in the ActiVis project [4]. This project aims to enable people with vision impairments to independently navigate and find objects within an unknown indoor environment using only a mobile phone and its camera. Indeed, one main problem for blind or visual impaired people is to move across an indoor environment that they have never seen before, while avoiding possible obstacles. Consider for example a real scenario in which a blind person is looking for a chair in a new office. The person stands at the entrance of the room and wants to know where the chair is in the office, in order to reach it. He could wait for someone to escort him to the chair, or take out his smartphone and point it in front of him as in Figure 1. The device prompts the user to point the camera to the chair's most likely location, using any detected objects during this transition to refine the location, ultimately guiding the user towards the chair. Such system could hence help blind people to be more autonomous in the mobility in indoor locations and decrease the time for many operations. Although the project focuses on office environments, it is easy to find

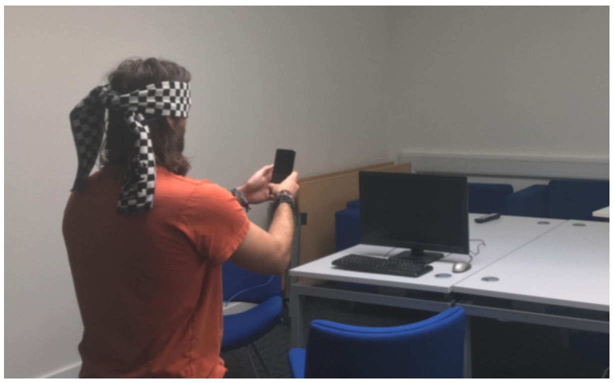
Fig. 1: Example of the system proposed in the ActiVis project, during an experiment with a blindfolded participant. Source: [4].

possible uses in other areas, such as waiting rooms, kitchens, etc. In such scenario object detection plays a central role. It should be very accurate and precise to correctly localize the objects required by the user, but also all the possible obstacles in the room to guide the user safely within the environment. Moreover, it should work in real-time since the user needs an instantaneous feedback from the system regarding where and how to move. In addition, the project foresees that the object detection is realized using the user's mobile phone, which introduces some challenges in the development. One is the hardware limitation, since mobile devices are not equipped with powerful GPUs commonly used for state-of-the-art object detection models. Another challenge is related to robustness: object detection should work on images acquired with a mobile phone, which could be blurred or very shaky due to anomalous movements of the user. This is especially important when one considers that the visually impaired population are often elderly people. Note that running the object detector directly on the smartphone is a key element of the ActiVis project. It aims to create a fully-enclosed system that does not need to rely on external services and could work also in poorly-connected or totally unconnected areas; solutions such as cloud detectors are not suitable in this case.

In this work, we analyze and compare several state-of-the-art object detection models to be integrated in the proposed application. As described, performance and computational complexity are crucial elements, since the object detection models will be integrated on low performance devices. This leads to narrowing the choice between two possible architectures, namely SSD Lite [5] and Tiny-DSOD [6]. Both networks have been tested on different datasets, in order to investigate which architecture offers the best compromise between speed and accuracy. In particular, we collected a new office-themed dataset with video and image data to test the networks performance in a real scenario, similar to the one used in the project's evaluation [4]. It is important to test the application also on videos to evaluate the robustness of the models under conditions similar to those in which a person uses the system. A test made only on images, does not take into account the noise introduced by the user movements, which can produce shaky and blurred images that are more challenging for the object detectors. Summarizing, this work presents 3 main contributions: (i) a novel object detection scenario targeting mobile devices, which can push research in this field; (ii) experiments evaluating several object detection models, in order to find the most suitable to be implemented in the proposed application; (iii) a new dataset, named L-CAS Office dataset[1], collected to test object detection accuracy and robustness on a real scenario.

The remainder of the paper is organized as follows. Section II presents the state-of-the-art for object detection and the related works made till now. Section III describes the architectures we selected and the main differences between them. In Section IV we present the datasets used for training and testing, while in Section V the results achieved by all networks are reported and discussed. Finally, in Section VI, conclusions are drawn and future directions of research identified.

