Convolutional Neural Network for Object Detection System for Blind People

Y.C. Wong, J.A. Lai, S.S.S. Ranjit, A.R. Syafeeza, N. A. Hamid
Micro and Nano Electronic (MINE) Research Group, Centre for Telecommunication Research & Information
Fakulti Kejuruteraan Elektronik dan Kejuruteraan Komputer, Universiti Teknikal Malaysia Melaka,
Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia
ycwong@utem.edu.my

***Abstract*— Blind or visually impaired people are usually unaware of the danger that they are facing in their daily life. They may face many challenges when performing their daily activity even in familiar environments. This work proposed a smart object detection system based on Convolutional Neural Network (CNN) to provide a smart and safe living for visually impaired people. To reduce the complexity load, region proposals from the edge maps of each image were produced using edge box algorithm. Then, the proposals passed through a fine-tuned CaffeNet model. The object scene was captured by the webcam in real time and the feature of the image was extracted. After that, audio-based detector was generated on the detected object to notify the visually impaired people about the identified object. The result was evaluated by using mean average precision (mAP) and frame-per-second and it was found that the Single Shot MultiBox Detector (SSD) reduces the complexity and achieves higher accuracy as well as faster speed in object detection compared to Fast R-CNN.**

***Index Terms*—Blind, Cloud, Object Detection, Object Recognition, Image Processing.**

I. Introduction

The purpose of this paper is to implement convolutional neural network (CNN) for image recognition application. CNN has the advantage over conventional approaches since CNN is able to extract features individually, decrease dimensional of the data as well as the category in one network structure. As a result, CNN uses relatively little computational load compared to other conventional algorithms. Hence, many people choose CNN as their choice for overcoming the image classification problems [1, 2]. The purpose of object detection was to recognize the objects from the scene and predict the corresponding bounding boxes. The image data were trained by using CNN algorithm. A popular open source deep learning framework, called Caffe was used to build the classifier. The trained Caffe model was stored in the Microsoft Azure cloud storage database. The system was put on the cloud platform so that the trained caffe model could be retrieved; hence, performing the image recognition process anywhere and anytime.

In this paper, the complexity of RCNN approach in object detection has been reduced. As shown in Figure 1, an edge box was chosen to generate region proposals rather than the selective search used in RCNN [3]. Generally, the edge box produces an outcome of the proposal based on the edge map of the image. Edge boxes can achieve much faster speed over selective search in RCNN [4]. The average runtime was 0.3 seconds for edge box, while 10 seconds for selective search. Therefore, the edge boxes are able to decrease the computational load with excellent performance [5]. Each pixel carries a magnitude and orientation data of the edge in edge detector. An edge map is produced with a structured edge detector. Although the mean average precision (mAP) for both methods were almost similar, edge boxes take shorter time to process compared to selective search. In addition, all of the classes specific to Support Vector Machine (SVM) were removed and the output of SoftMax was used [6].

In this paper, an intelligent object detection system for blind people based on CNN has been developed to achieve safer and better quality of life. Video scene in real time was taken in the object detection systems and activities of interest was determined by computer vision-based techniques, in which the primary task was to produce a high-level understanding of the imaged scene and to generate application-specific data to be used in an autonomous and intelligent system. Figure 1 shows the model used in this project.

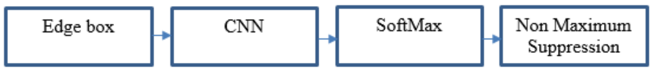


Figure 1: The overview of the model used

The aim of this paper was to present a smart living for the blind people by assisting them in their daily life using object detection system. The detection system will first detect the object through the camera in real time. The video will then be processed by making comparison with the trained models in the cloud database.

II. Background

Recently, visual recognition has become a popular topic and gained interest by many researchers. In the past, the most successful features applied into multiple classes and object detection in a single image were such as, scale-invariant feature transform (SIFT) and histogram of oriented gradients (HOG) combined with SVM [7]. Due to the poor performance of the previous models, people have started to use CNN in object detection. The utilization of a deep CNN method allows for significant accuracy [8].

