Cloud-based Embedded System for Object Detection and Recognition

Z T Olalekan and S A Anwar
School of Computer and Communication Engineering, Universiti Malaysia Perlis (UniMAP), Pauh Putra Campus, 02600 Arau, Perlis, Malaysia
said@unimap.edu.my

Abstract. Object detection and recognition techniques require large image datasets, memory, a workstation with specific graphics processing capability to train the algorithm and might have high power consumption. Embedded platforms on the other hand are characterized by portability, low power consumption and space, and energy resources making the deployment of such algorithms on them difficult. In order to overcome these drawbacks, cloud-based processing embedded system for object detection and recognition is proposed in this work. The system consists of an image acquisition device set up using embedded board and camera to capture, process and send images to the remote computer via cloud storage platform. This cloud platform serves as an interface between the embedded board and the remote computer. The detection algorithm of Faster R-CNN is executed on the remote computer and is trained and validated with 3000 images obtained from ImageNet. The training of the algorithm aims to detect five classes of object. The proposed system was validated off-line and have achieved a mean Average Precision (mAP) of 0.67. The performance of entire system procedure took about 45 seconds and have obtained an average confidence score of 0.86.

1. Introduction
In recent years, computer vision domain has not been left out from thriving due to the endless efforts of researchers and significant advancement in related fields. Computer vision uses digital images to model and emulate human vision using a computer through three major steps. These steps are image acquisition, image processing with the third being image analysis and understanding. As a result of this, applications of computer vision such as pattern recognition, medical imaging, 3D model building, surveillance, object detection and recognition and face detection has been made a reality.

Object detection is basically determining instances of real-world objects in images or videos while object recognition is the identification of target objects in still images or videos. Deep learning is one of the approaches used in object detection and recognition. Although several deep learning models have been proposed for object detection, Convolutional Neural Networks (CNN) is the most popular because it produces state-of-the-art recognition results, eliminates the requirement for manual feature extraction and it can be retrained for a new recognition task [1]. CNN models learn directly from image data and classify the images using pattern thus removing the need for manual feature extraction. They are used to recognize objects, scenes, and faces. CNN can be used to visually detect cancer cell, enable autonomous vehicles to detect objects [2].

The region proposal methods and region-based CNN have shown advances in object detection tasks [3]. However, this R-CNN algorithm has drawbacks where it was found to be slow in execution time due to its heavy computation. It was later overcome by the introduction of Fast R-CNN algorithm
The algorithm was very recently improved by merging the Region Proposal Network (RPN) with Fast R-CNN into a single network by sharing their convolutional features [5].

Notwithstanding the improvement in terms of object detection accuracy, the implementation of Faster R-CNN still involves a lot of computational intensity. The general-purpose computer with only central processing unit (CPU) becomes heavily loaded and more often cannot achieve the real-time requirements when implementing the algorithm except when equipped with graphics processing unit (GPU) [6]. With the recent trends towards portable and mobile devices, the issue of power supply, memory size and processing capability becomes a major problem if the detection algorithm is about to be implemented such devices. These issues are extended to the use of embedded or single board computer system.

The availability of embedded platforms which are reliable, portable, easy to use, have low energy consumption with lower prices has increased [7]. These merits make the embedded system suitable for the role of acquiring images. However, as mention earlier, due to its limitation in terms of processing capability and memory resource, the implementation of object detection algorithm such as Faster R-CNN is almost impossible. Hence, a new strategy to incorporate the use of embedded board for image acquisition and the benefits of the accuracy of object detection algorithm has been proposed. This involved utilizing high speed of the internet connection and the availability of access to the cloud storage facility.

In this paper, the implementation of cloud-based processing embedded system for object detection and recognition is presented. The front-end system utilizes the commercially available embedded system board Rasberry Pi 3 Model B with the camera module attached for image acquisition and pre-processing. On the other end is a remote computer to execute the Faster R-CNN algorithm and perform the object detection and recognition tasks. Both elements in the system are connected to the internet while the cloud storage service was used as a medium for uploading and downloading the images.

The rest of this paper is organized as follows. In Section 2, related previous works on the detection and recognition of objects with their implementation on embedded platforms were presented. In the next section, the implementation of the system is described and follows by the brief description of Faster R-CNN algorithm. This section also includes the explanation of the datasets and quality metrics used in this work. In Section 4, experimental results and discussion are presented to report the performance of the proposed system. Finally, the conclusion is presented.

2. Related Works

This section presents the related works previously done in terms of the implementation of the deep learning algorithm such as CNN on the embedded platforms or general-purpose computers. These works are aims to solved various problems and applications.