[1]Available at https://lcas.lincoln.ac.uk/wp/research/data-sets-software/

II. Related works

Object detection systems based on deep learning (DL) could be divided into two main categories: single-stage based methods and two-stage based methods. Two-stage based methods use a "region proposal" approach: first candidate regions of interest are produced by an external algorithm, then a convolutional neural network (CNN) performs classification on each candidate region and computes a bounding box for each object within that candidate region proposal. For example, Fast R-CNN [7] rely on the Selective Search algorithm [8] to produce region proposal candidates. Faster R-CNN [1] builds upon the previous architecture proposing an efficient and accurate Region Proposal Network (RPN) to generate region proposals without using external algorithms. Generally, two-stages based methods are accurate but with heavy computing cost, and thus yield slow processing speed.

Single-stage based methods like SSD [3] and YOLO [2] [9] instead, directly compute classification and detection from the images with a single forward pass by applying sliding windows of different size to the input image. In this way, they are able to check if the object appears in the windows without the necessity of involving region proposal systems or post classification layers. Single-stage based methods are usually faster than the two-stage based methods, but also less accurate. In many cases they share a common structure composed of two sub-networks: one sub-network to extract features from the input and one to classify and localize the objects. The former part is called backbone network, while the latter one is called front-end network. Single Shot Detection (SSD) network [3] uses VGG16 [10] as backbone, and a front-end with multiscale convolutional layers to achieve fast detection speed and high detection quality. Deeply Supervised Object Detector (DSOD) [11] uses instead DenseNet [12] as backbone and introduces Deep Supervision to address the issue of the vanishing gradient.

A common strategy is to use deeper networks to get higher performance, but that leads to a higher computational cost which makes it impractical to integrate them on mobile devices with limited computational power. In this scenario a break-through is represented by depth-wise separable convolutions proposed in the Mobilenet architecture [13] for classification tasks. Depth-wise separable convolution consists of a depth-wise convolution followed by a point-wise convolution (also named 1×1 convolution), which greatly reduces the number of operations required to compute convolutions. Many object detection models adopted Mobilenet as a features extractor, like SSD Lite [5], reducing their size with respect to the same models implemented with standard convolution. This drastically reduces the memory usage and improves the speed on low resources devices. Even without using the Mobilenet architecture as backbone, other models like Tiny-DSOD [6] rely on depth-wise separable convolutions to reduce model complexity. Indeed, Tiny-DSOD introduces the Depth-wise Dense Block, an efficient network structure to combine depth-wise separable convolutions with densely connected networks like DenseNet [12]. These final considerations motivate our choices for implementing a real-time object detector on a smartphone-based assistive device.

III. Methods

We restrict our attention to two architectures selected on the basis of real-time performance and model complexity, namely SSD Lite with Mobilenet V2 [5] and Tiny-DSOD [6]. Indeed, we want a model which is light and capable of running on a mobile device like a smartphone with real-time performance. State-of-the-art models like Faster-RCNN [1] are not able to reach real-time performance with a Nvidia Titan X GPU. Therefore, we can assume it can not reach real-time performances on a mobile platform. Other networks like YOLOv3 [9] achieve better performance in terms of speed and accuracy, but they are too big in terms of memory to be ported on a mobile device. YOLOv3 has also a light version, named Tiny-YOLOv3, which is about 30 times smaller compared with its standard version. We tested Tiny-YOLO in a preliminary setup on a smartphone, namely an Asus ZenFone AR 4.2, using a mobile implementation developed with OpenCV library and YOLO [2] framework. However, the results were not satisfactory: the implementation was not compatible with the smartphone GPU and runs only on CPU, limiting the possible frame rate to an average speed of 2 FPS. Furthermore, when tested with video data, the network was only able to classify objects when the device was held still. The slow frame rate compounded the motion blur and made classification impossible.

These initial tests confirmed that the structure of this network is not able to reach real-time performance on a mobile device, therefore it was disregarded. SSD Lite and Tiny-DSOD architectures instead, when tested on the same smartphone CPU, reach a higher frame rate around 20 FPS. The higher frame rate makes such networks even more robust at small camera movements, since they can work on less blurred images. In the following, these two architectures are described in detail while their performance are reported and discussed in Section V.

