TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones

OBJECT RECOGNITION FROM VIDEO SEQUENCES IN REAL TIME USING DEEP LEARNING TECHNIQUES

Eduardo García Cotrina[1], Omar Antonio Hernández Duany[2], Amado García Albert[3]

[1,2,3]Universidad Tecnológica de la Habana "José Antonio Echeverría", CUJAE, La Habana, Cuba
[1]eduardogarcot@gmail.com, [2]omar.hd@tele.cujae.edu.cu [3]maddoc@tesla.cujae.edu.cu

RESUMEN

En el campo de la visión computacional se ha desarrollado una amplia gama de aplicaciones con diversos propósitos. El reconocimiento de objetos en imágenes siempre ha sido una tarea relevante en este dominio de aplicación. En la actualidad se han desarrollado nuevos métodos para lograr una mayor precisión y eficiencia en el proceso de detección de objetos. En esta investigación se realizó un análisis comparativo de los principales métodos basados en aprendizaje profundo que se encuentran entre los que ofrecen mayor efectividad para la detección y clasificación de objetos. Se propuso una implementación basada en el algoritmo YOLO para el reconocimiento de objetos presentes en secuencias de video, que utiliza la biblioteca para visión computacional OpenCV. Se realizaron experimentos diseñados para validar la solución propuesta sobre dos nodos de cómputo diferentes y fueron analizados los resultados obtenidos de acuerdo a las principales métricas empleadas para la evaluación del rendimiento de los métodos de reconocimiento de objetos.

PALABRAS CLAVES: Reconocimiento de Objetos, Arquitecturas Profundas, Visión Computacional.

ABSTRACT

In the field of computer vision, a wide range of applications has been developed for various purposes. Object recognition in images has always been a relevant task in this application domain. Nowadays, new methods have been developed to achieve greater accuracy and efficiency in the process of object detection. In this research, a comparative analysis of the main methods based on deep learning was carried out, which are among the most effective for object detection and classification. It was proposed an implementation based on the YOLO algorithm for object recognition in video sequences, which uses the library for computer vision OpenCV. Experiments designed to validate the proposed solution were performed on two different computation nodes and the results obtained were analyzed according to the main metrics employed to evaluate the performance of object recognition methods.

INDEX TERMS: Objects Recognition, Deep Learning, Computer Vision.

1. INTRODUCTION

In the field of telecommunications, there has been a notable growth in the bandwidths at which information is exchanged [1]. This has led to an increase in video streams carrying information of interest for decision making, which are transmitted through telecommunication networks [2].

In this context, it is important to integrate the emerging trends in the field of computer vision (CV) in terms of the processing of these video streams [3]. CV is the scientific branch that includes different methods to acquire, process, analyze and understand real world images in order to produce data related to these images for later analysis and decision making [4].

Nowadays, CV has been developed on the basis of Machine Learning (ML), a field of computer science whose objective is to promote automatic learning through computers, wich means the generation of knowledge from a group of experiences. This area of scientific knowledge originally emerged in the scope of Artificial Intelligence (AI) [5].

ML covers several techniques employed to cover different types of applications. Among these can be quote linear regression models, decision trees, support vector machine algorithms, k-nearest neighbor algorithms and others cited in [6]. In addition, as the authors show in [7, 8, 9], of all the methods used for CV applications the one that obtained the best results was the Artifical Neural Networks (ANNs).

The ANNs can be defined as a set of algorithms or an intelligent system that imitates the nervous system and the structure of the human brain, but they are very different in terms of scale. From the computational point of view, it is possible to appreciate them as a data processor that receives coded information and performs a series of operations resulting in another set of coded information [10].

The ANNs have a hierarchical structure, since they are formed by layers. The greater the number of layers, the more efficient the learning process, which increases the possibility of incorporating more knowledge and results in a significantly expensive process from the computational point of view [11]. The increase in the number of layers in ANNs makes these algorithms known as Deep Learning (DL) algorithms [12].

Currently, DL models have been adopted as solutions in the entire field of CV, which includes models for object detection and classification. Most of the object detectors used today are based on DL, with robust ANNs as the basis of their architecture [10].

The objects inside the images are characterized by their more general features, such as color, shape, textures or others. The determination of whether an image contains a defined object, and if so, its position within it, is a problem that is solved by the CV paradigm. If instead of a single image a video sequence is processed, then the position and size of an object seen in the previous frame can be tracked, or followed, and so on. This results in a high correlation between consecutive frames in the sequence. Eventually, an object will disappear from the sequence and the detection task will be started again [13].

