

# Object Detection Techniques: Overview and Performance Comparison

Mohammed Noman

Electronic and Electrical Engineering  
University of Strathclyde  
Glasgow, United Kingdom

Vladimir Stankovic

Electronic and Electrical Engineering  
University of Strathclyde  
Glasgow, United Kingdom

Ayman Tawfik

Engineering and Information Technology  
Ajman University  
Ajman, United Arab Emirates

**Abstract**—Object detection algorithms are improving by the minute. There are many common libraries or application program interfaces (APIs) to use. The most two common ones are Microsoft Azure Cloud object detection and Google Tensorflow object detection. The first is an online-network based API, while the second is an offline machine-based API. Both have their advantages and disadvantages. A direct comparison between the most common object detection methods helps in finding the best solution for advance system integration. This paper will discuss both methods and compare them in terms of accuracy, complexity and practicality. It will show advantages and also limitations of each method, and possibilities for improvement.

**Index Terms**—Object detection, Tensorflow, Azure Cloud

## I. INTRODUCTION

Object detection methods are vast and in rapid development. There are algorithms proposed based on various computer vision and machine learning advances. The testing and compatibility of choosing the best suitable object detection method takes time. There are many factors for this choice, including the system specification and application. In this paper, common object detection algorithms will be discussed. The two most used methods will be tested and compared in their performance and features. The direct comparison between the methods will shows the best suitable method to choose when designing a new system.

The organization of this paper is as follows. In Section II we give a background of several popular object detection methods. Section III covers the background of Tensorflow and its training capabilities. Section IV covers the Microsoft Azure Cloud and its object detection API. Section V will detail the testing setup for the comparison, and section VI will shows the testing result and comparison discussion.

## II. BACKGROUND

In this section, we describe several popular object detection methods that will be used in this study.

Region-Convolutional Neural Network (R-CNN) [1] is a recent algorithm proposed to deal with the tasks of object detection, localization and classification. R-CNN is a special type of Convolutional Neural Network (CNN) that is able to locate and detect objects in images. The input image is processed to get the regions extracted. A huge number of regions are then individually warped to compute the features. Then the regions will be classified. The output is generally a

set of bounding boxes that closely matches each of the detected objects, as well as a class output for each detected object.

Some more recent improvements of the algorithm include Fast-R-CNN [2] and Faster-R-CNN [3]. Even with these improvements, there were several practical issues with R-CNN. To overcome these issues, two architectures were recently proposed: YOLO (You Only Look Once) [4] and SSD (Single Shot Multi-Box Detector) [5].

Fig. 1 shows a quick comparison between the algorithms in terms of accuracy (mAP) vs. execution speed. Faster-R-CNN has the highest raw accuracy, and YOLO has worse accuracy compared to Faster-R-CNN and SSD, while SSD is very close to R-CNN in accuracy. This makes SSD the best algorithm that balances between the speed and accuracy.

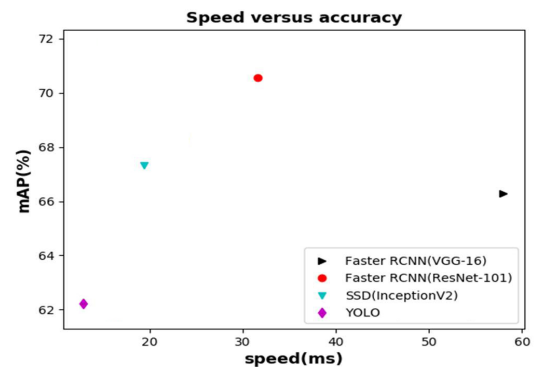


Fig. 1. Comparison between R-CNN, SSD, and YOLO in terms of speed across accuracy. Image from [6]

Due to this, the SSD algorithm is used widely in object detection systems. Tensorflow [7] object detection API is a common used powerful API tool that uses SSD algorithm and several models based on it.

## III. TENSORFLOW

The Tensorflow Object Detection API is an open source framework built on top of Tensorflow that makes it easy to train and use object detection models. Each model is called a Tensor. Tensorflow supports many platforms.

The workflow of Tensorflow Object Detection algorithm is divided into four main stages. Firstly, data images are captured, secondly the most important features are extracted, then these

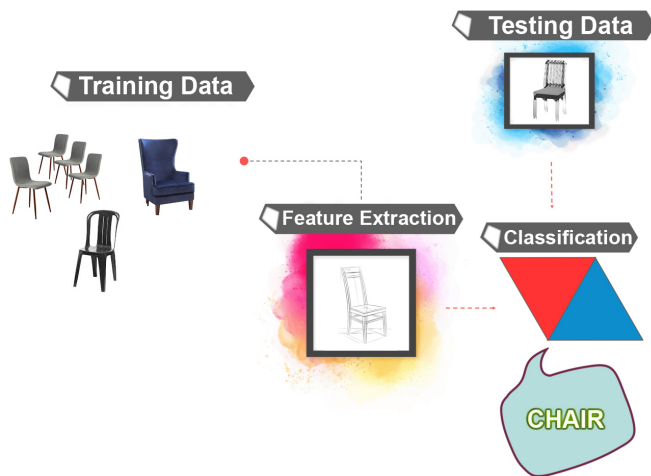


Fig. 2. Tensorflow Object Detection workflow

extracted features are classified and finally the classification models are loaded and tested. Fig. 2 shows the workflow.

