IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2022.Doi Number

Optimization algorithm to reduce training time for deep learning computer vision algorithms using large image datasets with tiny objects

Sergio Bemposta Rosende[1], Javier Fernández-Andrés[2], and Javier Sánchez-Soriano[3,*]

[1] Department of Science, Computing and Technology, Universidad Europea de Madrid, 28670 Villaviciosa de Odón, Spain
[2] Department of Industrial and Aerospace Engineering, Universidad Europea de Madrid, 28670 Villaviciosa de Odón, Spain
[3] Escuela Politécnica Superior, Universidad Francisco de Vitoria, 28223 Pozuelo de Alarcón, Madrid, Spain

[*]Corresponding author: Javier Sánchez-Soriano (e-mail: javier.sanchez@ufv.es).

This work is part of the I+D+i projects with reference PID2019-104793RB-C32, PIDC2021-121517-C33, PDC2022-133684-C33, funded by MCIN/AEI/10.13039/501100011033/, S2018/EMT-4362/"SEGVAUTO4.0-CM" funded by Regional Government of Madrid and "ESF and ERDF A way of making Europe". (Sergio Bemposta Rosende and Javier Sánchez-Soriano contributed equally to this work) (Corresponding author: Javier Sánchez-Soriano)

ABSTRACT The optimization of convolutional neural networks (CNN) generally refers to the improvement of the inference process, making this as fast and precise as possible. While inference time is an essential factor in using these networks in real time, the training of CNNs using very large datasets can be very costly in terms of time and computing power. This paper proposes a technique to reduce training time by an average of 75% without altering the results of CNN training with an algorithm which partitions the dataset and discards superfluous objects (targets). This algorithm is a tool that pre-processes the original dataset, generating a smaller and more condensed dataset to be used for network training. The effectiveness of this tool depends on the type of dataset used for training the CNN and is particularly effective with sequential images (video), large images and images with tiny targets generally from drones or traffic surveillance cameras (but applicable to any other type of image which meets the requirements). The tool can be parameterized to meet the characteristics of the initial dataset.

INDEX TERMS Computer vision, dataset, deep learning, training optimization, OpenCV, YOLO

I. INTRODUCTION

Cameras and video technology is continuously improving, and it is increasingly common to find images in FullHD, 2K, 4K or even 8K used as input for training convolutional neural networks (CNN) [1]. Computing capacity has also increased significantly [2] and a great deal of effort is being made to develop hardware with the capacity to run neural networks in real time [3]. This hardware is increasingly compact, efficient, and affordable, enabling embedded or distributed training systems for the construction of distributed object detection and surveillance systems [4-6]. Limited progress has been made however in CNN training [7]. While neural networks are, in theory, trained only once and then later depend on inference, the fact is that neural networks are continuously being retrained, either with new datasets or modifications in the parameters of training algorithms.

Given the current size of images [8], and the need for increasingly exact or precise detection of objects within these images, training times are growing [9] as classic methods of optimizing training become less effective [10].

There are two commonly used methods to reduce training times for deep neural networks:

1. **Image size reduction** [11]. This is an effective method if the objects to be detected or classified occupy a sufficiently large part of the total image so that, even when the image is reduced, these objects still provide sufficient information for the training algorithm [9] [13].
2. **Partition of the original image into a mosaic of images** [7, 14, 15]. This method reduces the size of the image, dividing it into several parts with a predefined size (usually 3x3 or 4x4) with equal dimensions (length and width) to maintain the same proportions as the original image.

Both methods reduce the size of images, which can be processed using more modest hardware, particularly when memory is the principal limitation to processing large images. Both methods, however, have certain drawbacks:

- **Image size reduction** [16]. If objects are small, the loss of resolution may mean these objects become undetectable.
- **Partition of the original image into a mosaic of images**. The image being processed may be smaller but there are more images to process. Additionally, objects may be cut between two images. The superimposition of the regions is a way to minimize this although it does not solve the problem as the area of superimposition must be very large resulting in an even greater reduction of the object, reducing the effectiveness of this solution.

