# An Evaluation of Convolutional Neural Network Models for Object Detection in Images on Low-End Devices

David Foley[1] and Ruairi O'Reilly[2]

[1] Cork Institute of Technology, Ireland,
david.foley@mycit.ie
[2] Cork Institute of Technology, Ireland,
Ruairi.OReilly@cit.ie,
www.cit.ie

**Abstract.** This research paper investigates the running of object detection algorithms on low-end devices to detect individuals in images while leveraging cloud-based services to provide facial verification of the individuals detected. The performance of three computer vision object detection algorithms that utilize Convolutional Neural Networks (CNN) are compared: SSD MobileNet, Inception v2 and Tiny YOLO along with three cloud-based facial verification services: Kairos, Amazon Web Service Rekognition (AWS) and Microsoft Azure Vision API. The results contribute to the limitations of running CNN based algorithms to solve vision tasks on low-end devices and highlights the limitations of using such devices and models in an application domain such as a home-security solution.

**Keywords:** Computer Vision, Object Detection Models, Neural Networks, Convolutional Neural Network, Low-end devices, Facial recognition

## 1 Introduction

Computer vision is the theoretical and technological concern that arises when building an artificial system capable of obtaining information from images or multi-dimensional data. Object detection is a process widely used in computer vision, and image processing, to detect semantic objects of a particular class (e.g. cat, fire hydrant, human) in digital images or video. Typically, object detection algorithms require a collection of pre-annotated images to extract features related to a specific class. These images are fed into a supervised learning algorithm resulting in the creation of a trained model. This model enables the detection of class instances in an image or video that has not been processed before. Approaches required to solve computer vision problems are typically categorized into one of the following:

***Image Classification***: Algorithms that have the task of solving vision problems where predicting a single correct label for an object within a given image from a predefined set of labels is the goal.

CEUR-WS.org/Vol-2259/aics_32.pdf

***Classification & Localization***: Algorithms that have the task of predicting the correct label of an object and are also expected to display a bounding box around the objects (x, y)-coordinates [1]. "Classification & Localization" techniques are usually applied to tasks for making predictions based on one object within a given image, for problems with multiple objects within an image Object detection and/or Instance Segmentation is usually applied.

***Object Detection***: This is used to classify multiple objects within an image, it not only attempts to solve the problem of correctly predicting a correct label for each object but it is required to place bounding boxes around each object's (x, y)-coordinates [2].

***Image Segmentation***: Similar to Object Detection, instead of placing bounding boxes around the objects within an image, this process groups pixels based on the similarity of their characteristics in order to display objects in a meaningful way that simplifies analysis [3].

ImageNet [4] is a public dataset that consists of 14,197,122 images. Comprised of 20 thousand classes of real-world objects, it has been used as a benchmark on how well algorithms compare relative to each other. Competitions such as the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) are organized and hosted by ImageNet on an annual basis, enabling research teams to evaluate the performance of their computer vision algorithms. The ILSVRC has seen classification models' accuracy improve and their error rates decrease. Prior to 2012 classification models entered had an average error rate of up to 25.8%. In 2012 an algorithm based on a CNN called AlexNet was created. AlexNet entered and won the ILSVRC with an error rate of 16% with the closest rival achieving an error rate of 26% [5].

CNNs' [6] can be described as a construct of neural networks [7]. A CNN has the benefit of being a form of feature extraction that can detect the layers of a feature from a simple to complex representation.

AlexNet is an adaptation of earlier work into CNNs published by *Y. LeCun* where an architecture called "*LeNet-5*" [8] was built for the purpose of character recognition tasks such as reading digits and zip codes. AlexNets adaptation of the LeNet-5 model is seen as a turning point in machine vision algorithms. It has resulted in the architectural design being widely adopted and a multitude of similar approaches being investigated. Since the creation of AlexNet several algorithms have modified their underlying architectures resulting in a decrease in the Top-5 error rate of those models e.g. In 2015 a proposed model from Microsoft achieved an error rate of 3.57% [9].