Tensorflow uses many models. Common Objects in Context (COCO) [8] SSD-based model is one of them. This model is used in Tensorflow applications, as it is fast, and has a large database and easy to train for improvements.

#### A. Tensorflow Training

The object detection accuracy depends on the model used. To improve the accuracy, it is possible to train the model. Tensorflow allows using custom models by training custom data-sets [9]. The object detection library has the tools needed to export a new model or an updated model.

The Tensorflow training data-set format is TFRecord. A TFRecord file stores the data as a sequence of binary strings. To create the TFRecord file, a process must be taken. LabelImg create labels for each part of the image. After selecting the object, the user insert the label name as shown in Fig. 3. Then LabelImg exports the labels as xml files. Each image will have its corresponding xml file with the same name, for example image 3.jpeg with 3.xml. The more data is inserted, the higher the accuracy becomes but this will increase the processing time. Each data-set must have a label map. This label map defines a mapping from string class names to integer class Ids. Label maps should always start from id 1.

The final step is to run the training script to generate TFRecord file. Once the file is generated, the training tool-set needs to run the Evaluation Job. The evaluation job will periodically pool the train directory for new checkpoints and evaluate them on a test data-set. Finally, Tensorboard is used to export the Tensorflow Graph model. The model is then used as the new object detection model.

Tensorflow supports a wide range of tensor models. The use of a more complex model will yield higher accuracy results, while training the model will improve the accuracy on trained objects. This means that with a basic model it is possible to achieve high accuracy results when focused on certain objects.

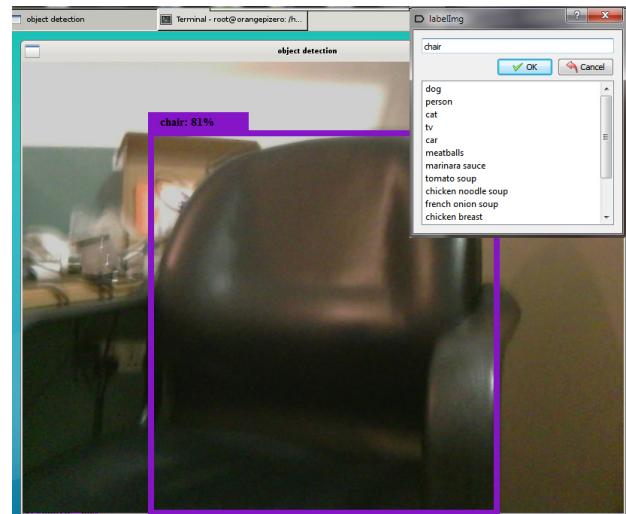


Fig. 3. Tensorflow model training process

#### IV. AZURE CLOUD

Microsoft introduced the Azure Cognitive Services for different categories. The Azure Cloud implements object detection API in its Vision category. The Azure services uses the Cloud to do all the processing. The Azure Cloud object detection uses a more complex algorithm than the SSD-based Tensorflow. Due to being all network based, this method will eliminate any real-time difference because everything will be running in a chain of servers blocks (Cloud). Comparing this to the SSD-based Tensorflow, The Cloud has the advantage of speed without any sacrifices of accuracy. Although it is still dependant on the network and this is the important advantage Tensorflow has against it, that is able to work offline. Azure cloud services have many fields including the compute field. The compute field has sub-categories shown in Fig. 4.

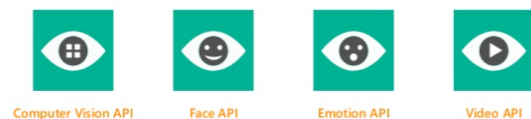


Fig. 4. Microsoft Azure Compute Services

The Face library is used for face recognition, while the emotion library is used for emotion recognition. The video library is used to analyze videos. The computer vision supports the object detection and recognition library. The biggest limitation of Azure Cloud is that it relies on constant network connection to the cloud server (online status). This means that any interference or weak coverage will make the services redundant (unusable). Microsoft Azure Cloud is a powerful service. It consists of a huge server blocks that can have complex calculations with high accuracy.