The objects recognition can be divided into two processes. Firstly, the detection, which can be seen as a classification problem where the main task is to determine the presence or absence of a specific object within an image. As long as it is present, its position must be provided. Secondly, the classification process allows to define the class corresponding to each one of the previously detected objects [13]. Examples of object detection in images are the detection of human faces, hand gestures, cars and signs in traffic scenarios, among many others [13, 14, 15, 16, 17, 18].

With the purpose of searching for process automation and optimization of decision making, this article proposes the implementation and validation of a DL-based method for objects recognition from video sequences in real time. This method will be integrated into a multipurpose CV platform, which could be used in tasks such as autonomous driving, pedestrian control, vehicle recognition, among others. A state-of-the-art analysis of DL-based objects recognition is discussed in section 2. Section 3 presents the YOLO method as an alternative selected for the real-time objects recognition. In section 4 is performed an analysis of the validation results of the proposed method. Finally, section 5 contains the conclusions.

2. DEEP LEARNING-BASED OBJECT DETECTION

Nearly all objects detectors and classifiers these days use DL and ANNs to extract the main features from images or videos, then locate and classify them. According to [10], these are divided into two groups: two-stage detectors and single-stage detectors.

The two-stage detectors prescribe high accuracy for objects detection and classification, while the single-stage detectors achieve high processing rates. Two-stage detectors, in general, can be divided into a first module, responsible for detecting the Region of Interest (RoI). Secondly, another module is in charge of extracting the main characteristics from each candidate box for the following classification and regression tasks of the bounding boxes, by means of the RoIPool operation (RoI Pooling). On the other hand, the single-stage detectors propose prediction boxes from the input images, so that the RoI proposal step is omitted, which makes them more efficient in terms of execution time and thus, can be used for devices that require real-time operation [10].

Two-stage detectors

In 2014, Girshick et al. [19] proposed the use of Region-Based Convolutional Networks (R-CNN), which divides the problem into three key stages. In the first, possible regions where objects may exist are extracted using a region-proposal method consisting of a selective search process. In the second, for each proposed region, the main characteristics are extracted through a Convolutional Neural Network (CNN). Finally, each region is classified by Linear Support Vector Machines (LSVM) according to the characteristics extracted from the image by the CNN. A regression algorithm is applied to perform a better delimitation of the objects' boundaries within the image. Additionally, if several of the identified objects overlap too much, the least likely ones are removed to reduce the final number of identified objects.

The selective search, employed by the original version of R-CNN, generates 2000 different regions with a high probability of exsistence for an object of interest. This means that it translates the problem of detecting objects within an image into a classification problem of 2000 images. Consequently, this model allows the detection with precision values superior to 60%. However, it supposes a high computational cost and has a delay of almost one minute for each processed image, time in which a video sequence cannot be processed for a real-time application [10].

After the release of R-CNN, Girshick in [20] improved its previous proposal (R-CNN) by replacing the use of LSVM classifiers with a pure DL solution. The author introduce an end-to-end multilayer network, of conventional type, in charge of both, classifying the image region and carrying out a regression to delimit the boundaries of the identified object in that region. The new model, named Fast R-CNN, is easier to train (an ANN, perfectly distinguishable), although it is still based on the use of selective search.

In Fast R-CNN, the CNN is applied to the entire image and a RoI Pooling mechanism is implemented. The order is exchanged with respect to R-CNN, where regions were first selected and then CNN was applied. This means a computational saving, since the CNN is only used once to analyze the whole image. However, the inference process is still expensive, because it has to be tested with all the proposed regions. Nevertheless, a decrease in execution time was achieved, although it is still considered too high for real-time processing [20].

After Fast R-CNN, in [21], a new model called Faster R-CNN is proposed. This model eliminates the selective search algorithm that generates the RoI and replaces it with a Region Proposal Network (RPN). The latter is inserted after the last convolutional layer of the network and is capable of generating region proposals. These are then used in the rest of the network, as in Fast R-CNN. In general, the image is provided as an input to a CNN that offers a map of convolutional characteristics. This version of the algorithm has a shorter response time than the previous ones, since it works correctly with a processing rate close to 9 frames per second (fps). However, even at 9 fps it is not possible to process a video sequence in real-time [22].

The evolution of the R-CNN method allowed an increase in accuracy for object detection algorithms. As a consequence, the processing rate of these detectors was also increased. The processing times for these methods were so high that they did not allow real-time detection. For this purpose, so-called one-stage methods were developed.