In this paper we propose a method to optimize training times without the losses indicated above. The method was validated in a case study working with traffic images captured by drone. This involved a handicap because the objects of interest are very small compared to the total size of the image. Thus, the solution to reduce the original image was ruled out. For example, the size of a car or pedestrian in an image taken by a drone at a height of 50 meters may be approximately 20x20 pixels, if we reduce the image to a size that can be processed by a PyTorch or Tensor Flow type network, that is, up to 640x640 pixels, we are reducing the image to one-fifth, and the objects will be too small to be accurately detected by the neural networks. Although YOLO can theoretically be trained using target as small as 2x2 pixels [17], our tests with targets smaller than 16x16 pixels had a very low degree of precision.

In this paper we will describe the method used to significantly reduce processing times without diminishing the effectiveness of the trained network.

II. TRAINING OPTIMIZATION ALGORITHM

This algorithm is designed to pre-process the labelled images of a dataset prior to being used in the habitual training process for a deep neural network. The dataset must be labelled using the format for a YOLO type network [18]. Thus, the input of this algorithm is one dataset and the output another dataset constructed using the original images but optimized for training (also in YOLO format). For datasets other than YOLO type, the labels can be translated for use in other formats. For this reason, the method is replicable and extendible to other dataset formats.

A. TERMS USED IN ALGORITHM DEFINITION

- **Target (Object) or BoundingBox**: A labelled element in the image that the neural network should detect. This may be any of the type of object that the future neural network will detect by inference.
- **Selected object**: An object labelled in the image that has been selected as input for the neural network. This object was chosen to be part of the set of objects used to train the neural network.
- **Discarded object**: An object labelled to be discarded as input in the neural network. This object may be duplicated, cut, etc. and is discarded for training purposes.
- **Cropped region**: Portion of the image surrounding a "selected object". The size of this region is a configurable parameter of the algorithm. The region is the piece of the image inputted into the neural network for training in which there is at least one selected object.
- **Key image**: An image on which the object discarding process is not applied. This is established every "N" images. This "N" parameter is configurable in the algorithm.

The difference between an "object" and a "selected object" is that not all marked objects in the image to be recognized are part of the input for training the neural network. Of all the labelled objects, only a subset of these per image will be part of the input of the neural network, the rest being discarded.

B. ALGORITHM

This algorithm, as opposed to the methods described in the bibliography, consists of two phases:

1. Discard of objects and reduction of the training set.
2. Cropping of the training regions and new labelling of objects.

To clarify, we will use a training dataset from high-definition videos or consecutive images taken in short time intervals. In either case, these images are from a great distance where each image contains various marked targets for training with a very small size considering the total size of the image. Each of these images is inputted into the algorithm in the same order they were taken by the camera (see flow chart of all the steps of the algorithm in Figure 1).

In the first phase, that of discarding, all targets are checked against the objects in the previous image. The first image of the dataset is considered a "key" image, so no target is discarded, and this phase is omitted. If these targets show relatively little movement compared to the previous image they are discarded; they will not be selectable objects and will be discarded. This parameter, "relatively small distance", is configurable. The values which given the best results are 1% to 3% of the total image. In 2K or 4K resolution images these are approximately 5 to 15 pixels. The principal factors affecting the selection of this parameter are:

- **Type of recorded scene**. From very static scenes to scenes with lots of movement. The more the objects move the greater the discrimination distance.
- **Number of Frames Per Second (FPS) at input**. When the sequence of images is very close in time objects have a smaller displacement between frames. The higher the FPS the less the discrimination distance.

- **Rotation of objects**. If the target objects in the image move in rotation, that is, around a central axis within the BoundingBox rather than moving across the image, this may cause a loss of the object for training. In this case, pre-processing is simply not recommended.