The application domain for the purposes of this research paper is a home-security solution. The solution would provide intruder detection locally and facial verification (to authorize users) via a cloud-based service. This necessitates one, or more, low-end devices that are capable of acquiring a video feed to provide input for the home-security solution. A low-end device is a device that has constraints in terms of hardware resources, typically they consist of a single-core processor and a limited amount of RAM. They do not have sufficient resources to run traditional operating systems such as Linux/Free BSD, Windows-based

systems but instead opt for a scaled-down Debian Linux distribution [10]. However, they are affordable and could provide an economical solution in an applied setting.

The reason for choosing a home-security solution as the domain is twofold: firstly, the self-evident utility for object detection within the domain; secondly the authors consider it an exemplar capable of demonstrating the benefits that combining object detection and low-end devices have to offer.

The running of vision tasks is feasible as long, as the model parameters fit within the computational resources available to the device on which the task is running and a sufficient level of accuracy can be provided such that the device retains its utility. Vision tasks require a large amount of computing power in order to run effectively so this is a significant challenge for low-end devices.

## 2 Related Work

Several research papers in the area of computer vision and its application to home-security have been published in recent years.

In [11] a home-security surveillance system with SMS based alerts is presented, the research describes the possibility of using frame separation "*k-means*" using "*Background Subtraction*" as a means of object detection, once an object had been detected the homeowner would receive an alert via a GSM-based alert system. In [12] research detailing a Raspberry Pi combined with a PIR sensor that enables an attached camera to acquire images based on motion detection is presented. Once facial features are detected, the DCT facial recognition algorithm [13] is used to compare images. It was stated that the PIR sensors affect-ability is estimated to be 1.22 meters with homeowners being alerted via a GSM alert when an unknown facial feature is detected.

Prior publications from the same authors [14] detail the use of body detection using features based on "*histogram of oriented gradients*" [15]. Similarly, in another publication [16] the application of OpenCV [17] using "*Haar Cascade*" [18] to detect facial features is discussed.

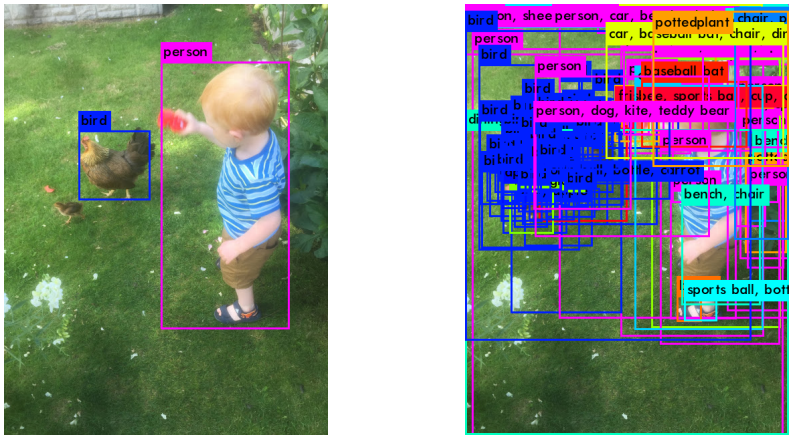
A similar theoretical approach is taken by the work detailed in this paper, in that an image is captured if the motion sensor on the Pi is triggered and if facial features are detected within the captured image these are sent to the homeowners' mobile device via SMS. However, [16] [14] [12] do not incorporate facial verification into the platform as a means of notifying the homeowner nor does it deal with the limitations that can be found with "*Haar Cascade*". This paper details the use of a real-time video feed for the detection of both object-based and facial detection.