One-stage detectors

The You Only Look Once (YOLO) algorithm, proposed by Redmon et al. [23], provides a strategic alternative for detecting objects in real time, without the need to employ regions. YOLO turns the object detection problem into a regression problem that can be solved with a single CNN.

YOLO starts from an image that is divided into a set of cells. Then, an ANN is used for each cell of the grid into which the image has been divided and generates a vector containing the probabilities that this cell contains an object from each of the classes to be recognized. Simultaneously, it creates a regression model to delimit the regions containing the objects in each cell. For each of them, one or several well-delimited regions are obtained. Those where an object is unlikely to exist are eliminated, wich means that the prediction indicates a low probability for all classes of interest. Finally, the overlaps between the proposed regions are eliminated, as in the case of R-CNN. YOLO can operate with an acceptable accuracy value at a speed of 45 fps, therefore, it can be used in real-time video objects detection [23].

Like it happened with the techniques based on the use of regions, the progress in the field is continuous. Thus, different variants of YOLO have been proposed to improve its accuracy and efficiency. After presenting the first version, Redmon et al released a second [24] and a third version [25] of the algorithm with significant improvements to its accuracy and processing rate.

YOLO has troubles dealing with small objects in groups, due to the strong spatial limitations imposed on the predictions of bounding boxes [23]. In order to solve these problems, Liu et al. propose a Single Shot Detector (SSD) [26] which, given a map of specific characteristics, instead of taking the fixed grids adopted in YOLO, uses a set of predetermined anchor boxes with different aspect ratios and scales to discretize the output space of the bounding boxes. To handle objects of various sizes, the network merges the predictions of multiple feature maps with different resolutions. The SSD300 (300x300 size version of the SSD for input images) operates at 59 fps, and consequently allows the performance of real-time objects detection.

Selection of the method

The performance of DL-based algorithms for objects recognition is measured by evaluating two metrics: the Mean Average Precision (mAP), expressed in %, and the processing rate, expressed in fps. In [10] and [22] an analysis of the performance of each of the algorithms described above is conducted, using the COCO data set for objects recognition. The results are illustrated in Fig.1.

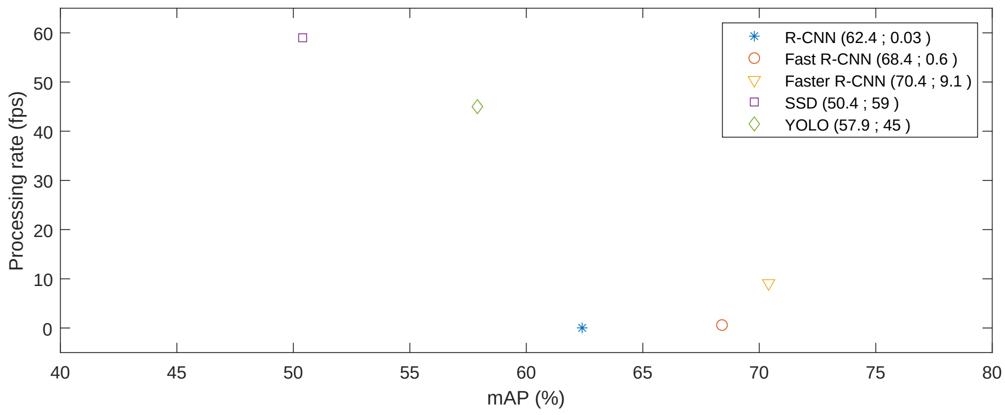


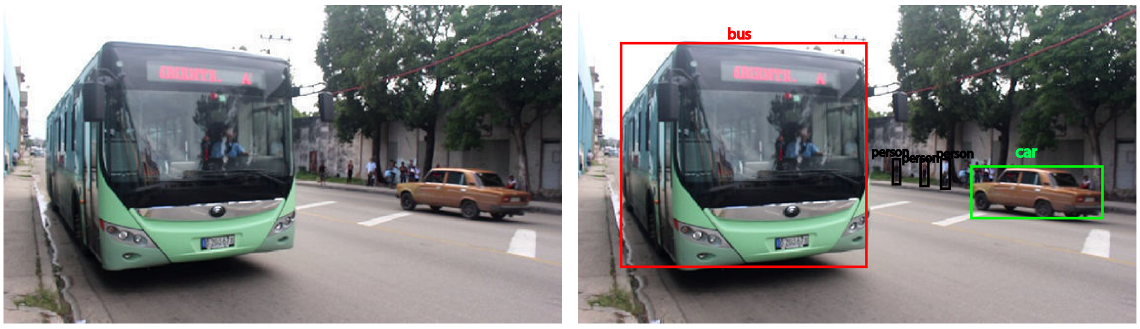
Figure 1: Performance of deep learning-based object recognition methods.