A. *Single Shot Detection Lite (SSD Lite)*

SSD Lite is a lighter version of the standard SSD architecture, a single-staged network composed of two parts: a backbone network that extracts the features from the input image, and a front-end part that classifies and localizes the objects from the features provided. Compared to SSD architecture, in SSD Lite the backbone network is substituted with Mobilenet V2 architecture, a light classification network designed for mobile devices. Moreover, to reduce computational burden, all the convolutions in the front-end part are substituted with depthwise separable convolutions.

The backbone network extracts feature maps from the input, with a smaller resolution as the network goes deeper. The front-end part uses such multiple features maps at different scale to compute predictions. In particular, SSD Lite uses a batch of six feature maps: two feature maps directly from Mobilenet V2 (output of block 13 and 17, the last block), and four feature maps computed from the last feature map of Mobilenet V2 by means of four depthwise separable convolutional layers. Each of the six feature maps is then subdivided into cells with a grid. Different bounding boxes are then applied at the center of each cell in the feature maps: four bounding boxes for feature maps directly from the backbone, and six bounding boxes for the feature maps obtained from convolutional layers. As in SDD architecture, the bounding boxes applied are not random guesses, but selected from a set of default bounding boxes which represent different boundary shapes. From each bounding box considered in the cells grid, the network produces one prediction. A single prediction is an array containing the confidence predicted for each class and the coordinates of the bounding box with the highest confidence. All the predictions are then combined and analyzed: predictions of the same class are compared and grouped together, predictions of different classes in the same location are checked. Finally, the final set of predictions is obtained after a Non-Maximal-Suppression operation to delete repeated and false bounding boxes.

B. *Tiny-DSOD*

Tiny-DSOD is an object detection network based on the single-shot detector (SSD) [3] framework and the deeply-supervised object detection (DSOD) framework [11], which consists of a backbone part and a front-end part. It promises good performance in mobile devices while maintaining a good trade-off between speed and accuracy. The main novelties proposed by Tiny-DSOD are the Depthwise Dense Block (DDB) based backbone and a feature-pyramid-network (D-FPN) based front-end. The Depthwise Dense Block (DDB) is an efficient network structure to combine Depthwise Separable convolutions with densely connected networks as DenseNet [12], inspired by the inverted residual blocks proposed in Mobilenet [13]. Given an input features map, this is first compressed to g channels using a 1×1 convolution, and then a depthwise convolution is applied. The generated output is finally concatenated to the input feature map.

The feature-pyramid-network (D-FPN) is a lightweight version of FPN [14], that fuse semantic information from neighborhood scales to speed up object detection. The front-end predictor D-FPN consists of a down-sampling path made of convolutions and max-pooling layers, and a reverse up-sampling path using a Bilinear interpolation layer followed by a depthwise convolution. The feature maps obtained in each up-sampling step are then merged with the same-sized feature maps in the down-sampling path via element-wise addition. The backbone network in Tiny-DSOD is composed of four DDB stages followed by one transition layer to fuse channel-wise information from the last stage and compress the channel number for computing and parameter efficiency. The D-FPN takes as input the feature map computed by the backbone and outputs six feature maps at different scales which are then used for the detection using the same mechanism described in Section III-A for SSD Lite. Since they rely on the same detection framework, Tiny-DSOD and SSD Lite have the same layers at the end of the network. Moreover, they use the same defaults bounding boxes and the same multi-scale features parameters. The only difference is how the set of feature maps for the multi-scale detection it is generated: with D-FPN in Tiny-DSOD, while with Mobilenet V2 and convolutional layers in SSD Lite.

IV. Datasets

To evaluate the networks we consider different datasets, namely the OpenImage dataset [15] and a new self collected dataset specifically created for the project. OpenImage is used for training and testing the networks, while the remaining dataset is used to test the models in conditions similar to the project's evaluation environment. Testing the models on different datasets allows us to better investigate their performance. On the one hand, testing on OpenImage allows to understand the weaknesses and the strengths of the networks in term of learnability, since test images come from the same distribution of the training data. On the other hand, using a test set coming from a different distribution, helps to understand how robust the model is. Furthermore, using a real environment as test set, we can better understand how well the model works in the real-world.