## 3 Experimental setup

The intent of the experimental work is to evaluate the robustness and assess the associated limitations when developing systems that require object detection to

be integrated into low-end devices, such as a home security system. This will include the processing of digital images using three cloud-based facial verification services: *Kairos API, Amazon Web Service Rekognition and Microsoft Azure Face API*. To that end, the experimental work will focus on two real-time object detection techniques:

***You Only Look Once (YOLO)*** [19] is an Object Detection model released in 2015 with a performance accuracy of 58.7% (based on VOC 2017 data set) [19]. YOLO applies a CNN which takes the complete image and divides it into regions and calculates predicted bounding box probabilities for each region. Bounding boxes are weighted by the predicted probabilities and increasing or decreasing the threshold of the weights reduces or increases the number of bounding boxes displayed in an image [20] as depicted in Figure 1




(a) 0.5% threshold. (b) 0.1% threshold.

Fig. 1: YOLO Object Detection

***Single Shot Detector (SSD)*** [21] was released in 2016 and has an accuracy performance of over 74%. Similar to YOLO, SSD uses the concept of applying a CNN to the complete image. The image is passed through multiple convolutional layers; each layer operates at a different scale [22] squeezing the layers from 3x3 down to 1x1. This enables the network to detect an object within the image at various scales. SSD discards the fully connected layer and adds instead a set of auxiliary layers. This enables the network to extract features at multiple scales and progressively decreases the size of the input to each subsequent layer [21]. SSD does not classify objects based on threshold sensitivity by default, rather it classifies them based on the Top-5 predictions.

It is envisaged that the results will enable informed decisions to be made when selecting an appropriate object detection algorithm for an application do-

main that utilizes low-end devices, the suitability of an algorithm will become apparent and trade-offs such as accuracy vs. run-time evident. A Raspberry Pi 3B consisting of a quad-core ARM Cortex-A53 1.8 GHz ARM v7 processor with 2 GB RAM was used. An 8 Megapixel Sony IMX219 Image Sensor NoIR camera would be attached to the Raspberry Pi. The specification of the NoIR camera is the following:

Lens: f=3.04 mm, f/2.0
Angle of View: 62.2 x 48.8 degrees
Field of View 2.0 x 1.33 m at 2 m

## 4 Experiments

### 4.1 Object Detection Evaluation

The goal of this section is to evaluate the performance of the object detection models. Considerations to be kept in mind while evaluating the system are the run-time of a given model when classifying the presence of a human and the accuracy of that classification. Two experiments were carried out for each object detection model: (1) Average Execution Time and (2) Maximum Distance and Classification accuracy based on the following three models:

***SSD MobileNet*** [23] was designed by Google for Mobile devices and embedded vision applications. This model has a total of 3,191,072 Neurons across 22 hidden layers [7]. It has a Top-1 accuracy of 71.1% against the ImageNet database. Top-1 accuracy refers to the classifier guessing the correct answer with the highest score.

***Inception v2*** [24] was designed by Google as a scaled down version of Inception, Inception having been introduced in 2014 when it won the ImageNet competition. SSD Inception v2 has 10,173,112 neurons across 22 hidden layers, with a Top-1 accuracy of 73.9% against the ImageNet database.

***Tiny YOLO*** [25] is a scaled down version of YOLO (and is also open source), the model comprises of approximately 1,000,000 Neurons across 16 hidden layers, with a Top-1 accuracy of 58.7% and Top-5 accuracy of 81.5%.

***Average Execution Time:*** The intent of this experiment is to find the average execution time in which a model is able to classify objects within a static image while running on a Raspberry Pi device. The first experiment processed 30 images (96x96 pixels) taken from STL-10 Image Recognition data-set [26] with the processing time being recorded.

Prior to carrying out this experiment, it was anticipated that MobileNet would have a lower execution time than Inception or Tiny YOLO as the model had been designed for running on mobile devices and embedded systems. Similarly, the model has the lowest number of Neurons associated with its CNN and the reduced computation overhead should be reflected in its execution time.

The three models were evaluated: MobileNet, Inception and Tiny YOLO. As expected MobileNet returned the best execution time, followed by Inception and then Tiny YOLO as depicted in Figure 2.