CNN has become an interesting topic in visual recognition since 2012 because it is able to classify the images with high accuracy. It has become the best method in solving image classification problems in visual recognition and object detection. The connections of CNN between the neuron networks lead to numeric weights that are tuned during the training process. These weights allow for a properly trained network to respond correctly when an image or pattern is presented for recognition. The network consists of multiple layers of feature-detecting "neurons". Each layer has many neurons that respond to different combinations of inputs from the previous layers. The layers are built up so that the first layer detects a set of primitive patterns in the input, followed by the second layer that detects patterns of the patterns, and the third layer that detects patterns of those patterns and so on.

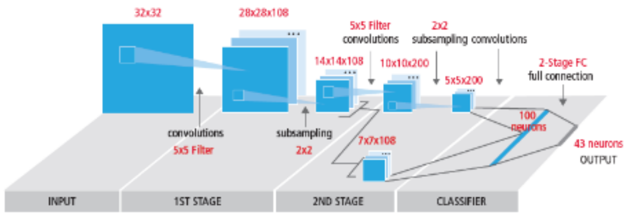


Figure 2: Block diagram

While the feature extractors are hand designed in the conventional algorithm [9], the weights of the convolutional layer in CNN is used for feature extraction and the fully connected layer is used for classification during the training process. As shown in Figure 2, the improved network structures of CNN results in saving the memory requirements and computation complexity requirements as well as giving better performance for applications with input that has local correlation. The advantages of using CNN for image recognition are its ruggedness to shifts and distortion in the image, fewer and faster memory requirements, and better classification training.

III. METHODOLOGY

The paper started with training the image, importing the image to the cloud storage, developing a webpage, and implementing object recognition on the board, namely the Raspberry pi 3. The system was running on the board with the camera used to capture the scene that the blind people encounter in their daily life. The captured image was then compared with the trained image in the database. After that the system generated a voice synthesis of the detected object to assist the blind people to identify the object easily. The steps in developing this system and software are as shown in Figure 3 and Figure 4 respectively.

A. *Obtaining the image dataset*

A standardized dataset called CIFAR-10 [10] has been used. The CIFAR-10 dataset consists of 60000 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images: The example of the dataset is as shown in Figure 5. The dataset has been divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. Each image has a size of only 32 by 32 pixels. The small size makes it sometimes difficult for humans to recognize the correct category, but it simplifies things for our computer model and reduces the computational load required to analyze the images.

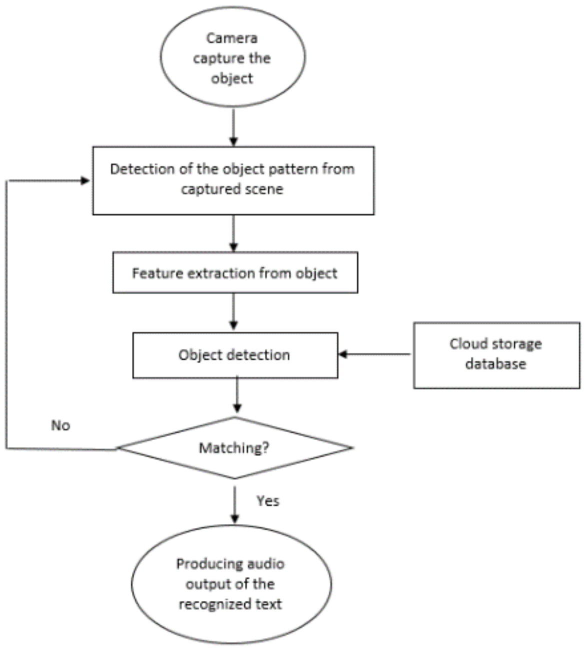

Camera capture the object
Detection of the object pattern from captured scene
Feature extraction from object
Object detection
Cloud storage database
No
Matching?
Yes
Producing audio output of the recognized text