A. *OpenImage Dataset*

OpenImage [15] is a dataset of 9M images annotated with image-level labels, object bounding boxes, object segmentation masks, and visual relationships. It contains a total of 16M bounding boxes for 600 object classes on 1.9M images, making it the largest existing dataset with object location annotations. The OpenImage dataset contains a large number of object categories, especially object from office environments. We selected a subset of images to train and test our models, considering common objects in an office room that an impaired person would like to look for.

The selected categories are reported in Table I, together with the number of images for each object category. Categories like *Person*, *Man* or *Woman* were not considered since the scope of the project was specifically limited on finding objects in an unknown indoor environment. People can make their location known simply by talking, so the search for a person is not so interesting for the project. To limit the dataset size, we take a maximum number of 10.000 images per category. The selected images are equally divided in three groups following a 60/20/20% partition in train, validation and test sets.

TABLE I: Selected object categories from OpenImage dataset.

Object category	# Images	Object category	# Images
Backpack	1314	Light switch	114
Book	10000	Monitor	6384
Bookcase	5708	Mouse	841
Chair	10000	Mug	2403
Desk	10000	Plant	10000
Door	10000	Telephone	316
Keyboard	4757	Whiteboard	1056
Lamp	3849	Window	10000
Laptop	10000		

B. *L-CAS Office Dataset*

The L-CAS Office dataset is created on purpose for the ActiVis project. The pictures are taken from a real office in our university and it contains all the 17 objects reported in Table I. To acquire the images we use an Asus ZenFone AR 4.2 equipped with: 4GB of RAM, 64-bit CPU Qualcomm Snapdragon Quad-Core 821 2.35 GHz, optimized for Tango, Adreno 530 GPU, 23 megapixel camera, with Android 8.

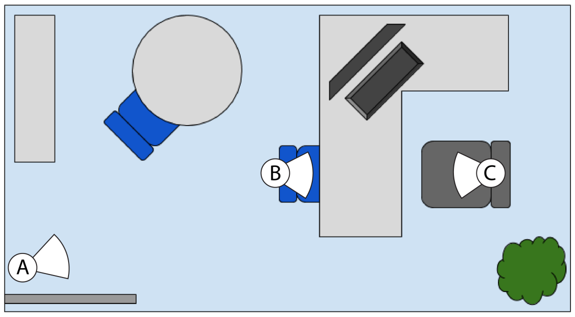

Fig. 2: Schematic representation of the office and the three different points of view considered: door position (A), guest position (B), owner position (C).

The dataset is composed of pictures captured in the same environment, an office, with the objects fixed in the same position. A schematic representation of the office is depicted in Figure 2. We take pictures from three different points of view: (i) door position, when the user opens the door and looks inside the room office; (ii) guest position, when the user sits down on the chair in front of the desk; (iii) owner position, when the user sits down on the chair behind the desk. An example of images taken from the three different points of view is shown in Figure 3. For each point of view, pictures have been acquired according to three different heights and three different inclinations of the smartphone. As possible heights we consider: when user holds approximately the smartphone at the height of the chest, head height and pelvis height. Regarding the different inclinations we assume no rotation when the user holds the smartphone in portrait mode, and consider rotations of 45° on the right and on the left. We decide not to take pictures in the landscape mode, because it is uncommon or uncomfortable position to hold the smartphone when a person is using it. The total amount of captured pictures is 459 pictures, with 27 images for each of the 17 objects considered. All the images have been annotated using the LabelImg annotation tool[2], which offers a graphical interface to quickly annotate the dataset by drawing a bounding box around each object of interest; for

[2]https://github.com/tzutalin/labelImg

each bounding box it is possible to assign a label representing the class of the object. The annotations are then saved as XML files in PASCAL VOC format.