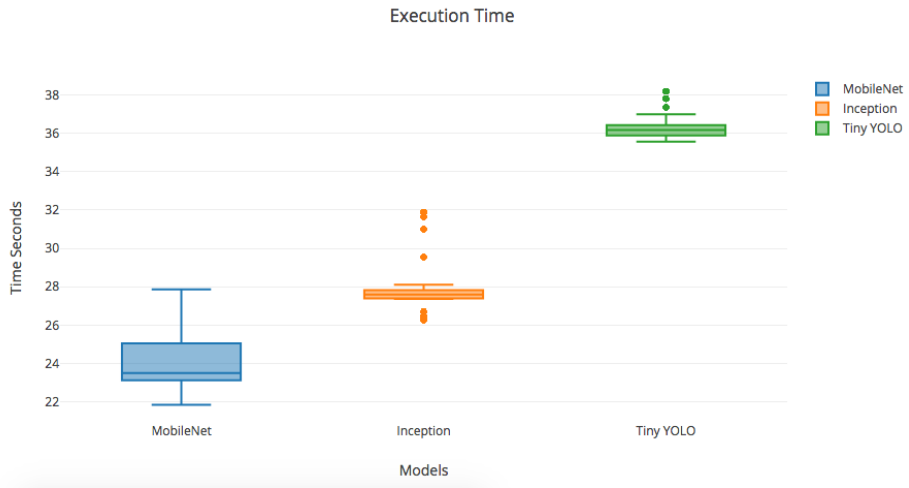



Fig. 2: Model Execution Time

***Maximum Distance and Classification:*** The intent of this experiment is to evaluate the impact of distance and light on a model's ability to detect an object. The experiment was carried out by taking static images of a person at various distances from the camera. Each image was taken with high light pollution from an overhead light in a hallway. Thereafter additional images were taken at the same distance with low light pollution from a landing light located on the second floor. Evaluating a model's ability to detect an object under a variety of lighting and distance based conditions, assists in determining its suitability for an applied setting such as a home-security application.

The results, detailed in Table 1, indicate that the models perform robustly when detecting a "*Person*" at 3.96 meters under both lighting conditions. Inception had the best all-round performance in its classification and prediction scores, outperforming the MobileNet & YOLO models in 5/8 instances. MobileNet performed well and was able to detect objects in both light conditions. However, it failed to classify an object under a low lighting condition within 0.61 meters. YOLO had the best performance for objects at a distance greater than 3.05 meters, but as the object got closer to the camera, accuracy began to decrease. However, it was successfully able to detect an object at 0.61 meters in both lighting conditions.

| Person Detection | | Score | | |
|---|---|---|---|---|
| Distance (meters) | Light | MobileNet | Inception | YOLO |
| 3.96 | High | 59% | 86% | 89% |
| 3.96 | Low | 53% | 58% | 72% |
| 3.05 | High | 66% | 84% | 98% |
| 3.05 | Low | 64% | 81% | 69% |
| 2.44 | High | 62% | 88% | 88% |
| 2.44 | Low | 61% | 87% | 75% |
| 0.61 | High | 97% | 90% | 33% |
| 0.61 | Low | -% | 82% | 22% |

Table 1: Distance Score

### 4.2 Facial Detection Evaluation

The purpose of the facial detection experiments was to evaluate, and compare, the performance of the three cloud-based providers of facial detection services: *Kairos, Amazon Web Service Rekognition (AWS) and Microsoft Azure Vision API.* Understanding the limitations of facial recognition services is important as they would be a critical component for the authorization of users availing of the envisaged home-security system.

***Kairos API*** is a cloud-based Software-as-a-Service (SaaS) offering that provides users access to "*facial detection and facial verification*" services [27]. Kairos stores facial imagery in a private database "*Gallery*" in-order to verify facial features, no training is needed on the front-end of the application as this takes place automatically. Users can add and remove people from the gallery without the need of retraining.

***Amazon Web Services Rekognition*** [28] is a cloud-based SaaS offering that provides users the ability to "*Detect facial features, Compare faces and searching faces in a Collection*". Amazon Rekognition stores facial information into a collection group called "Face Collection". Running an "IndexFaces" operation captures images that are sent to "SearchFacesByImage" operation which produces a prediction score based on similarity of facial features stored within the collection to the captured group.