Figure 3: Flowchart of the system

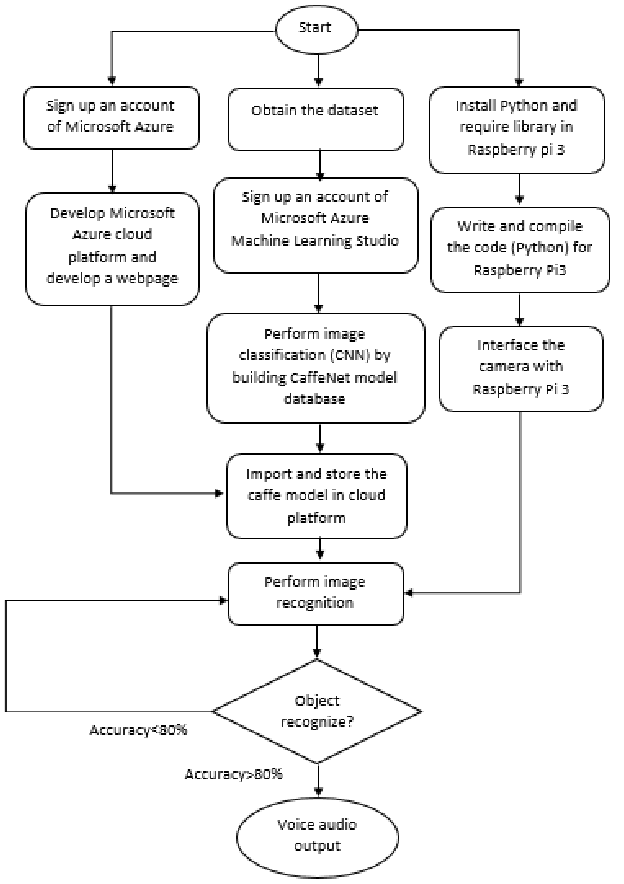

Start
Sign up an account of Microsoft Azure
Obtain the dataset
Install Python and require library in Raspberry pi 3
Develop Microsoft Azure cloud platform and develop a webpage
Sign up an account of Microsoft Azure Machine Learning Studio
Write and compile the code (Python) for Raspberry Pi3
Perform image classification (CNN) by building CaffeNet model database
Interface the camera with Raspberry Pi 3
Import and store the caffe model in cloud platform
Perform image recognition
Object recognize?
Accuracy<80%
Accuracy>80%
Voice audio output

Figure 4: Flowchart of software

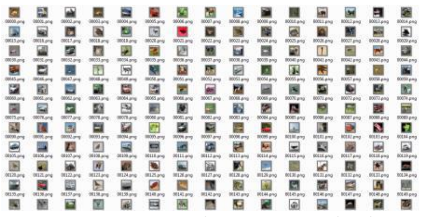


Figure 5: Example of train dataset of Cifar-10

B. Building Models Database

To apply the machine learning, all objects that exist in the environment of the blind people have been manually extracted and identified by the user. A database of the models has been created by first converting the dataset to lmdb database. The files symlinks and labelmap.prototxt have been created in the current directory. To generate the required training prototext, gen_model.sh has been used to generate the deployed Caffe model and the result has been evaluated.

In the CNN process, the image was first broken into overlapping image tiles. The entire original image was broken as a separate tiny image tiles. Secondly, each of the image tile was fed into a small neural network. The same neural network weight for every single tile was kept in the same original image, as shown in Figure 6.

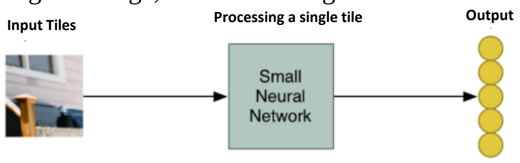


Figure 6: An input tile processed through neural network

Then, as shown in Figure 7, the result from each tile was saved into a new array. The result from processing each tile was saved into a grid in the same arrangement as the original image. The processed output resulted in a slightly smaller array that records which sections of the original image are the most interesting to be detected.

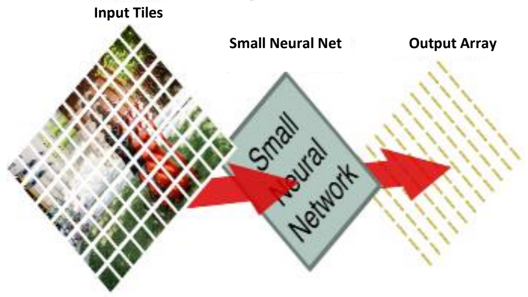


Figure 7: The tile has been saved to a new array

The next step was the down-sampling process. The size of the array was reduced by using an algorithm called max pooling, which kept the most important bits only, as shown in Figure 8.

