Artificial Intelligence in Visual Impairment Rehabilitation Engineering

S. Maya, *UP201700077* S. Pedro, *UP201605633*

I. Abstract

Artificial Intelligence (AI) has recently proven successful in several applications related to a myriad of different fields. In this work, we will briefly cover the state of AI in the ongoing task of visual impairment rehabilitation. We will fully describe a complete AI pipeline for aiding the blind, its results and the concepts behind each part of the process.

II. Introduction

Artificial Intelligence (AI) refers to the processing and intuition displayed by machines. It is an ample field of study that encompasses concepts such as Machine Learning (ML) and Deep Learning (DL). In recent years, AI has seen exponential growth, particularly in the field of DL, due to the models' increased capacity for processing data, as well as the insurgence of interesting novel architectures for Deep Neural Networks (DNNs). The cause of this phenomenon is the increase in processing power of computers currently in the market. Breakthroughs in computing efficiency have led to smaller, more powerful devices, that can run costly algorithms.

These advances are particularly interesting in rehabilitation engineering, as one of the greatest catalysts for rehabilitation projects are portable devices such as smartphones. These devices now have the capability to not only connect to servers that run DL models, but also run these models themselves with their own processors. They also contain powerful cameras that are able to capture the user's surroundings, in video format with high resolution, which is interesting to consider for navigation software for the visually disabled capable of detecting objects and transmitting the information to the user in real time.

Efficient object detection is only possible due to the use of dense Convolutional Neural Networks (CNNs) and image processing algorithms such as You Only Look Once (YOLO) and Single Shot Multibox Detector (SDD). Kyr et. al [1] describe how these algorithms perform and how they compare with each other, as well as other previously proposed pipelines.

A person's freedom is in great part dictated by the actions they are allowed to perform. Freedom of transport is limited by an individual's ability to see and analyse their surroundings, which is why navigation systems are important to the rehabilitation of the visually impaired. In this work, we will describe how one of these systems is built and implemented, based on the work in [1], using novel DNN and image processing technology combined with already established rehabilitation systems.

III. Visual Impairment and Object Detection

According to Kyr et. al [1], a test of algorithms of type YOLO and SDD, both optimized to guarantee better performance scores, was done to compare them in matters of both Detection Speed as well as Detection Accuracy.

A. YOLO

With the aim of making Object Dectection faster, the YOLO algorithm works through a "single neural network" to detect and classify objects based on regression models. The algorithm is constantly being trained, current libraries used are "Picasso and People-Art Dataset" and, for the purpose of this project, Tensorflow Lite. The latter is an open-source, google owned application programming interface (API) that allows users to run complex machine-learning models on mobile devices, minimizing latency.

Using YOLO, the image is divided into a SxS grid of size 448x448) and possible object boxes are drawn around the whole image,those with higher 'Intersect Over Union (IOU)' are detected as being true objects and thus, proceeds to be classified by regression techniques, appointing the "grid box" containing the center of the detected object as the responsible for classifying it.

Although the explanation of the process made it seems working linearly - one thing happening at a time - the big differential of this algorithm is its capacity of doing both detection and classification at the same time. Results presented show a high mean Average Precision (mAP) when comparing to others "Real-Time detectors" while maintaining groundbreaking results up to 155 detected frames per seconds, outperforming the 2nd fastest model by 55%. Those results may be analysed on the image 1 of Appendix A. [2]

To conclude, the proposed limitations refers to the ability of the model to detect objects with little to none physical separation in between - a limitation of the object detection method - as well as those who appears slightly disfigured - pointed as a limitation of the database used to train the model.

B. SDD

A Single Shot MultiBox Detector is composed of two simultaneous phases:

1) The extraction of feature maps using "convolutional neural networks architecture" of type VGG16. Those work by receiving images of resolution equals to 224x224 and applying 16 layers of "learned" filters with size 3x3 - basic requirement to store information

about X,Y axis - in order to "summarize the presence of certain features in the image". By stacking those filters the researches manage to separate the level of feature detection i.e, those closer to the input image would detect 'easier' and tangible' features like lines while those on the back of the list, and closer to the output map, would be responsible for detecting abstract features such as "shapes", "forms" or objects, achieving an overall better solution. To have a better understanding, refers to the scheme present on image 2 of Appendix B.

The problem with feature maps, however, is that for each small movement / adjustment of any detectable object on the image, a completely new feature map would have to be created, sacrificing performance. On that note, VGG16 architecture uses what is called "pooling-layers" to downsample the overall size of the feature map created in each step by a size of 2, turning the process overall more efficient. Image 3 of Appendix B provide a practical example on the importance of Feature Maps. [3]–[6]

2) The detection of objects is, then, made with a "CONV4_3" Layer of size 38x38 that makes 4 object predictions for each pre-determined cell of the image and runs a classification algorithm to test it for all the classes that it has been trained on. Each class has 21 different score and, finally, the higher one is then picked as the object detection. This process is summarized in the scheme present on image 4 of Appendix B.