Shiddieqy et al. [8] implemented CNN using Python programming language and TensorFlow on the Raspberry Pi 3 to classify images (Dogs and Cats). They limited the number of ConvNet layers in the neural network to two and five. The model was trained on a computer with i7 7700 Quad-core 3.6GHz, 32GB RAM, 256GB SATA SSD and 8GB Nvidia GTX 1080 GPU. The accuracy of the two layers and five layers network are 55.6% and 77% respectively. The trained model was tested on 100 images on the i7 7700 PC and Raspberry Pi. It is worthy to note that the time recorded was only the time in the CNN model. On the Raspberry Pi, 13.699ms and 31.479ms were gotten for the best and worst run while the duration of the best run and worst run on the i7 7700. Although the layers of the neural network were greatly reduced, the processor usage on the Raspberry Pi was 100% while the PC was 17%. Moreover, a five-layer CNN is not enough for real-time applications as the more the layers of a neural network, the better the accuracy of the network's prediction.

Faster R-CNN algorithm was used, trained and tested on a computer (64-bit, Windows 7, 4GB Nvidia Quadro K2200) using which is used to automatically recognize books and their positions [9]. The database consists of 10 classes and exceed 4000 images in total. About 1000 images which were selected randomly was used to test the detector performance in terms of speed and accuracy. Testing
of 1000 images took about 190.85 seconds and the classification accuracy of the network was 98.6%. In this work, the detector was not implemented on an embedded system as all stages (training and testing) was done on a general-purpose computer.

Xu et al. [10] introduced an application of depth learning in commercial video analysis and accelerates the embedded depths of learning algorithm model. They implemented a CNN for face attribute recognition based on a pre-existing model GoogleNet and added the structure of inception (pre-existing CNN model) to increase network's depth and width. They have deployed two independent networks for the identification of age and gender. Adience dataset was used which consists of images that were automatically uploaded from smartphones. The accuracy of 95% and 96% were achieved on age and gender classification respectively. The embedded hardware used was NVIDIA Jetson TX1 GPU alongside TensorRT optimization library to achieve a dual model with a speed of 15 frames per second when analysing commercial videos.

On another application, a real-time vehicle detection using deep learning scheme was proposed [11]. This algorithm was implemented on an embedded system in real-time. The detection system has two classes (vehicle and non-vehicle). The authors modified an already existing Integral Channel Feature (ICF) for the detection. The classifier used was Adaptive boosting (Adaboost) to generate strong detectors by linearly combining weak weighted detectors. In order to reduce the rate of false detection, a model which was generated by deep learning was applied to the result obtained from AdaBoost. The processing time was approximately 70ms on Freescale's I.MX6Q embedded board. The accuracy was 92.3% after the validation phase. The deep learning model was used only to validate.

Lee et al. [12] implemented a built-in system with Nvidia Jetson TX1 board to recognize a vehicle number plate. The plate recognition was based on AlexNet (pre-existing CNN model). AlexNet was trained on 500 vehicle license plate images each having a size of 1392x1040. About 20% of the image along the vertical and horizontal plate was removed with the remainder taken as the region of interest. The numbers and letters were recognized using DIGITS. The system was tested on the embedded platform which was equipped with a CCTV camera installed on the road. 63 images were used at the testing phase and a recognition accuracy realized was 95.24%. Although the system was able to identify the numbers that exist in the region of interest, images with reflective or broken plate number were not recognized. The processing time was not stated.

It can be inferred that in most cases, the models are implemented on general-purpose computers with the GPU acting as accelerators when training the neural networks. For actual implementation on the embedded board, the network layer is reduced which is most likely not enough for real-time applications. In the case of the main algorithm that is running on embedded platform, there are no effort were reported in terms of outsourcing the heavy computational works to the more powerful system through any means of cloud usage.

3. Methodology
In this section, the proposed embedded system with the cloud-based processing ability is presented. It consists of the system overview where the hardware elements and the flow of the processes involved are described in detail. Then, the object detection and recognition algorithm featuring Faster R-CNN is briefly described. Next, the description of standard image dataset and the quality metrics used to evaluate the system are also presented.