After reviewing Fig.1, it can be determined that with the development of new detection and classification methods, a compromise between processing rate and accuracy has emerged. The algorithm that presents the best balance between the considered parameters is YOLO, because it maintains a high mAP value and a processing rate that allows real-time operation over video frame sequences.

3. DESCRIPTION OF THE PROPOSED METHOD

The proposed method is based on version 3 of the YOLO algorithm for objects recognition in video sequences and images. It was developed in Python 3.7 programming language and uses, as a base, the openCV 4.2.0 library, an open source and free software library for CV solutions. In addition, it uses the numPy 0.42 library to perform mathematical operations on matrices.

Version 3 of YOLO algorithm receives an image as input and makes the predictions of the objects contained in it. Then, it offers as result the same image with drawn boxes that delimit each one of the recognized objects. Each box is associated with a label indicating the class to which the object belongs. In Fig.2 is shown an image before and after being processed by the method.



(a) (b)

Figure 2: (a) Image before applying YOLO algorithm; (b) Image after applying YOLO algorithm.

The proposed version of YOLO is designed to process images with dimensions up to 4000x3000 pixels. Each image is divided into squared cells of predefined size, where each cell is responsible for detecting any objects present in it [25].

Each cell generates a number of bounding boxes, with a confidence level corresponding to each one of them. The bounding boxes are rectangles that limit a proposed region where an object is contained, with a certain probability that such object exists. As a result, five values are returned, of which 4 correspond to the coordinates of each box (x,y,h,w), where x,y represent the center of the bounding box, and h,w represent the height and width of that box respectively; these last two are referred to the size of the original image. The fifth value is the confidence level, it represents the probability that there is an object in the box. Formally it is defined in [23] as the product of the probability of existence of an object ($P_r(Object)$), and the intersection over union of truth (IOU^{truth}_{pred}):

$$Confidence = P_r(Object) \cdot IOU^{truth}_{pred} \tag{1}$$

In YOLO v1, each of the cells predicted a set of conditional class probabilities P(Classi|Object) for each class detected in the box in question. This set did not depend on the number of bounding boxes detected. If this methodology is maintained, it is very difficult to detect more than one object with the appropriate precision if they are all within the same cell. Therefore, for later versions it is decided to determine a set of class probabilities for each predictor box instead of determining a single set of probabilities per cell [24].

To determine the final confidence value of each bounding box, the one with the highest value is chosen. If it meets an established decision threshold, the presence of an object of that class in the region marked by the coordinates will be determined. While the confidence value is not greater than the threshold, the absence of an object in that box will be assumed [23].

The release of the second and third version brought a number of improvements among which were the limitation of the number of predictor boxes generated by each bounding cell. In the case of the second, to five boxes per cell, while in the third, three scales of bounding cells were defined. The latter ones dimensions are of 13x13, 26x26 and 52x52 respectively. This allowed to improve the performance of small objects detection in the images, which is the biggest limitation of YOLO. Moreover, the number of boxes per cell was limited to three, so in total 10647 boxes will be generated for each image [25].

Up to this point, the algorithm has performed the extraction of the main characteristics, detection and classification, and thus, at the output of the network, a high number of predictor boxes with their confidence values, their respective coordinates and their class probabilities, have been obtained. In order to achieve a single detection of an object, a technique known as Non-Maximum Suppression is applied. The same eliminates the non-maximum confidence values associated with bounding boxes with coordinates similar to the one in question [27].

To enable the operation of the algorithm, the architecture presented by Redmon et al. in [25] is used, which is shown in the Table 1. This architecture uses a variant for feature detection that merges the Darknet-19 architecture of YOLOv2, with the architecture of the new pyramidal networks for feature detection. This new architecture is called Darknet-53 and contains 53 convolutional layers, each of them followed by the batch normalization layer and the Rectified Linear Unit (ReLU) activation function. In this case, pooling sub-layers are not used, instead convolutional layers are introduced with step 2 to reduce the sample of feature maps [25]. This approach helps to prevent the loss of low-level features often attributed to the clustering process [27].