***Microsoft Azure Face API*** [29] is a cloud-based SaaS offering that provides users with the ability to "*Detect, identify, analyse, organize, and tag faces in photos*". Similarly to the Kairos API, the Face API uses a cloud storage "*PersonGroup*" for known facial features and unlike Kairos the user must train the group before using it, the model must also be retrained after adding, or removing, people from the "*PersonGroup*".

***Facial Detection:*** In order to evaluate the facial recognition services, the "*Wider Face*" data-set [30] (a publicly available data-set with over 30 thousand images with a variance of scale, pose and occlusion) would be used as a benchmark. Ninety images were randomly chosen across a variance of poses and occlusion, the images were then manually inspected, and occurrence of faces

indicated by an annotated bounding box. This enabled the robustness of the cloud-based service providers to be ascertained and their ability to detect facial features to be evaluated. A total of 199 faces were featured across the ninety images, the evaluation showed that AWS had the best performance with 178 facial features detected. Thereafter came Kairos detecting 143 while Azure had the lowest detection score with 135. The overall evaluation of facial detection algorithms has shown that AWS Rekognition performs best with regard to the detection of facial features, while Kairos and Azure are robust in the detection of facial features, they do have limitations in their ability. both were limited in their capacity to detect facial features that had at least a 30° angled away from the camera. This was more prevalent towards Azure, effectively Azure was only able to detect faces that are either directly angled towards the camera or slightly angled in the direction of the camera as depicted in Figure 3.

(a) Azure (b) Karios (c) AWS

(d) Azure (e) Karios (f) AWS

Fig. 3: Facial Detection Comparison [30]

### 4.3 Application Domain Evaluation

The intent of this section is to evaluate the performance of the proposed application domain. A home-security solution that provides intruder detection locally and facial verification (to authorize users) via a cloud-based service running on a low-end device. These experiments were conducted in a controlled environment and evaluate the distance in which object and facial features are detected by the envisaged application.

**Application Configuration and Setup** The image size processed by the initial object detection model experiment "Average Execution Time" relating to execution time was 96 x 96 pixels. A Raspberry Pi Camera has the ability to support up to 3280 x 2464 pixels from a static image. It is assumed that the run-time of the object detection models would increase proportionately with the increase of pixel depth. As such it was decided to deploy the system using MobileNet to achieve the best execution time at the expense of classification accuracy. Amazon Web Service Rekognition performed best overall for facial detection, however, due to the complexity associated with its configuration requirements upon setup this approach is not considered sufficiently user-friendly e.g. each user would need access to the AWS portal and have a user account in order to create groups via Amazon Rekognition. As such, it was decided not to use this platform in evaluating the application domain. Similar to Amazon Rekognition, Azure had its own complexity introduced by requiring users to retrain their model from the image collection (or "*PersonGroup*") each time a change occurs excluding it for consideration when evaluating the application domain.

It was decided to integrate Kairos into the application for its ease of use and it robustness for detecting features at a certain degree. Kairos does not require the manual process of having to retrain the model each time a change is made, reducing the complexity of our proposed system.

**Experiments for the envisaged system carried out were as follows:**

**Experiment One**: This will evaluate the distance at which an object detection model is able to detect human features, facial features and the Kairos API facial recognition in a high-lit area, using a low-end device.

**Experiment Two**: This will evaluate the distance at which an object detection model is able to detect human features, facial features and Kairos API facial recognition in a low-lit area, using a low-end device.

Initial evaluations of the systems ability to detect facial features noted an issue, in that once MobileNet detected an object, a continuous image capture was triggered resulting in these images being continuously uploaded to the cloud-based facial verification service until facial features were detected by Kairos.

This dramatically increased the number of API calls that were being made by the system which could result in a higher running cost of such an application and a reduction in facial classification performance due to the sheer volume of images to be processed. MobileNet was not trained for the detection of facial features and so would trigger the systems event-listener invoking functions resulting in high usage of the Kairos API. In order to resolve this issue, it was decided that the Open Source Computer Vision Library (OpenCV) using the "*Haar Cascade*" algorithm [17] should be integrated into the system.