## V. TEST SETUP

Testing Object Detection functions with high-quality images is difficult. Because high-quality images will yield very close and accurate results, leading to any comparisons being within margins of error. To counter-effect this, the use of lower quality test images will show the true relation between the methods. To make this test, images of resolution of 800x600 (0.48MP) will be used. The same images will be passed to both Tensorflow and to Microsoft Azure cloud to get the output result. The camera module (GC2035) that will be used is shown in Fig. 5.

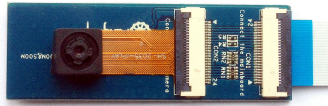


Fig. 5. CCD Camera module (GC2035)

Fig. 6 shows the Microsoft Azure Cloud results page. The result that will be used with the comparison is the “Tag” result with the highest confidence. Confidence rate is the accuracy out of 1.00 (100%). To keep the comparison simple and direct, multi-object detection is ignored. The focus will be only on single object detection.

FEATURE NAME:	VALUE
Description	{ "tags": [ "train", "platform", "station", "building", "indoor", "subway", "track", "walking", "waiting", "pulling", "board", "people", "man", "luggage", "standing", "holding", "large", "woman", "yellow", "suitcase" ], "captions": [ { "text": "people waiting at a train station", "confidence": 0.8330993 } ] }
Tags	[ { "name": "train", "confidence": 0.9975446 }, { "name": "platform", "confidence": 0.995543063 }, { "name": "station", "confidence": 0.9798007 }, { "name": "indoor", "confidence": 0.9277198 }, { "name": "subway", "confidence": 0.838939548 }, { "name": "pulling", "confidence": 0.4317156 } ]
Image format	"Jpeg"

Fig. 6. Microsoft Azure object detection cloud interface

Tensorflow model that is used in this testing setup will be the standard “ssd\_mobilenet\_v1\_coco” model. As the comparisons is only focused on the accuracy, the speed will be ignored. Thus the Tensorflow will run on a standard Linux machine similar to the cloud servers. Also as Tensorflow is an offline API, any latency effect due to network connections will be ignored to keep both sides fair.

The testing room is assumed to be well lit, with standard office size (8 x 10 ft). The layout is shown in Fig. 7. The testing room is important and depending on the room size and/or lighting, it will change the results. Using a standard room setup will insure a close testing to ideal conditions. This means that any outlier will be eliminated in the testing.

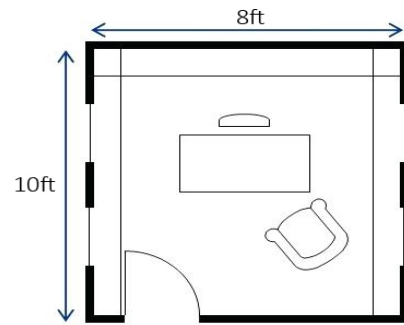


Fig. 7. Standard room layout is selected for ideal conditioning

All testing images will follow the same test setup.

## VI. TENSORFLOW VS. AZURE CLOUD

The testing is done with 300 test images. The images were in the same low quality resolution of 0.48MP. The results of both Microsoft Azure Cloud and Tensorflow were recorded. Any result with lower accuracy than 70% is considered “Not identified” as it is the ideal margin. One of the results is shown in Fig. 8.

Image	
Object name	Water Bottle
Tensorflow result	Bottles

Fig. 8. Testing Result Table 1 (Tensorflow)

As the figure shows, The results are grouped into tables, each table is with the image and object name recorded. The images are in the same resolution taken with the selected CCD module at (0.48MP). The object name is the standard name for the object in the English dictionary. Then the output result of both Tensorflow and Azure Cloud. In this test, the object is a “bottle”. The Tensorflow was able to detect it correctly. The Azure Cloud also detected the object correctly with a high confidence rate of 97% as shown in Fig. 9. The Azure Cloud was able to get more details about the image like “indoor” and “table”. This shows an advantage of the Cloud as it can get more information, although some are very weak in accuracy (confidence rate).



Description	{ "tags": [ "bottle", "indoor", "table", "sitting", "food", "refrigerator", "small", "kitchen", "man", "standing", "wine", "counter", "water", "laying", "computer", "holding", "wooden", "desk", "white", "dog", "woman", "people", "plate", "phone", "group", "room", "pizza" ], "captions": [ { "text": "a bottle of wine", "confidence": 0.7719 } ] }
Tags	[ { "name": "bottle", "confidence": 0.973724246 }, { "name": "indoor", "confidence": 0.8513229 }, { "name": "beverage", "confidence": 0.550468445 }, { "name": "glass", "confidence": 0.0539516956 }, { "name": "person", "confidence": 0.0394385643 } ]

Fig. 9. Testing Result 1 (Azure Cloud)

Fig. 10 shows another test image. This test has a “wall clock” as the object. Tensorflow was able to detect the clock correctly. The Azure Cloud got “wall” as the highest result as shown in Fig. 11.