For the development of the method, a workflow was adopted, as illustrated in Fig.3. For this purpose, the *dnn* module of the aforementioned openCV library was used, which allows a limited handling of deep architectures. It contains an Application Programming Interface (API) for the creation of new layers, as well as a set of the most useful layers. In addition, it has another API for building and modifying complete neural networks from the layers, and offers the functionality for loading serialized network models from different frames. The major disadvantage offered by this module is that it does not support the training of these architectures [28]. This is why it was proposed to work with the weights predefined by [29].

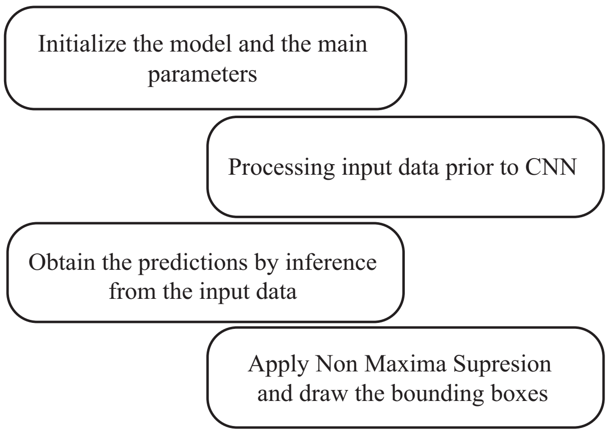

Initialize the model and the main parameters
Processing input data prior to CNN
Obtain the predictions by inference from the input data
Apply Non Maxima Supresion and draw the bounding boxes

Figure 3: Workflow of proposed method.

Table 1: YOLO v3 Architecture [25].

Layer Type	Filter	Size	Output Size	
Convolutional	32	3x3	256x256	
Convolutional	64	3x3/2	128x128	
Convolutional	32	1x1	128x128	
Convolutional	64	3x3	128x128	
Residual			128x128	
Convolutional	128	3x3/2	64x64	
Convolutional	64	1x1	64x64	
Convolutional	128	3x3	64x64	
Residual			64x64	x2
Convolutional	256	3x3/2	32x32	
Convolutional	128	1x1	32x32	
Convolutional	256	3x3	32x32	
Residual			32x32	x8
Convolutional	512	3x3/2	16x16	
Convolutional	256	1x1	16x16	
Convolutional	512	3x3	16x16	
Residual			16x16	x8
Convolutional	1024	3x3/2	8x8	
Convolutional	512	1x1	8x8	
Convolutional	1024	3x3	8x8	
Residual			8x8	
Avg Pool				
Fully Connected		1000		
SoftMax				

In the first stage of the workflow, the model of the network and its weights are loaded through the function *dnn.readNetFromDarknet(...)*. The input parameters are the model's path and the weights' path. To load the class labels, the file *"labels.txt"* is read, which contains the predefined 80 classes of the COCO model, the one employed by the creators of YOLO. Additionally, the confidence threshold is set.

During the second stage, the image to be processed is converted into a Binary Large Object (BLOB). To achieve this, the function *dnn.blobFromImage(...)* is used, which main parameters are the image in question, the scale factor, and the size of the image. Furthermore, the location of the output layer of the model is obtained by means of the functions *dnn.getLayerNames()* and *dnn.getUnconnectedOutLayers()*.

At the third stage, the image is passed to the network using two functions. Initially, the function *dnn.net.setInput(...)* prepares the data in the input layer and has as a parameter the image in the BLOB format. Secondly, the function *dnn.net.forward(...)* sends the data to the network and obtains the output in the corresponding layers.

In the fourth and last stage, the process of Non-Maximum Suppression occurs. It is performed using the function *dnn.NMSBoxes(...)*, which has as its main parameters the delimiter boxes generated in the previous stage and the confidence threshold set in the first stage. The resulting bounding boxes are then drawn on the image. Once the complete process finishes, it is repeated until the analysis of all the frames in the video sequence is completed.

It should be noted that the proposed method does not take advantage of the benefits offered by high-performance computing, which is focused on the use of multiple processors and graphics cards.

4. RESULTS

In order to evaluate the performance of the method, it was proposed to carry out the detection and classification of objects in a set of images composed of 50 samples. There was a previous knowledge of the objects contained in them. The experiment was conducted in two computation nodes, defined as Node A and Node B, with the technical characteristics shown in Table 2.

Table 2: Node A and Node B technical characteristics.