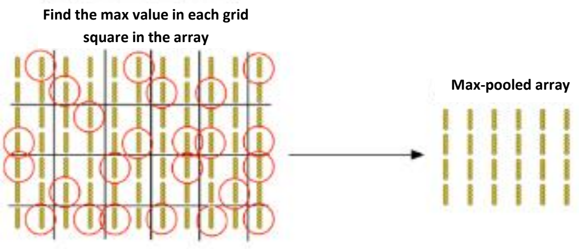


Figure 8: Max pooling (down sampling)

A prediction was made as the final step in CNN, as shown in Figure 9. The small array was used as input into another neural network. The final neural network was used to decide whether or not an image is matched.

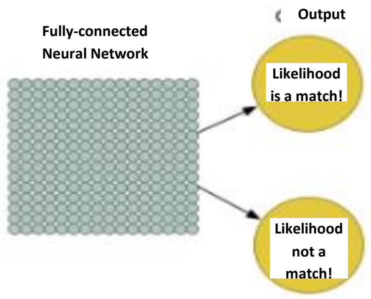
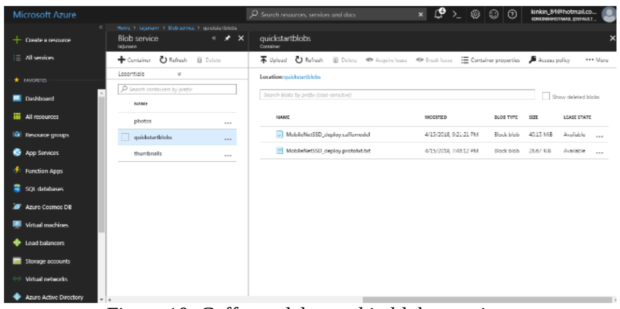


Figure 9: Decision making process

C. Developing cloud storage webpage

A storage account was created to store the trained caffe model, which include the MobileNetSSD_deploy.caffe model and MobileNetSSD_deploy.prototxt.txt in the cloud storage, as shown in Figure 10. Python code was used to upload the trained caffe model and stored in the blob container. In addition, the built Caffe model was saved in the cloud storage to allow the system to retrieve the model and perform object recognition anywhere and anytime.



NAME	MODIFIED	BLOB TYPE	SIZE	LEASE STATE
MobileNetSSD_deploy.caffemodel	4/15/2018, 9:21:21 PM	Block blob	40.15 MiB	Available
MobileNetSSD_deploy.prototxt.txt	4/15/2018, 7:48:12 PM	Block blob	28.67 KiB	Available

Figure 10: Caffe model stored in blob container

D. Embedded application of CNN on Raspberry Pi

The real time object detection code was compiled on the Raspberry Pi 3 using Python 3.5. All the necessary packages library such as numpy, cv2, pyttsx, imutils and so on were imported to the Raspberry Pi 3. Next, the argument parse was constructed and the list of the class labels MobileNet SSD was trained for initialization and subsequently, a set of bounding box colors for each class was generated. In addition, the serialized model was loaded from the disk and the video stream was initialized to allow the camera sensor to warm up as well as to initialize the frame-per-second counter. The frames from the video stream were resized to a maximum width of 40 pixels. The frame dimensions were grabbed and converted to a blob. The blob was passed through the network and the detection as well as the prediction was obtained. Moreover, the detections were looped over to extract the confidence such as probability. The weak detections were filtered out by ensuring the confidence is greater than the minimum confidence. The index of the class label was extracted from the detections and the coordinates of the bounding box for the objects were computed as well as the prediction on the frame is drew. If the system was terminated by user, the loop will break and the timer will stop by displaying the frame-per-second information.