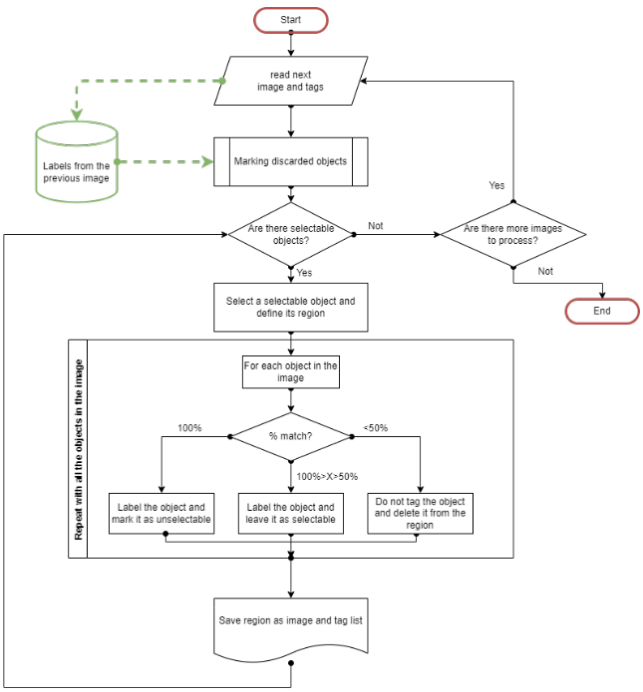

FIGURE 1. **Flow chart diagram of the functioning of the algorithm.**

In the case studies, most of the recorded images are scenes from highways with an average of 2 FPS, although there are some images of agglomerations of people in pedestrian streets or at sports events with a resolution of 1920x1080. In these cases, the parameter was configured at 15 pixels of displacement.

The second phase uses a set of objects that have not been discarded and so are selectable objects. Each of the selectable objects are delimited by a cropped region that is labelled in the image for training purposes. This region is configurable in terms of size and position, but all are the same size with the same ratio or proportion as the original image. The size of the region will depend on the grouping of the objects used for training as well as the size of the image. Larger regions encompass more space within the image, thus reducing the total number of regions but also increasing the computation cost of training. Furthermore, it is important that the region is sufficiently large for the selected object to be contained entirely within it. The cropped region must have the same length-width proportions as the images used for training and later for inference. This is a critical factor in the effectiveness of convolutional neural networks. It is also important that the region does not extend beyond the limits of the image, maintaining the same proportions and size.

Each of the cropped regions is checked for other objects, including those discarded in the first phase. For each of the objects identified within the region one of the following options is applied:

The object is entirely within the cropped region: This object is labelled to be part of the training. If the object is selectable, that is, not discarded in the first phase, it will now be marked as "not selectable" as it is now part of a training region.

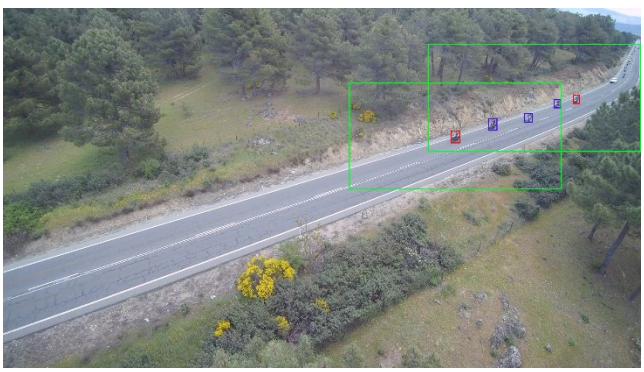

FIGURE 2. **Example of the pre-processor in operation. Using the original image, two regions or sub-images are generated (green squares), centered on the red target objects. The blue targets are ROI included for training which do not generate their own sub-image.**

The object is partially within the cropped region:

- If more than 50% of the object is within the region (this value is configurable and set at 50% for the training process), it is labelled but not marked as "not selectable" and will continue to open to creating its own training region.
- If less than 50% of the object is within the region, the object is not labelled and is deleted (for example, by blurring the image through Gaussian elimination) and it is not marked as "not selectable". In order not to pollute the training process, these images are blurred rather than eliminated (painted a background color) to prevent the network from learning that a specific color (the background color) has any specific utility and incorporating it into its training criteria.
- If the object has been deleted, the labelled and selectable objects will be rechecked to verify that the deletion of a specific object has not led to the elimination of any complete objects if the areas of interest (BoundingBoxes) overlap. If this is the case, restore the image in this area to ensure the selectable object to be used in the training is complete.