Integrating OpenCV would reduce the number of API calls to Kairos as OpenCV would locally detect facial features before firing a call to Kairos's facial verification service. Once OpenCV was added to the application, we were able

to proceed with the application domain evaluation.

**Experiment One Results** (high-lighting pollution) as depicted in Table 2 indicates that MobileNet was able to detect the features of a person at 3.96 meters during real-time processing, subsequently OpenCV was able to detect the facial features of a person at 3.05 meters uploading the images to Kairos which was capable of performing facial verification that resulted in a positive "Match" at 2.74 meters.

**Experiment Two Results** (Low-lighting pollution) as depicted in Table 2 indicates that MobileNet was able to detect features of a person at 3.96 meters during real-time processing, subsequently OpenCV was able to detect the facial features of a person at 2.74 meters uploading the images to Kairos which was capable of performing facial verification that resulted in a positive "Match" at 1.52 meters.

The results from both experiments indicates that low-end devices, such as a Raspberry Pi, are capable of using CNN based object detection algorithms and cloud-based facial recognition services and could potentially be utilized in an applied setting such as a home-security solution. The results based on our "Application Domain Evaluation" demonstrates that with MobileNet as the model of choice, an accurate classification solution can be provided for the detection of objects and facial recognition at varying distances.

| Experiment | Human Detection | Face Detection | Face Verification | Kairos Result |
|---|---|---|---|---|
| One | 3.96 Meters | 3.04 Meters | 2.74 Meters | "Match Found" |
| Two | 3.96 Meters | 2.74 Meters | 1.52 Meters | "Match Found" |

Table 2: System Evaluation

## 5 Conclusion

This paper investigated the running of object detection algorithms on a low-end device such as a Raspberry Pi to detect individuals while leveraging cloud-based services to provide facial verification of the individuals detected.

The performance of three computer vision object detection algorithms that utilize CNNs were compared: SSD MobileNet, Inception v2 and Tiny YOLO along with three cloud-based facial verification services: Kairos, Amazon Web Service Rekognition (AWS) and Microsoft Azure Vision API.

The evaluation of SSD MobileNet, Inception v2 and Tiny YOLO demonstrated the correlation between execution time in processing an image and the compute power available to the underlying device. As was expected, the lower the number of neurons associated with a CNN the lower the computational overhead. This was reflected by a quicker response time when processing an image.

As a result of evaluating Kairos, Amazon Web Service Rekognition (AWS) and Microsoft Azure Vision API, it is evident that Kairos was capable of the detection of facial features of individuals detected.

The evaluations carried out demonstrate that low-end devices have sufficient computing power to run certain object detection algorithms with a real-time video feed. Object detection on a low-end device is not limited by the algorithm but by the "compute power" available to it. It is envisaged that this may become less of an issue as compute power increases into the future and object detection algorithms for mobile devices are further researched and developed.

An application domain, in the form of a home-security solution, was chosen to demonstrate the utility of embedding object detection algorithms in low-end devices. The resultant system demonstrated that a low-end device such as a Raspberry Pi is capable of human detection, being able to capture and identify a person's facial features and validate whether the person is known or unknown. This is considered an acceptable use case as the prevalence of inter-connected low-end devices continues to increase within the home.