E. Developing Raspberry Pi prototype

Figure 11 shows the prototype, which consists of the Raspberry Pi 3 as the microcontroller to perform the object detection and recognition as Raspberry Pi 3 consumed less power compared to the laptop since the system is portable for the blind people so that they can bring it everywhere. The power source for the Raspberry Pi can be replaced by a power bank, in which the power input for the Raspberry Pi 3 is 5V. The python code was run in the Raspberry Pi 3 to perform object detection process, while the webcam was used to capture the surrounding scene. The headphone was connected to the 3.5 audio jack of the Raspberry Pi 3 to allow the blind people to listen to the system on the detected objects.

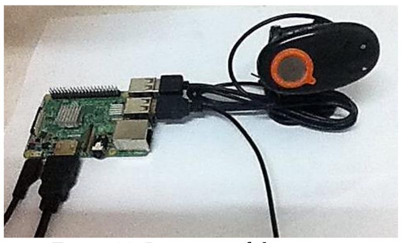


Figure 11: Prototype of this project

F. Functionality of the system

In the proposed system, a webcam was used to capture the scene that the blind people are facing every day. For object detection, after the webcam captured the scene object, the feature was extracted and compared to the database. If the object matched with the database, a speech was generated by the system to assist the visually impaired people to identify the object. This prototype was able to recognize visual objects and presented the detection information as sound. A webcam was used to capture the object and the microcontroller was used to perform the object detection and recognition by retrieving the Caffe model that uses CNN algorithm to train images. The trained Caffe model is able to retrieve the sound-based visuals from the Microsoft Azure cloud storage through WiFi. The system could still perform the object detection under offline condition if the previous Caffe model has been downloaded and stored in the local storage of Raspberry Pi 3. This system is easy to operate and it is portable, that is it is user friendly.

IV. RESULT

The experimental results show that the proposed approach notifies a visual disable people about the surrounding environment such as persons, chairs, doors, tables, or screens. In the test scenario, the system was tested with common objects like chair, person, and bottle, as shown in Figure 12 and Figure 13. During the testing period, we firstly generated regional proposals by edge boxes, and then performed the forward pass for all the proposals through the fine-tuned CNN.

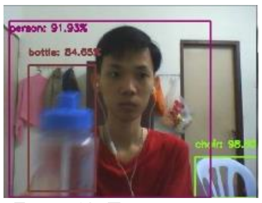


Figure 12: Test scenario

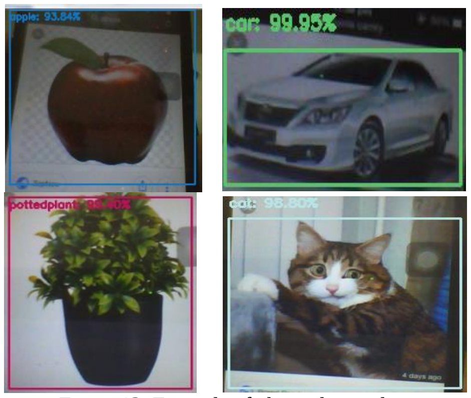


Figure 13: Example of object detected

A. Accuracy

After the image was trained, the quality of the model was measured using different criteria, such as precision, recall, accuracy, area-under-curve, and many others. The mean Average Precision (mAP) was computed by taking the average over the APs of all the classes. For a given task and class, the precision curve was computed from a method's ranked output. Recall is defined as the proportion of all positive examples ranked above a given rank. Precision is the proportion of all examples above that rank, which are from the positive class.

$$AP = \frac{1}{11} \sum_{r€\{0,0.1 \ldots,1\}} Pinterp\ (r) \tag{1}$$

$$Pinterp(r) = \max p\ (\tilde{r}) \tag{2}$$

The precision at each recall level r was interpolated by taking the maximum precision measured for a method for which the corresponding recall exceeds r, where p(˜r) is the measured precision at recall ˜r. The intention in interpolating the precision/recall curve in this way was to reduce the impact of the "wiggles" in the precision and recall curve, caused by small variations in the ranking of examples. It should be noted that to obtain a high score, a method must have a precision at all levels of recall, this penalizes methods, which retrieve only a subset of examples with high precision like side views of cars.