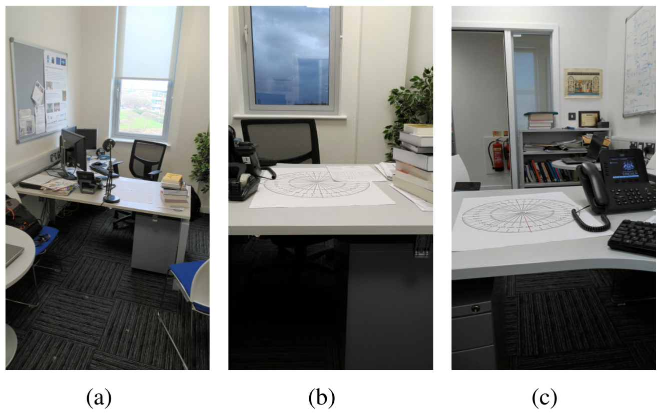
Fig. 3: Three pictures taken from the L-CAS Office dataset. The images are centered on the desk object, respectively from: door position (a), guest position (b), and owner position (c).

C. *Video Test Dataset*

We also collected nine different videos, taken from three different office rooms. The first one is a small office of approximately $10\,m^2$, the same used to acquire the Office Dataset previously described. Then a medium office of approximately $50\,m^2$ and a big room of approximately $300\,m^2$. We choose three different room dimensions, to better understand if the models are able to recognize objects at different distances. In each of the three rooms, three different videos are recorded at three different speeds: slow speed, medium speed and high speed. The speed is intended as the velocity of the movements that the user produces, while he is walking in the room. At slow speed, the user walks very slow and the acquired video is not blurred and the scene change gradually; increasing the speed, the user's movements become faster and the acquired video gets blurred with rapid scene changes. The videos are only used for a qualitative evaluation of weaknesses and strengths of the implemented object recognition systems.

V. Experimental results

We perform two sets of experiments to investigate which network is more suitable for our project. Indeed, we are looking for an algorithm with real-time performance ($\geq$ 20 FPS), robust to anomalous movements of the user and to blurred frames, and which avoids as many detection errors as possible. We focus on SSD-Lite and Tiny-DSOD models. We train and evaluate both models on the subset of OpenImage dataset described in Section IV. The same trained models are then tested on the Office dataset to evaluate their performance in a real scenario. To evaluate the performance of each model we refer to the mean Average Precision (mAP) metric, computed as the mean over the average precision on each object category.

A. *Experimental setup*

Experiments have been developed using the Keras and Tensorflow libraries. We do not use the original implementation of the models, but we developed our own Keras version of SSD-Lite with Mobilenet V2 and Tiny-DSOD. Developing these models with the Tensorflow framework is convenient, since it is compatible with Android and allows to easily export the models to run on mobile devices. For both architectures we follow the respective paper guidelines and hyper-parameters, trying to be as close as possible to the original implementations. Experiments have been performed on the Asus ZenFone smartphone used to acquire the datasets and a desktop computer with GPU Nvidia Geforce GTX 1050Ti 4GB, 16GB RAM, i7-4770 CPU 3.40GHz. The smartphone was used for qualitative evaluation on the Video dataset and measuring the average speed in terms of FPS, while the computer was used for training the models and quantitative evaluation of their performance on OpenImage and Office datasets. Both architectures have been trained for 120 epochs, with 943 iterations each and a batch size of 24 (maximum permitted from the GPU).

B. *Networks comparison*

The results obtained are reported in Table II, which highlights SSD-Lite as the best model on both datasets. Indeed, it reaches a higher mean accuracy and also regarding single object categories, it gives better results in most cases. Tiny-DSOD architecture instead, has a much lower mean accuracy and the accuracy per object reveals that the model learns to recognize really few of them: more than half have less then 10 AP accuracy, and five of them have less then 1 AP accuracy. Also SSD-Lite has four objects with accuracy less then 1 AP, but more then half of the objects have an accuracy greater then 10 AP. A good threshold of confidence for SSD-Lite's predictions is of 70%: using this threshold the majority of the wrong predictions are discarded.

However, both models do not achieve high accuracy in our experiments. The main reason for that could be the number of training images, not enough for some object categories. For example, Light-switch and Telephone categories which achieve an accuracy less then 1 AP are also the categories with less training images as reported in Table I; in such cases it is difficult for both models to learn an effective object representation. Moreover, the total number of training images could be not enough for the complexity of the networks. Indeed, a complex network has a huge amount of parameters to be trained, requiring more examples and a longer training to perform well.