Characteristics	**Node A**	**Node B**
CPU	Intel Core i7-5500 2.4 GHz	Intel Core i7-8550U 1.8 GHz
RAM	8 GB DDR3	32 GB DDR3
HDD	SATA 1 TB 6 MB/s	SSD 1TB 100 MB/s
GPU	No	Intel UHD Graphics 620 16 GB
OS	Windows 10 Pro	Windows 10 Pro

The method was configured with the model and weights predefined by Redmon et. al in [25], so it allowed the classification of objects using the classes of the image dataset COCO, which covers 80 different classes. For each experiment, 5 runs of the method were performed and the execution time for the detection of each image was measured. Furthermore, the number of true positives (TP), false positives (FP), and false negatives (FN) were obtained. These values represent the number of objects that were correctly and incorrectly recognized as well as not recognized, respectively. The results are displayed in the Table 3.

Table 3: Results of the detection experiments.

Amount of Images	**Amount of objects to detect**	**True Positives**	**False Positives**	**False Negatives**
50	387	341	13	46

Three fundamental metrics are used to determine the quality of objects recognition methods. The first metric is called *Precision*, and represents the extent to which all detected objects are correctly classified, it is mathematically defined by:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

On the other hand, the *Recall* represents the extent to which all objects that should be recognized are, indeed recognized. It is mathematically defined by:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Finally, the third and main metric to evaluate the performance of the objects recognition methods is the mAP, which expresses how trustable is the proposed method. It is determined by calculating the area under the *Recall* vs *Precision* curve.

From the obtained results, each one of these metrics was calculated. A *Precision* value of 96.14% and a *Recall* of 88.86% was reached. Fig.**??** shows the *Recall* vs *Precision* graph for the proposed method.

From it, an mAP value equal to 88.43% was determined. This result is positive, because it shows that the proposed method would have a high accuracy rate in the objects recognition for scenarios with similar characteristics to the set of images employed. It also results in a value with a margin of improvement of 12% in order to achieve values close to 100% of mAP. This requires adjustments to the method and a more robust data set than the one used for these experiments.

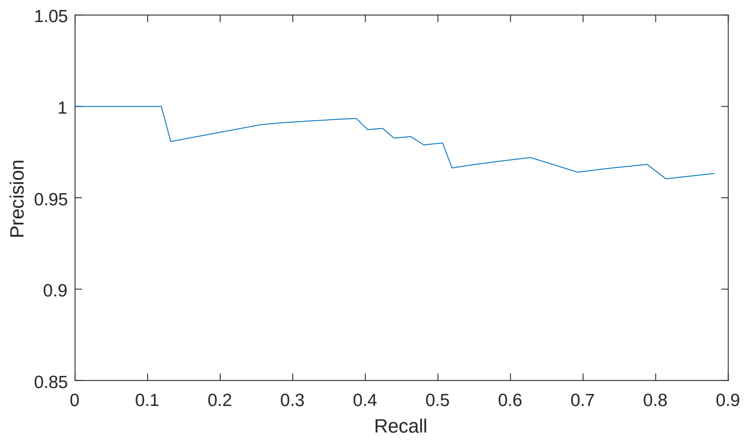


Figure 4: Recall vs Precision.

The measurements of the execution times of the method showed results with an expected behavior. For the node A, which presented the least computing power, the execution time was between 1.17 and 2.26 seconds per image, with an average of 1.63 seconds per image. While, for node B there was a reduction of 44% compared to the previous one, due to the fact that this node has a notably higher computing power than node A. For the latter, values ranging from 0.63 to 0.98 seconds per image were obtained, with an average of 0.73 seconds per image. The Fig.5 shows the execution times for each of the images on which the experiment was performed.

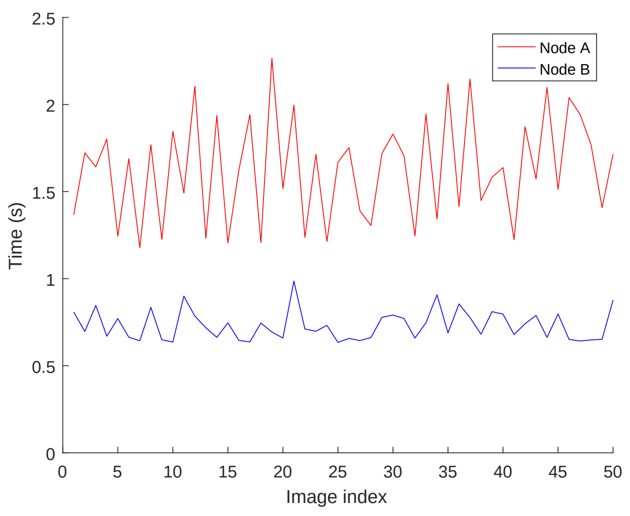


Figure 5: Experiments execution times for Node A and Node B.