B. Fast R-CNN

The faster R-CNN used the CNN feature extractor to extract image features. Then, it used a CNN region proposal network to create region of interests (RoIs). RoI pooling was applied to warp them into fixed dimension and then fed into fully connected layers to make classification and boundary box prediction [11]. The Faster R-CNN built all the ground works for feature extractions and ROI proposals. Next, more convolution layers were added to build the mask. The class non-maximum suppression (nms) was performed to group highly-overlapped boxes for the same class and to select the most confidence prediction only.

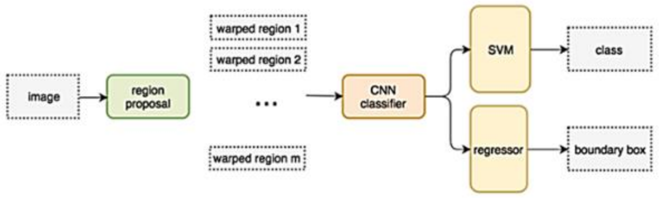


Figure 14: System flow of RCNN

Figure 14 shows the overall system flow of RCNN. Region proposal method was used to create regions of interest for object detection. In the selective search, it started with each individual pixel was considered as its own group. Next, the texture for each group was calculated and the two closest groups were combined. Grouping smaller group is preferred to avoid a single region in gobbling the others. These regions were merged until everything was combined together. Some of the feature maps used by the RPN, some of the corner locations of the boundary boxes and the distributions for the offsets from the anchors are shown in Figure 15, 16 and 17 respectively.

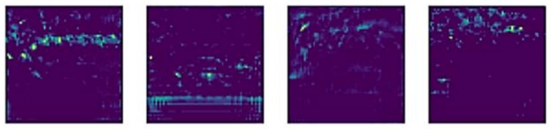


Figure 15: Some feature maps for the RPN

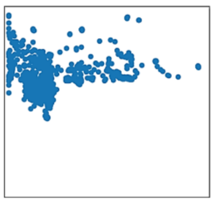


Figure 16: The corner locations of the boundary boxes

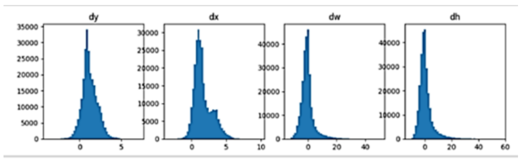


Figure 17: The distributions for the offsets from the anchors

C. YOLO

YOLO is an object detection system targeted for real-time processing. For the grid cell of YOLO, the input image was split into an S×S grid, as shown in Figure 18. Each grid cell predicts only one object and a fixed number of boundary boxes [12].

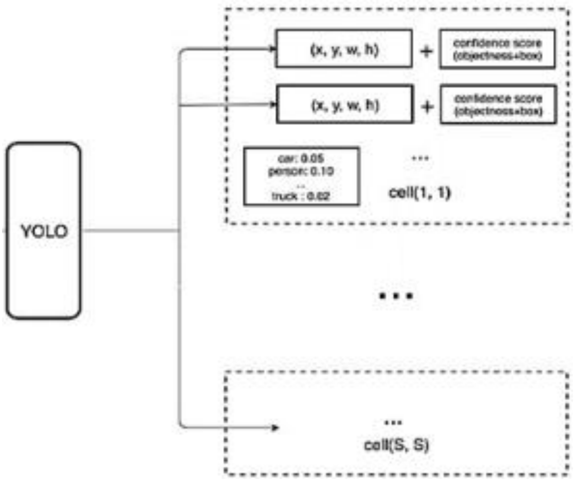


Figure 18: YOLO make SxS predictions with B boundary boxes

Each boundary box contains 5 elements. The confidence score reflects how likely the box contains an object and how accurate is the boundary box. The bounding box width, w and height, h is normalized by the image width and height, x and y are offset to the corresponding cell. Hence, the elements x, y, w and h are all between 0 and 1. Each cell has 20 conditional class probabilities. The conditional class probability is the probability that the detected object belongs to a particular class. The basic concept of YOLO is to build a CNN network to predict a tensor. It uses a CNN network to reduce the spatial dimension with 1024 output channels at each location. YOLO performes a linear regression using two fully connected layers to make the boundary box predictions. Convolution layers decreases the spatial dimension gradually and it is difficult to detect small objects as the corresponding resolution decreased. Other object detectors, like SSD locate the objects from different layers of feature maps. So, each layer specializes at a different scale. YOLO adopted a different approach, called passthrough. For example, it reshapes the 28 × 28 × 512 layer to 14 × 14 × 2048. Then, it concatenates with the original 14 × 14 ×1024 output layer. We applied the convolution filters on the new 14 × 14 × 3072 layer to make predictions. After the fully connected layers were removed, YOLO can take images of different sizes.