Image	
Object name	Wall Clock
Tensorflow result	Clock

Fig. 10. Testing Result Table 2 (Tensorflow)

Although it is not wrong as most of the picture is a “wall”, the target should have been the clock. There is a low confidence result describing a clock hanging on the wall. But with an accuracy of 43.6%, it can not be considered an acceptable result.

FEATURE NAME:	VALUE
Description	{ "tags": [ "indoor", "white", "sitting", "hanging", "dark", "light", "air", "laying", "room", "clock", "man" ], "captions": [ { "text": "a clock hanging on the wall", "confidence": 0.426163822 } ] }
Tags	[ { "name": "wall", "confidence": 0.9985228 }, { "name": "indoor", "confidence": 0.873450756 }, { "name": "vacation", "confidence": 0.873450756 }, { "name": "art", "confidence": 0.706003547 }, { "name": "plane", "confidence": 0.6130702 }, { "name": "fog", "confidence": 0.5812746 }, { "name": "winter", "confidence": 0.5785161 }, { "name": "light", "confidence": 0.3909366 } ]

Fig. 11. Testing Result 2 (Azure Cloud)

Other tests followed the same process, and the results were grouped in the testing summary. The testing summary of Microsoft Azure Cloud is shown in Fig. 12.

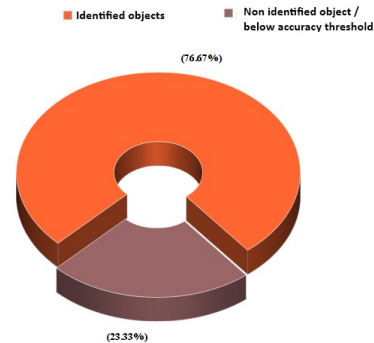


Fig. 12. Pie Chart of testing summary - Microsoft Azure Cloud

As the figure is showing, The identified objects average detection rate results is 76.67%. On the other side, fig. 13 shows the Tensorflow testing summary. The objects average detection rate is 73.33%.

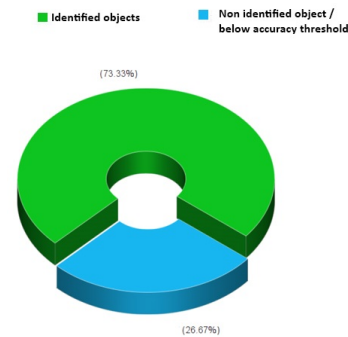


Fig. 13. Pie Chart of testing summary - Tensorflow

The testing summaries are showing that the average difference between Microsoft Azure Cloud and Tensorflow is 3.34%. This small difference can be even reduced with further training, and by increasing the testing pool.

Both object detection methods have limitations, one common limitation is the processing power required. The Azure cloud depends on the servers to do its work, while Tensorflow needs a local machine to do the processing. Another limitation can be cost. The Azure Cloud, and all other Cloud services, have a subscription-based model. This means using the Cloud processing adds up to the net cost with time. While Tensorflow is an open-source library, there is no maintained cost other than the local system needed.

Lastly, both object detection methods can be trained further, but the issue is that training takes time, and results will vary depending on the training method. This motivates further research to look into other or create new object detection methods.

## VII. CONCLUSION

This paper compares two APIs for object detection: Tensorflow Object Detection API and Microsoft Azure Cloud. The performance of the two techniques is close despite their differences. To use either of them in a system integration will depend on many factors. For example, the choice will depend on if the network coverage is available or not, or if cloud processing instead of local processing is preferred. Finally, the method of training to improve the accuracy will have an impact on the choice. The technique and available duration for training sessions also affect their performances. Day by day, there are more algorithms introduced. Object detection is heading towards advance Artificial intelligent, with this - accuracy and performance will be improved.

The results of this research show the importance of looking for other object detection technique that could avoid the limitations of the existing techniques.

## REFERENCES

- [1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 11 2013.
- [2] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 1440–1448.
- [3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, 06 2015.
- [4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 779–788.
- [5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37.
- [6] W. Yuebin, "Pano-rsod: A dataset and benchmark for panoramic road scene object detection," *Electronics*, vol. 8, p. 329, 03 2019.
- [7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, and X. Zheng, "Tensorflow : Large-scale machine learning on heterogeneous distributed systems," 01 2015.
- [8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. Lawrence Zitnick, "Microsoft coco: Common objects in context," pp. 740–755, 2014.
- [9] T. Naga Lakshmi, N. Janaki, and R. Team In, "Deep learning based detection and recognition of objects using mobile nets and ssds," *International Journal of Development Research*, vol. 3, 07 2018.