Analyzing the predictions of the two models, it is possible to see that Tiny-DSOD produces many wrong bounding boxes with high confidence, as depicted in Figure 4b. SSD-Lite architecture instead, produces only few correct detections with a nice confidence as shown in Figure 4a. When tested on the Video dataset, both SSD-Lite and Tiny-DSOD reach a speed of approximately 20 FPS, which is sufficient to detect objects if the user is not moving too fast. Indeed, on the slow

TABLE II: Results in terms of accuracy of SSD-Lite and Tiny-DSOD, both on OpenImage and Office test set.

Object	**SSD-Lite**		**Tiny-DSOD**	
	OpenImage	**Office**	**OpenImage**	**Office**
Backpack	15.5	0.0	3.2	0.0
Book	9.1	0.0	0.6	0.0
Bookcase	31.4	43.7	14.9	22.6
Chair	1.0	0.0	0.1	0.0
Desk	11.1	3.1	9.8	5.1
Door	17.0	18.2	9.7	8.4
Keyboard	18.5	0.2	3.3	0.0
Lamp	12.9	0	4.6	0.0
Laptop	22.4	19.6	10.0	11.1
Light-switch	0.0	0.0	0.0	7.0
Monitor	2.4	11.4	5.1	5.6
Mouse	15.3	0.0	4.5	0.0
Mug	28.8	0.0	5.4	0.0
Plant	24.3	60.3	16.4	32.7
Telephone	0.0	0.0	0.0	0.0
Whiteboard	64.7	7.3	19.3	3.9
Window	0.3	0.0	0.8	0.0
MEAN	**16.2**	**9.6**	**6.3**	**5.3**

speed videos SSD-Lite model is able to recognize with a good confidence only few objects and has difficulties to detect most of the objects in both medium and high speed videos, with many wrong predictions. In particular it is able to detect Desk, Door, Laptop, and Whiteboard if the user movements are slow and the objects are close enough. Tiny-DSOD model instead, produces too many wrong predictions with high confidence, invalidating the actual purpose of our application.

C. Fine-tuning

Between the two model candidates, the previous experiment highlighted SSD-Lite as the most promising object detector for our application. Therefore we tried to improve its performance on our scenario by using the fine-tuning technique, generally used when the dataset for a particular application is limited: the model is first trained on bigger dataset (e.g. ImageNet [16]) and then again on the small application-specific dataset. In such a manner, the model can improve its performance on the small dataset by relying on powerful features learned using the bigger one. We used SSD-Lite with Mobilenet V2 pre-trained on ImageNet dataset, and finetuned the whole network on the same train set of images from OpenImage used previously.

The results obtained are reported in Table III together with the results achieved from the network trained from scratch. As shown in the fourth column of Table III the mean accuracy on OpenImage Test set is doubled. Although there is no great improvement for the mean accuracy on the Office dataset, the fine-tuned model detects more objects with a good confidence. The predictions are now more often correct and with less mistakes, as shown in Figure 4c. Regarding the results on the Video dataset, the model is not able to detect all the objects in all the frames, but the improvements are visible: although few wrong predictions and some missed ones, the model works well when the objects are big in the frame (either close to the

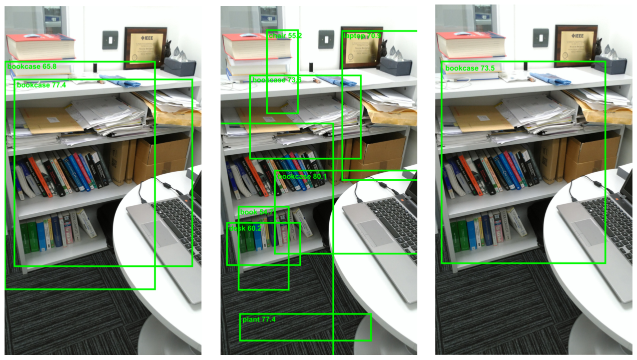
(a) (b) (c)

Fig. 4: Prediction examples on the L-CAS Office dataset of: (a) SSD-Lite, (b) Tiny-DSOD and (c) SSD-Lite fine-tuned.

camera or big objects). Also in the video at medium speed, it detects big and close objects, while in the fast speed videos, it still has some difficulties.