3.1. System Implementation
The proposed cloud-based object detection and recognition embedded system are shown on figure 1. It consists of the embedded board Raspberry Pi with the camera attached for image acquisition and preparation at the front-end and the computer at the backend. Both of these elements are connected to the internet and have access to the cloud-storage where the images are kept during the processing.
Figure 1. Proposed object detection and recognition system

The specification of the embedded board and computer involves in this system are tabulated in table 1. The Raspberry Pi Camera Module is connected to the embedded board has 5MP native resolution with a sensor capability of 2592x1994x3 pixels. The computer has the technical computing software package MATLAB 2019A installed. Meanwhile, the embedded board is programmed using Phyton programming language. The cloud storage used in this work is Dropbox with the capacity of 5GB.

Table 1. Specification of the system.

| Item              | Raspberry Pi Model 3B                  | Remote Computer          |
|-------------------|---------------------------------------|--------------------------|
| Processor         | 64-bit Quad Core, 1.2GHz               | 64-bit Core i5, 2.50GHz   |
| Memory            | 1GB RAM                               | 12GB RAM                 |
| Storage           | 32GB SD                                | 1TB HDD                  |
| Connectivity      | WiFi on board                         | WiFi                     |
| Operating System  | Raspbian Stretch                      | Windows 10               |

In general, the proposed work has four processes which are image acquisition, image preparation and transmission, deployment of Faster R-CNN algorithm (briefly explained in section 3.2) and lastly, the display of the detected object. Image acquisition and preparation are executed on the embedded board and the image later is transmitted through internet and stored in the cloud storage. After this process, the image is download by the remote computer and the Faster R-CNN algorithm is performed to detect and recognize the objects in the image. Later, the resulted output image is uploaded back to the cloud storage where it is available for embedded board to download and display them. Figure 2 shows each process implemented in the system and which element is involved.

Figure 2. Flow of the process involved.
3.2. Object Detection and Recognition

In this work, the object detection and recognition tasks are performed on the remote computer using Faster Region-based Convolutional Neural Network (Faster R-CNN) [5]. This algorithm is trained on the remote computer by using MATLAB 2019A software package. Training is done by feeding the network myriads of already labelled images as input. The network then learns features of the desired objects in the images which in turn allows it to find objects of the same class in unlabelled images. The flowchart in figure 3 shows the implementation of this algorithm.

As can be seen from figure 3, in the Region Proposal Network (RPN), image is fed to CNN for feature extraction and anchor boxes are generated. These input image can be labelled (in the case of training) or unlabelled and must at least be of size 224x224 pixels because it is the minimum size that the base network can accept. The extracted features and anchors are sent to the region proposal layer where the probability of it being a background or foreground is predicted.

Non-maximum suppression is executed on the generated proposals to eliminate duplicate proposals. Based the foreground probability, the top 2000 proposals are sent to the Region of Interest (ROI) pooling layer where the fixed-sized feature maps for each region proposal is extracted in order to classify the proposals into one of the object classes and also readjust the bounding box in accordance to the predicted class.

In this work, options like maximum number of times an entire dataset is passed through a neural network, mini batch size and initial learn rate are configured to 10, 1 and 0.001 respectively. The mini batch size is set to 1 because multiple image regions are processed from one training image on every iteration. Initial learn rate is the amount that the weights are updated during training and if the value is set too low, the training takes longer time to complete and if it is too high, the result might be suboptimal. In addition, a checkpoint path is set to a temporary location thereby saving the partially trained detector during the process.

3.3. Datasets

The dataset containing 2500 images is used for the training of the Faster R-CNN algorithm and was collected from ImageNet (http://imagenet.stanford.edu/). The collected images were of the 8-bit RGB colour scheme and in various sizes ranging from 110x110 to 1036x776 pixels. This dataset contains five classes of objects which are Broom, Fan, Television, Keyboard and Mouse. Each of the classes has 500 images.

Objects in the training images were labelled by drawing bounding boxes over them with an aid of the MATLAB Image Labeller application. The resulting ground truth file which was exported from MATLAB played a crucial role while training the object detection algorithm. For training purposes, all images are resized to 800x600x3 (SVGA resolution) in order to reduce the sampling point hence speeding up the training process.

After training, the algorithm is validated using another 500 independence images from the ImageNet where 100 images are allocated for each of the classes. Lastly, for the testing images, camera on the embedded board was used to capture the real scene. The scenery chosen to be captured as testing image must contains at least one out of five objects described earlier. Similar to the training images, 20 captured images were then resized to 800x600x3 before being uploaded to the cloud storage.
3.4. Quality Metrics

Mean Average Precision (mAP) is a metric used to measure the accuracy of object detection algorithm. The ability of the detection process to find the relevant objects only is a measures precision of the algorithm. Sometimes this can also be defined as the percentage of correct predictions. In this case, Intersection-over-Union (IoU) is the precision measure of the object detection algorithm used in this work [12] is given below

$$\text{IoU} = \frac{A_o}{A_{u}} \quad (1)$$

where $A_o$ and $A_u$ is area of overlap and union respectively. It measures the overlap between two bounding boxes (i.e. the predicted box and the ground truth) and a prediction is correct when IoU $\geq 0.5$.