To achieve real-time detection of the objects found in the video sequences, it is necessary to minimize the execution time. The chosen method must be capable of processing approximately 30 frames per second. Then, the maximum possible delay for processing each frame should be 33 milliseconds. According to the obtained results, the processing of the video sequences in real time to detect the objects within each frame will not be achieved. It is necessary to emphasize that the hardware environments used for the validation of the method have low computing power in comparison with those used in the literature to obtain satisfactory results regarding this parameter. Thus, this problem could be solved by scaling the method with a parallel architecture or using it in a hardware environment with higher computing power.

5. CONCLUSIONS

The detection and classification of objects contained in video sequences is a complex issue addressed within the main lines of the CV. This article describes a method for objects recognition in video sequences based on the use of DL techniques.

The implementation of the version 3.0 of the YOLO algorithm for the detection and classification of objects corresponding to 80 classes of interest was accomplished. In addition, experiments were carried out to measure the performance of the method, from which it was concluded that it presents a mAP value of 88%. Besides, the method execution time for two computation nodes with different technical characteristics was determined. It was obtained a lower processing time for the node that presented a more powerful hardware architecture.

Finally, it was concluded that the execution time is not low enough to perform the detection in real time. Therefore, one of the future lines of research is the optimization of the method to reduce the execution time of the process associated with the detection by using more powerful computer architectures and parallel computing techniques.

6. REFERENCES

[1] Francisco García Izquierdo. «Desarrollo de una herramienta para la medida de calidad de vídeo». Grado en Ingeniería de las Tecnologías de Telecomunicación. Universidad de Sevilla, 2017.

[2] Maria Torres Vega et al. «An experimental survey of no-reference video quality assessment methods». In: *International Journal of Pervasive Computing and Communications* 12.1 (2016), pp. 66–86.

[3] Romeo Granados Sebastian Juan. «Métodos de visión por computador para localización e identificación de objetos y texturas en exteriores». PhD thesis. Universidad Complutense de Madrid, 2014.

[4] L Enrique Sucar and Giovani Gómez. *Visión computacional.*

[5] Ana González Muñiz. «Aplicaciones de técnicas de inteligencia artificial basadas en aprendizaje profundo (deep learning) al análisis y mejora de la eficiencia de procesos industriales.» Máster en Ingeniería de Automatización e Informática Industrial. Universidad de Oviedo, Feb. 2018.

[6] S. Ray. «A Quick Review of Machine Learning Algorithms». In: *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon).* Feb. 2019, pp. 35–39. DOI: 10.1109/COMITCon.2019.8862451.

[7] Eralda Nishani and Betim Çiço. «Computer vision approaches based on deep learning and neural networks: Deep neural networks for video analysis of human pose estimation». In: *2017 6th Mediterranean Conference on Embedded Computing (MECO).* June 2017, pp. 1–4. DOI: 10.1109/MECO.2017.7977207.

[8] Rahul Suresh and N Keshava. «A Survey of Popular Image and Text analysis Techniques». In: *2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*. Vol. 4. Dec. 2019, pp. 1–8. DOI: `10.1109/CSITSS47250.2019.9031023`.

[9] Rigoberto Vizcaya, José M. Flores Albino, and Saul Lazcano-Salas. «Desempeño de una red neuronal convolucional para clasificación de señales de tránsito». In: vol. 5. 2017, pp. 795 –802.

[10] Licheng Jiao et al. «A Survey of Deep Learning-based Object Detection». en. In: (July 2019). DOI: `10.1109/ACCESS.2019.2939201`.

[11] Fernando Berzal. *Redes Neuronales & Deep Learning*. Granada, 2018.

[12] C. Dong et al. «Learning a depp convolutional network for image super-resolution». In: *Proc. IEEE Eur. Conf. Comput. Vis. Pattern Recog.* (2014), pp. 184–199.

[13] Bogusław Cyganek. *Object Detection and Recognition in Digital Images Theory and Practice*. AGH University of Science and Technology, Poland: A John Wiley & Sons, Ltd., Publication, 2013. ISBN: 978-0-470-97637-1.

[14] K. Dang and S. Sharma. «Review and comparison of face detection algorithms». In: *2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence*. Jan. 2017, pp. 629–633. DOI: `10.1109/CONFLUENCE.2017.7943228`.