## References


1. Gregory Rogez, Philippe Weinzaepfel, Cordelia Schmid. LCR-Net: Localization-ClassificationRegression for Human Pose, IEEE Conference on Computer Vision, 2017
2. Ye Yuan, Shijian Tang, Object Detection based on Convolutional Neural Network, Standford University, 2017
3. J. Shi and J. Malik, Normalized Cuts and Image Segmentation, IEEE Conference on Computer Vision and Pattern Recognition, 1997, pp. 731-737.
4. ImageNet Image Database, http://www.image-net.org/
5. Alex Krizhevsky, Ilya Sutskever, Geoffrey E.Hinton, ImageNet Classification with Deep Convolutional Neural Networks, ILSVRC Conference, 2015
6. Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks for Large Scale Image Recognition, ICLR conference on computer vision, 2015
7. Simon S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall, 1999
8. Yann LeCun, Leon Bottou, Yoshua Bengio, Patrick Haffner, Gradient-Based Learning Applied to Document Recognition, IEEE Conference, 1998
9. Forrest N. Iandola, Song Han, SqueezeNet: AlexNet-Leve Accuracy with 50x fewer parameters and ¡ 0.5 MB Model Size, International Conference on Machine Learning, 2017
10. O. Hahm,E. Baccelli,H. Petersen,N. Tsiftes, Operating Systems for Low-End Devices in the Internet of Things: A Survey, IEEE Internet of Things Journal, 2016
11. Ms.Godlin Jasil S.P,Shaik Asif Moinuddin, Shaik Bab Ibrahim, M.Sakthivel,B.Sakthi Arjun, Home security alert system using moving object detection in video surveillance system, ARPN Journal of Engineering and Applied Sciences, 2016
12. Rajat Agarwal, Bhushan Gajare, Omkar Kute3 Pravin Wattamwar, Shobha S. Raskar, Review of Security System based on P.I.R Sensor using Face Recognition Concept, IJSRD - International Journal for Scientific Research Development, 2017

13. Ziad M.Hafed, Martin D.Levine, Face Recognition Using the Discrete Cosine Transform, International Journal of Computer Vision, 2001
14. N. A. Othman, I. Aydin, A new IoT combined body detection of people by using computer vision for security application, International Conference on Computational Intelligence and Communication Networks, 2017
15. Mukesh B.Rangdal, ME VPCOE Baramati,Dinesh B. Hanchate, Animal Detection Using Histogram Oriented Gradient, International Journal on Recent and Innovation Trends in Computing and Communication, 2014
16. N. A. Othman, I. Aydin, A new IoT combined face detection of people by using computer vision for security application, International Artificial Intelligence and Data Processing Symposium, 2017
17. Ivan Culjak, David Abram, Tomislav Pribanic, Hrvoje Dzapo, Mario Cifrek, A brief introduction to OpenCV, International Convention MIPRO, 2012
18. L. Cuimei, Q. Zhiliang, J. Nan, W. Jianhua, Human face detection algorithm via Haar cascade classifier combined with three additional classifiers, IEEE International Conference on Electronic Measurement Instruments, 2017
19. Mohammad Javad Shafiee, Brendan Chywl, Francis Li, Alexander Wong, Fast YOLO:A Fast You Only Look Once System For Real-Time Embedded Object Detection in Video, CoRR,2015
20. Redmon, Joseph and Farhadi, Ali, YOLOv3: An Incremental Improvement, arXiv, 2018
21. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, Alexander C. Berg, SSD: Single Shot MultiBox Detector, CoRR, 2015
22. Gereon Vienken, Scale Selection in Convolutional Neural Networks with Dimensional Min-pooling and Scaling Filters, Masters Thesis, University of Groningen, 2016
23. Andrew G. Howard,Menglong Zhu,Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, CoRR, 2017
24. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, Rethinking the Inception Architecture for Computer Vision, CoRR, 2015
25. Alexander Wong, Mohammad Javad Shafiee, Francis Li, Brendan Chwyl, Tiny SSD: A Tiny Single-shot Detection Deep Convolutional Neural Network for Real-time Embedded Object Detection, CoRR, 2018
26. STL-10 Image Recognition Data Set
https://www.kaggle.com/jessicali9530/stl10
27. Ben Virdee-Chapman, The Business Case for Face Recognition, Whitepaper,
28. Detecting and Analyzing Faces
https://docs.aws.amazon.com/rekognition/latest/dg/faces.html
29. Azure Face verification
https://azure.microsoft.com/en-us/services/cognitive-services/face/
30. Yang, Shuo and Luo, Ping and Loy, Chen Change and Tang, Xiaoou, WIDER FACE: A Face Detection Benchmark, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016