In order to compute the mAP for a class, average IoU of all classes is determined and the mean of the IoU averages is calculated. The average precision can also be considered as the area beneath the
precision-recall curve. Recall is the ability of algorithm to identify all relevant (i.e. the percentage of predicted true positive out of all positive instances based on the ground truth). Equations (2) and (3) shows the mathematical definition of precision $P$ and recall $R$.

$$P = \frac{tp}{tp + fp} \tag{2}$$

$$R = \frac{tp}{tp + fn} \tag{3}$$

where $tp$ (true positive) is a correct prediction, $fp$ (false positive) is a wrong detection (i.e. predicted positive when it is negative) and $fn$ (false negative) is when an object is predicted as negative when it actually is positive.

4. Result and discussion
In this section, the experimental results from the methods and procedures described in the above sections are presented. It consists of two parts: offline is where the object detection algorithm is validated and online is when the system is test in real-time.

4.1. Offline validation
For the purpose of evaluating the algorithm, 500 ground truth images were fed to the detector. The result from the detector and the available ground truth was compared. The mAP is computed for each image and the precision-recall graph is derived for each object class as shown in figure 4.

![Figure 4. mean Average Precision (mAP) of the object detection algorithm.](image)

As can be seen, the lowest Average Precision (aP) is 0.39 which is for the Broom class. This is translated into only 39% of the data point are relevant out of the total detected object. Meanwhile, the Keyboard class scores the highest aP at 0.89 follows by the Television class at 0.84. These two classes have more than 80% of their detected points in the dataset are relevant. However, upon closer
examination, the precision for both classes start to reduce at 0.24 and 0.58 respectively. In terms of spans of precision value, the Television class have tighter range between 0.87 to 1 compared to the Keyboard class which is between 0.68 to 1.

Overall, figure 4 shows the mean average precision (mAP) of the multi-class detection. Since the mAP is 0.67, it can be deduced that 67% of objects detected in that dataset is significant. This shows that the detector locates one-third of the objects that are irrelevant as relevant.

4.2. Online testing
In order to test the system online, 20 images some of which contains multiple objects were captured and processed. It is important to note that the program on the embedded board pauses for 30 seconds to prevent error while it waits for the output image to be uploaded to cloud. In addition, the network strength also plays an important role as a weak network signal will result in an increase in the process duration.

Figure 5 shows the first 9 output images from the online testing. The bounding box is drawn by the detection algorithm to highlight the object found as well as the class and the confidence score. The result also shows that the algorithm can detect more than one object in a single image as can be seen in figure 5 (g) and (i).

![Figure 5. Example of detected objects in real scene.](image)

In addition, table 2 tabulated the confidence score and the execution time of the system starting from the image capturing to the output display for the all testing images. The average time taken to
process these images is approximately 44.66 seconds with the lowest execution time is 43.45 seconds and the highest is 47.20 seconds. Since the buffering period of 30 seconds is artificially introduced in the software, it can, therefore, be deduced that the major processes the system take about 13 to 17 seconds.