D. SSD

SSD are designed for object detection in real-time. Since Faster R-CNN utilizes a region proposal network to create boundary boxes and utilizes those boxes to classify objects, the SSD speeds up the process by eliminating the need of the region proposal network. Besides, SSD applies a few improvements including multi-scale features and default boxes to recover the drop in accuracy. These improvements allow SSD to match the accuracy of the Faster R-CNN using lower resolution images, which further pushes the speed higher. SSD uses VGG16 to extract feature maps. Then, it detects objects using the Conv4_3 layer. Each prediction composes of a boundary box and 20 scores for each class, and we picked the highest score as the class for the bounded object. Conv4_3 makes a total of 38 × 38 × 4 predictions, which is four predictions per cell regardless the depth of the feature maps. As expected, many predictions contained no object. SSD reserves a class "0" to indicate it has no objects.

E. Result

Based on Table 1, we can conclude that SSD has the highest mean Average Precision compared to the other two. Although the frame-per-second of SSD was low, the speed on image detection for this application did not require in high speed to detect the object. Hence, SSD was used as the model of this project.

Table 1
mAP and FPS for each model

Model	Mean Average Precision (mAP)	Frame-per-second(FPS)
Faster R-CNN	73.2	7
YOLO	63.4	45
SSD	73.7	5

F. Energy and performance evaluation comparison

Performance evaluation of different devices was discussed, and energy efficient computing was measured in this section, as shown in Table 2. The Raspberry Pi requires 5V to boot-up with 2.25 watt while laptop generally runs on 50 watts. Energy efficiency is one of the important factors that may affect power savings at national level. The energy consumption for Raspberry Pi and laptop are 8.1 kJ/h and 180 kJ/h respectively. On other hand, portability also plays an important role as the Raspberry Pi is just a credit card size compared to the briefcase size of laptop.

Table 2
Comparison on energy and performance

Device	Energy efficiency (kJ/h)	Power (watt)	Portability
Raspberry Pi	8.1	2.25	Credit card size
Laptop	180	50	Briefcase size

Table 3 shows the time required to process the same CPU benchmark on the Raspberry Pi and the Intel core i5 [13]. The Raspberry Pi was predictably beaten by even the nine-year-old AMD Athlon processor, but considering its functionality from the factor and power usage, it is a remarkably versatile little system on a chip. The Raspberry has only 1GB RAM, causing it to take a longer time to process, but the energy consumption is much lower compared to laptop, which runs on Intel core i5.

Table 3
Comparison on Processing Time

Processor	Kernel	Total time
Raspberry Pi	Linux raspberrypi 4.4.34+ #930 Wed Nov 23 15:12:30 GMT 2016 armv6l GNU/Linux	1341.2s
Intel core i5	Darwin Io.local 16.1.0 Darwin Kernel Version 16.1.0: Thu Oct 13 21:26:57 PDT 2016; root:xnu-3789.21.3~60/RELEASE_X86_64 x86_64	8.8698s

V. DISCUSSION

After comparing these three models that were used to train the image, SSD model is chosen as our model for this object detection system since SSD model was able to achieve the highest mAP compared to other model, as shown in the result section. There are several advantages of edge box used in SSD over the selectively search in RCNN. The basic idea of edge boxes is that the algorithm generates scores for the proposal based on the edge map of the image. The performance was not affected even the processing time for the edge box was reduced. An edge map was generated with a structured edge detector, in which each pixel contains a magnitude and orientation information of the edge. Secondly, all of the classes specific to SVMs were removed, and the output of softmax was directly used in the last layer of CNN as the score. To compensate the possible performance, degrade raised by removing SVMs, the training data to fine-tune CNN has been designed.