Figure 2 shows how from the original image 2 regions or sub-images are generated (green boxes) with the training targets marked in red. These targets generate regions or sub-images, the blue targets are ROI included for training which do not generate their own regions or sub-images. Figure 3 shows how from an original, full-sized image, 4 regions or sub-images are

generated (green boxes) containing the training targets. In this example we see how one of the labelled targets is partially blurred, marked in white, because it was marked as discardable in one of the regions or sub-images because it there is less than a 50% overlap.

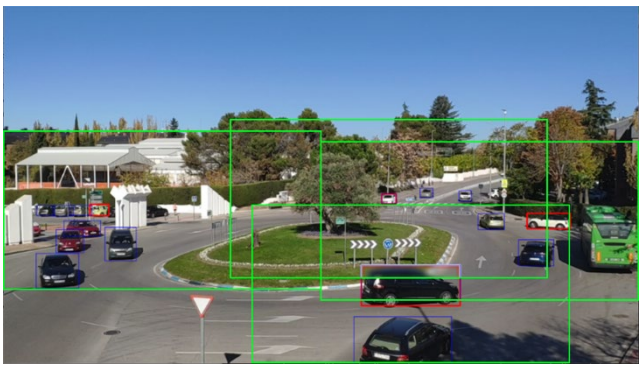

FIGURE 3. **Application with blurred targets (white box).**

By creating a region centered on a selected object, the network will always be trained with a central labelled object. This may cause the network to learn to always expect to detect objects in the center of the region or image. This will not be a problem for this project, given that we have used YOLO as a CNN, which initially divides the image into sections (by default, into 7x7 sections) [19] and that each section has the same probability of containing an object regardless of its position [20, 21].

It is important to note that this method is not ideal for all situations or for all datasets. For this article, we conducted tests using three different datasets, all of them public and verifiable, which allowed us to determine which factors are most beneficial for this algorithm. From these case studies it was determined that there is no optimum configuration for all the configurable parameters of the application. The complete set of images of the dataset determines the configuration and effectiveness of the algorithm. The most important factors which determine the effectiveness of the algorithm (as shown in Figure 4 and Figure 5) are:

- **Large images**, such as FullHD, 2K, 4K or even larger, and with small objects or "targets" to detect considering the size of the image. Examples may be images taken from a certain distance where the elements to detect are distant.
- Images **taken at short intervals**. That is, video images. It is not necessary that the time interval between images is very short (images at 1 FPS is optimum) but they must be sequential and taken from a relatively static camera.
- **Few objects within the image or the objects are not evenly distributed within the image**. That is, objects should be grouped in zones. These images will have large areas with no objects to detect and these areas can thus be eliminated from the training processes.
- **There are static objects of interest in the image**. It is not necessary that static object predominate in the scene but only that there be static objects of interest. This is a factor that reduces the set of images for training, thus reducing the size of the dataset and making the process faster.

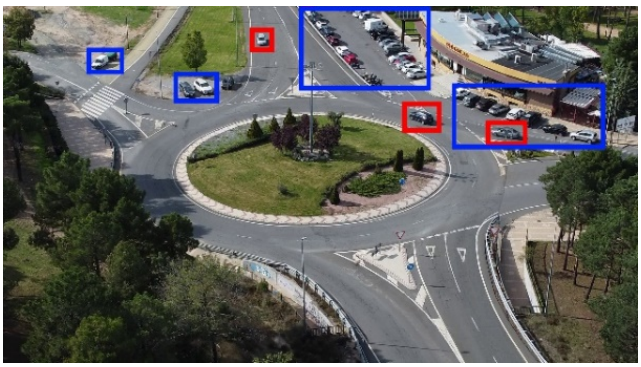

FIGURE 4. **Frame 1 of a sequence of two video frames with static (blue) and moving (red) objects of interest. There is a clear non-uniformity in the density of objects in the image.**