In conclusion, the best model we developed for our application is SSD-Lite with fine-tuning. Although the obtained accuracy is rather low on some classes, the model showed good overall performance on the Video dataset which replicates conditions similar to a real use of the system. This, together with the frame rate of about 20 FPS achieved, has led to choose this model for a first implementation of the system foreseen in the ActiVis project. In Table IV the performance of all models considered are reported in terms of accuracy, complexity and speed. Speed is computed in terms of Frame per second (FPS), measured on the Asus ZenFone smartphone with models running on CPU. Complexity is evaluated with respect to the number of parameters of the model (Params) and FLoating point Operations Per Second (FLOPS) required.

TABLE III: Results in terms of accuracy of SSD-Lite and SSD-Lite finetuned, both on OpenImage and Office test set.

Object	**SSD-Lite**		**SSD-Lite Fine-tuned**	
	OpenImage	**Office**	**OpenImage**	**Office**
Backpack	15.5	0.0	43.3	14.8
Book	9.1	0.0	0.6	0.0
Bookcase	31.4	43.7	39.4	9.9
Chair	1.0	0.0	1.2	0.0
Desk	11.1	3.1	27.9	1.7
Door	17.0	18.2	25.0	3.7
Keyboard	18.5	0.2	36.5	12.3
Lamp	12.9	0	22.82	0.0
Laptop	22.4	19.6	43.8	26.1
Light-switch	0.0	0.0	77.7	0.0
Monitor	2.4	11.4	35.6	4.9
Mouse	15.3	0.0	39.7	0.0
Mug	28.8	0.0	59.9	0.0
Plant	24.3	60.3	24.0	33.3
Telephone	0.0	0.0	11.4	0.0
Whiteboard	64.7	7.3	68.8	41.1
Window	0.3	0.0	2.5	22.2
MEAN	**16.2**	**9.6**	**32.9**	**9.7**

TABLE IV: Results in terms of mean accuracy (mAP), speed (FPS), and weights (Params and FLOPS)

Model	**mAP**		**FPS**	**Params**	**FLOPS**
	OpenImage	Office			
SSD-Lite	16.2	9.6	~ 20	4.30M	0.8B
Tiny-DSOD	6.3	5.3	~ 20	0.95M	1.12B
SSD-Lite (fine tuned)	32.9	9.7	~ 20	4.30M	0.8B

VI. Conclusions

In this work we focus on object detection targeting mobile applications as proposed in our recent project, which aims to develop a system to help visually impaired people to navigate in indoor environments. In such scenario, an object detector that is light and accurate is fundamental to scan the surrounding space. We select and evaluate two state-of-the-art models capable of working on mobile devices and that promise good accuracy: SSD-Lite and Tiny-DSOD. A novel dataset has been acquired to test the models on a real scenario and highlight their robustness. Among the two models, SSD-Lite achieves better performance, which is further improved in a fine-tuned version and currently used in the ActiVis project. The system presented is limited to find objects in an office scenario and guiding the user. In the future we wish to validate the system on other indoor environments (e.g. a domestic scenario) and considering new classes such as a *person* class to aid in avoiding possible collisions, hence improving navigation and achieving 3D guidance.

Acknowledgment

This research is partly supported by a Google Faculty Research Award (Winter 2015). Part of this work was supported by MIUR (Italian Minister for Education) under the initiative "Departments of Excellence" (Law 232/2016).

References

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conf. on computer vision and pattern recognition*, 2016, pp. 779–788.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[4] J. Lock, A. Tramontano, S. Ghidoni, and N. Bellotto, "Activis: Mobile object detection and active guidance for people with visual impairments," in *International Conference on Image Analysis and Processing*. Springer, 2019, pp. 649–660.

[5] M. B. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[6] Y. Li, J. Li, W. Lin, and J. Li, "Tiny-dsod: Lightweight object detection for resource-restricted usages," in *BMVC*, 2018.

[7] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[8] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[9] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] Z. Shen, Z. Liu, J. Li, Y.-G. Jiang, Y. Chen, and X. Xue, "Dsod: Learning deeply supervised object detectors from scratch," in *Proceedings of the IEEE international conf. on computer vision*, 2017, pp. 1919–1927.

[12] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[13] 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.

[14] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[15] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv preprint arXiv:1811.00982*, 2018.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.