[15] A. Ananthakumar. «Efficient Face And Gesture Recognition For Time Sensitive Application». In: *2018 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*. Apr. 2018, pp. 117–120. ISBN: 2473-3598. DOI: `10.1109/SSIAI.2018.8470351`.

[16] E. A. Sürücü and H. Doğan. «Traffic sign recognition with hierarchical Convolutional Neural Network». In: *2018 26th Signal Processing and Communications Applications Conference (SIU)*. May 2018, pp. 1–4. DOI: `10.1109/SIU.2018.8404702`.

[17] Y. Chiu, H. Lin, and W. Tai. «Implementation and Evaluation of CNN Based Traffic Sign Detection with Different Resolutions». In: *2019 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*. Dec. 2019, pp. 1–2. ISBN: 2642-3529. DOI: `10.1109/ISPACS48206.2019.8986319`.

[18] Yali Amit. *2D Object Detection and Recognition: Models, Algorithms, and Networks*. MITP, 2002. ISBN: 978-0-262-26709-0.

[19] R. Girshick et al. «Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation». In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. June 2014, pp. 580–587. ISBN: 1063-6919. DOI: `10.1109/CVPR.2014.81`.

[20] R. Girshick. «Fast R-CNN». In: *2015 IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015, pp. 1440–1448. DOI: `10.1109/ICCV.2015.169`.

[21] Shaoqing Ren et al. «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks». In: *arXiv.org* (June 2015).

[22] Zhong-Qiu Zhao et al. «Object Detection With Deep Learning: A Review». In: *IEEE Transactions on Neural Networks and Learning Systems* 30.11 (Nov. 2019), pp. 3212–3232. ISSN: 2162-2388. DOI: `10.1109/TNNLS.2018.2876865`.

[23] J. Redmon et al. «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. ISBN: 1063-6919. DOI: `10.1109/CVPR.2016.91`.

[24] J. Redmon and A. Farhadi. «YOLO9000: Better, Faster, Stronger». In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 6517–6525. ISBN: 1063-6919. DOI: `10.1109/CVPR.2017.690`.

[25] Joseph Redmon and Ali Farhadi. «YOLOv3: An Incremental Improvement». en. In: (Apr. 2018).

[26] Wei Liu et al. «SSD: Single Shot MultiBox Detector». In: *ECCV*. 2016.

[27] Python Lessons. *YOLO v3 theory explained.* `https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193`. Jan. 2020.

[28] *OpenCV: Introduction.* `https://docs.opencv.org/master/d1/dfb/intro.html`.

[29] *YOLO: Real-Time Object Detection.* `https://pjreddie.com/darknet/yolo`.

ABOUT AUTHORS

Eduardo García Cotrina: Telecommunications and Electronics Engineer. He is working as lecturer at the Telecommunications and Telematics Department in Technological University of Havana "José Antonio Echeverría"(CUJAE). He is currently working on developing software for Computer Vision. His research interests are: Computer Vision, Software Develop, Data Networks, Artificial Intelligence. His ORCID number is 0000-0003-2897-5697.

Omar Antonio Hernandez Duany: Bachelor in Computer Sciences-University of Havana. Master in Sciences Processing, Storing and Transmission of Signals-Technological University of Havana José Antonio Echeverría (CUJAE). Associate Researcher ACC, Associate Professor. Leader of Programming Discipline in Faculty of Telecommunications and Electronics in CUJAE. Leader of the High Performance Computing for Telecommunications in CUJAE. His ORCID number is 0000-0002-0073-1036.

Amado García Albert: Telecommunications and Electronics Engineer. He is working as professor at the Telecommunications and Telematics Department in Technological University of Havana "José Antonio Echeverría"(CUJAE). He is currently working on developing software for Computer Vision. He is studying the field of Parallel Computing applied to the Signal Processing and Computer Vision. His ORCID number is 0000-0002-7100-0281.

CONFLICT OF INTEREST

The authors declare no conflict of interests.

CONTRIBUTIONS OF THE AUTHORS

- Eduardo Garcia Cotrina: Conceptualization, contribution to the idea, preparation, creation, organization and development of the article.
- Omar Antonio Hernández Duany: Conceptualization, contribution to the idea and organization of the article, critical review of each version of the article's draft, suggestions for the conformation of the final version.
- Amado Garcia Albert: Contribution to the idea and organization of the article,critical review of each version of the paper's draft, suggestions for the conformation of the final version.