The prototype that we built was able to recognize visual objects and presented the detection information as sound. However, the prototype suffered from the following limitations. The user usually focused on the object from a far distance as well as to get the direction. In this work, distance of the object from approximately 15cm away from the user, which impose a very high requirement in designing the object detection model. This problem can be overcome by training the raw input images with larger scale ranges. The object detection models faced difficulty in classifying the object from a picture of ultimate scale. On the other hand, object tracking algorithm can be used to track the object, when the user has recognized the object. In future, these two methods can be implemented in the object detection application.

VI. CONCLUSION AND FUTURE WORK

In this paper, an object detection system for visual disable people based on CNN in real time has been proposed. The complexity of selective search in the Fast R-CNN has been reduced by using edge box in SSD. This leads to faster runtime and reduced time complexity without degrading the performance. The system has retrieved the trained model from the cloud database to perform object detection in real time. The proposed system is beneficial for the visual impaired people for better living quality.

ACKNOWLEDGMENT

The authors acknowledge the technical support by Universiti Teknikal Malaysia Melaka (UTeM).

REFERENCES

[1] T. Guo, J. Dong, H. Li and Y. Gao, "Simple Convolutional Neural Network on Image Classification," IEEE 2nd International Conference on Big Data Analysis (ICBDA), 2017, pp. 721-724.
[2] Y. C. Wong, Y. Q. Lee, " Design and Development of Deep Learning Convolutional Neural Network on an Field Programmable Gate Array, Journal of Telecommunication, Electronic and Computer Engineering, 2018, pp. 25-29.
[3] A. Rosebrock, "Real-time object detection with deep learning and OpenCV", [Online]. Availalable: https://www.pyimagesearch.com /2017/09/18/real -time-object-detection-with-deep-learning-and-opencv/. [Accessed: 16-December-2018].
[4] P. Viola, M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", Computer Vision and Pattern Recognition, 2001, pp. 1-9.
[5] S. Tang, "Object Detection based on Convolutional Neural Network," [Online]. Available: https://www.semanticscholar.org/paper/Object-Detection-based-on-Convolutional-Neural-Tang/1c64a84b1993cf7df6b3553c677c1011c448530a. [Accessed: 16-April-2019]
[6] T. Guo, J. Dong, H. Li and Y. Gao, "Simple Convolutional Neural Network on Image Classification," IEEE 2nd International Conference on Big Data Analysis (ICBDA), 2017, pp. 721-724.
[7] I. M. Gorovyi and D. S. Sharapov, "Comparative Analysis of Convolutional Neural Networks and Support Vector Machines for Automatic Target Recognition", IEEE Microwaves, Radar and Remote Sensing Symposium (MRRS), Kiev, 2017, pp. 63-66.
[8] S. S. Liew, M. Khalil-hani, and S. A. Radzi, "Gender classification : a convolutional neural network approach", Turkish Journal of Electrical Engineering and Computer Sciences, vol. 24, no. 3, 2016, pp. 1248–1264.
[9] B. S. Hijazi, R. Kumar, C. Rowen, and I. P. Group, "Using Convolutional Neural Networks for Image Recognition", [Online]. Availalable: https://ip.cadence.com/uploads/901/cnn_wp-pdf. [Accessed: 16-April-2019]
[10] K. Alex, N. Vinod and H. Geoffrey, "CIFAR-10 and CIFAR-100 datasets", [Online]. Availalable: https://www.cs.toronto.edu/~kriz/cifar.html, [Accessed: 22-November-2018]
[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," NIPS, 2015, pp. 91–99.
[12] C. R. Joseph, "YOLO: Real-Time Object Detection", [Online]. Availalable: https://pjreddie.com/darknet/yolov1/, [Accessed: 5 January 2018]
[13] B. V. Core, "Energy Efficient Computing : A Comparison of Raspberry PI with Modern Devices Devices included Energy And Performance Evaluation Comparision : Computing Energy Efficiency : Conclusion", 2012, pp. 6–7.