To conclude, tests results shows an overall better data accuracy with an SDD based model when compared to the regular YOLO method but a much slower processing speed in comparison with the fast YOLO method, such as seen in the figure 5 of Appendix B.

IV. System Implementation

Efficient object detection is only part of the large task of designing a good navigation system. It is equally important to be able to implement it in a way which is practical, accessible and safe.

Once trained, an object detection model must be ran on a device that is capable of fast processing, as the model will be making predictions for each video frame. Furthermore, they will have to be in the correct format depending on the device. Luckily there are several mobile development frameworks that have the tools to interpret these models and run them on the device. This is the case of CoreML (for iOS devices) [7] and TensorFlow Lite (for Android devices) [8]. These frameworks allow for the extraction of model predictions and information, which can later be passed on to other software present in the device or even other devices through the use of wireless communication protocols.

With full access to the model prediction, this information has to be transmitted to the user. One already established strategy is using text-to-speech technology [9], meaning the predictions are passed on to a software that converts the text into a sound signal, and then is transmitted to the user via headphones or speakers. The information would include what objects are close to the user and their relative locations.

One problem that this strategy might cause is sensory overload, i.e., when too much information is passed on to the user, who could become disoriented or confused. To mitigate this, there should be a part of the pipeline dedicated to filtering irrelevant information, by confidence levels for example.

An interesting application for this pipeline is its combination with intelligent wheelchairs [10]. For an individual that suffers from a visual and motor disability, this could improve safety. By having the object detection software linked to the intelligent wheelchair's control commands, one could code a way to control the wheelchair if need arises. An example could be activating a break system when the user is in a collision course with an object, with no signs of stopping.

V. Conclusion

This work analyses previous efforts to create object detection models for the visually impaired. We can see that different strategies must be evaluated in accuracy and computation weight.

Looking at results, it can be noted that SDD algorithms have the potential to perform better in terms of accuracy. However, Fast YOLO algorithms are presently much faster than any other algorithm, and thus could present a better solution for mobile applications.

We also note that there are other factors, aside from performance, that influence the utility of object detection software for the blind. Notably, the user interface must receive as much attention as the model training.

In conclusion, there are well established object detection technologies that grant visually impaired individuals a higher level of freedom, which can be used when combined with proper support.

References

[1] Rahime Kyr, Atike Iscan, and Pynar Kyrcy. A mobile application for the visually handicapped. pages 221–224, 12 2021.

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[3] Ashwani Kumar, Zuopeng Justin Zhang, and Hongbo Lyu. Object detection in real time based on improved single shot multi-box detector algorithm. *Eurasip Journal on Wireless Communications and Networking*, 2020, 12 2020.

[4] What is vgg16? — introduction to vgg16 | by great learning | medium.

[5] Ssd object detection: Single shot multibox detector for real-time processing | by jonathan hui | medium.

[6] A gentle introduction to pooling layers for convolutional neural networks.

[7] Oge Marques. Machine learning with core ml. In *Image Processing and Computer Vision in iOS*, pages 29–40. Springer, 2020.

[8] Oscar Alsing. Mobile object detection using tensorflow lite and transfer learning, 2018.

[9] Paul Taylor. *Text-to-speech synthesis*. Cambridge university press, 2009.

[10] Louis Lecrosnier, Redouane Khemmar, Nicolas Ragot, Benoit Decoux, Romain Rossi, Naceur Kefi, and Jean-Yves Ertaud. Deep learning-based object detection, localisation and tracking for smart wheelchair healthcare mobility. *International journal of environmental research and public health*, 18(1):91, 2021.

APPENDIX A
YOLO ALGORITHM MAP AND FPS PERFORMANCE RESULTS

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Fig. 1. Table comparing the mAP and FPS performance of different object detection algorithms [2]

APPENDIX B
SDD PIPELINE

A. VGG16 Schematic Representation

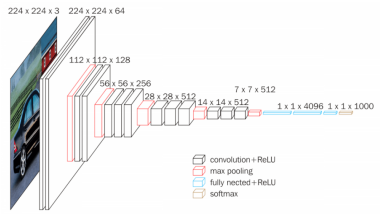

Fig. 2. Scheme representing the innumerous layers on a VGG16 model [5]

B. Feature Map Real World Application

Fig. 3. Feature Maps are important because different perspectives may alters an object shape and scale [5]

C. Schematic Representation of SDD

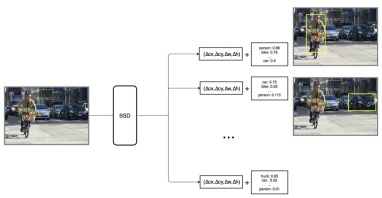

Fig. 4. Scheme exemplifying how an SDD algorithm works [5]

D. Results from SDD model

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

Fig. 5. Comparison SDD vs Others Object Detection Algorithms [5]