### Table 2. System performance evaluation

| Image | Detected Object | Score  | Time Elapse (seconds) |
|-------|-----------------|--------|-----------------------|
| 001.jpg | Mouse           | 0.9785 | 45.46                 |
| 002.jpg | Fan             | 0.9161 | 44.04                 |
| 003.jpg | Keyboard        | 0.9869 | 45.48                 |
| 004.jpg | Broom           | 0.5681 | 44.00                 |
| 005.jpg | Fan             | 0.6747 | 46.05                 |
| 006.jpg | Fan             | 0.9990 | 47.20                 |
| 007.jpg | Broom           | 0.9538 | 47.13                 |
| 008.jpg | Television      | 0.9989 | 45.13                 |
| 009.jpg | Television      | 0.9553 | 44.46                 |
|          | Keyboard        | 0.9205 |                       |
| 010.jpg | Mouse           | 0.6506 | 45.24                 |
|          | Keyboard        | 0.9873 |                       |
| 011.jpg | Mouse           | 0.9990 | 44.04                 |
| 012.jpg | Television      | 0.9999 | 43.72                 |
|          | Fan             | 0.9878 |                       |
|          | Keyboard        | 0.8012 |                       |
| 013.jpg | Broom           | 0.9992 | 43.70                 |
|          | Television      | 0.9995 |                       |
|          | Keyboard        | 0.7446 |                       |
| 014.jpg | Keyboard        | 0.8243 | 43.45                 |
|          | Broom           | 0.6488 |                       |
|          | Keyboard        | 0.7246 |                       |
| 015.jpg | Keyboard        | 0.8243 | 43.84                 |
|          | Mouse           | 0.7858 |                       |
|          | Keyboard        | 0.6637 |                       |
| 016.jpg | Broom           | 0.9693 | 43.70                 |
|          | Fan             | 0.5844 |                       |
| 017.jpg | Television      | 0.8750 | 45.09                 |
|          | Keyboard        | 0.9117 |                       |
|          | Broom           | 0.7818 |                       |
| 018.jpg | Fan             | 0.9775 | 44.05                 |
| 019.jpg | Keyboard        | 0.9626 | 43.71                 |
|          | Fan             | 0.9884 |                       |
| 020.jpg | Keyboard        | 0.7246 | 43.69                 |

**Average** 0.8668 44.66

5. Conclusion

Object detection and recognition are one of the aspects of computer vision that detects objects in still or moving images by placing a bounding box around it. The goal of object detection in this paper is to be able to recognize objects in real-time and classify them accordingly without requiring expensive and high computational power. This is achieved by employing the embedded board with camera as front-end processing elements to capture and prepare the images. Utilizing the internet and the cloud storage facility, the images are processed by remote computer elsewhere. In this work, Faster R-CNN
algorithm is implemented, and this multi-class detection algorithm was trained to locate five classes of objects. The performance of the algorithm was evaluated off-line by validating the detection process on random test images and has yielded a mAP of 67% accuracy. Lastly, the system was tested on-line in real-time by capturing real-scene images and sending them cloud storage, processed by the remote computer and downloaded back into the embedded board. Total of 20 real-scene images were tested and the average confidence score produced is 0.86 with the average processing time of 45 seconds. These results suggest that an embedded system platform can have access to the effectiveness of computationally heavy algorithm running elsewhere with the proposed strategy of utilizing the cloud storage and remote computer as external resources.

References
[1] Simonyan K and Zisserman A 2014 Very deep convolutional networks for large-scale image recognition
[2] Nagaoka Y, Miyazaki T, Sugaya Y and Omachi S 2017 Text detection by Faster R-CNN with multiple region proposal networks 2017 14th IAPR Int. Conf. on Document Analysis and Recognition (ICDAR) vol 06 pp 15–20
[3] Girshick R, Donahue J, Darrell T and Malik J 2014 Rich feature hierarchies for accurate object detection and semantic segmentation 2014 IEEE Conf. on Computer Vision and Pattern Recognition pp 580–587
[4] Girshick R 2015 Fast R-CNN 2015 IEEE Int. Conf. on Computer Vision (ICCV) pp 1440–1448
[5] Ren S, He K, Girshick R and Sun J 2017 IEEE Transactions on Pattern Analysis and Machine Intelligence 39 1137–1149
[6] Lee S, Son K, Kim H and Park J 2017 Car plate recognition based on CNN using embedded system with GPU 2017 10th Int. Conf. on Human System Interactions (HSI) pp 239–241
[7] Fuente A M, Castro V G, Robles L F and Gutiérrez E A 2015 Design and implementation of an embedded system for image acquisition of inserts in a headtool machine
[8] Shiddieqy H A, Hariadi F I and Adiono T 2017 Implementation of deep-learning based image classification on single board computer 2017 Int. Symposium on Electronics and Smart Devices (ISESD) pp 133–137
[9] Zhu B, Wu X, Yang L, Shen Y and Wu L 2016 Automatic detection of books based on Faster R-CNN 2016 3rd Int. Conf. on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC) pp 8–12
[10] Xu J, Wang B, Li J, Hu C and Pan J 2017 Deep learning application based on embedded GPU 2017 First Int. Conf. on Electronics Instrumentation Information Systems (EIIS) pp 1–4
[11] Shin J, Kim U, Lee D, Park S, Oh S and Yun T 2017 Real-time vehicle detection using deep learning scheme on embedded system 2017 9th Int. Conf. on Ubiquitous and Future Networks (ICUFN) pp 272–274
[12] Rosebrock A 2017 Deep Learning for Computer Vision with Python (pyimagesearch)