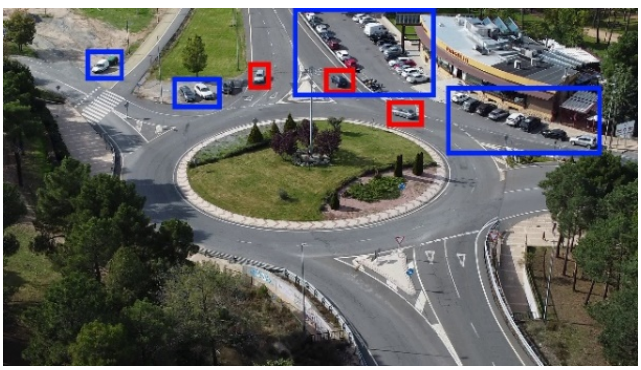

FIGURE 5. **Frame 2 of a sequence of two video frames with static (blue) and moving (red) objects of interest. There is a clear non-uniformity in the density of objects in the image.**

In summary, the types of datasets which best respond to these factors are those consisting of video images from drones or high-resolution static cameras. In this type of dataset, the images are chronological and usually have high resolution. Examples are drone videos observing beaches, roadways, parks, large agglomerations of people or animals, etc. other examples include security or surveillance cameras on highways, streets, buildings, etc. where there are many objects of interest distributed in specific areas of the image, such as cars on a highway, or doorways and entrances for surveillance cameras, etc. These images also generally contain static objects of interest, such as people lying on the beach or parked cars on a city street.

In this case, we used the YOLO neural network which has a series of limitations which make it ideal for the pre-processing algorithm. YOLO divides the image into regions for analysis

and each region is assigned a maximum number of objects [20, 21]. Thus, YOLO is limited to a specific number of objects per region. By dividing the image around groups of objects these are distributed within the new image, permitting a greater number of detections given that there are objects within each of the regions created by YOLO.

It is important to note that this limitation is not critical, but it is a factor to be considered since the number of objects can be [16-19] in the YOLO network. However, the higher the number of objects the slower the process becomes with greater memory consumption. This is a generic factor for all the regions into which YOLO divides the original image, and so the number of objects must be adjusted for the region with the most objects rather than the average number.

III. EVALUATED DATASETS

To determine the effectiveness of the dataset pre-processing algorithm, we experimented with three different datasets, all publicly accessible: Drone [22], Roundabout [23] and VisDrone [24]. The following section provides a description of the principal characteristics of these datasets.

A. "DRONE" DATASET

This dataset consists of images of road traffic in Spain [22], with 12 video sequences recorded by a UAV (Unmanned Arial Vehicle) or drone and from static cameras. These are principally images of critical traffic points such as intersections and roundabouts. The videos are generally recorded at 1 frame per second in 4K resolution. The total dataset consists of 17,570 images of marked objects (types) such as "cars" and "motorcycles". In total there are over 155,000 labelled objects in the dataset: 137,000 cars (88.6%) and 18,000 motorcycles (11.4%). Three frames extracted from the dataset are presented in Figure 6.

FIGURE 6. Frames extracted from the dataset, corresponding to a section of interurban roadway and a split roundabout.

B. "ROUNDABOUT" DATASET

This dataset consists of areal images of rotundas in Spain taken with a drone [23], along with their respective annotations in XML (PASCAL VOC) files indicating the position of the vehicles. In total, the dataset consists of 54 sequences of drone video with a central view of roundabouts. There are a total of over 65,000 images with a resolution of 1920x1080 with 245,000 labelled objects (types): 236,000 cars (96.4%), 4,900 motorcycles (2.0%), 2,000 trucks (0.9%) and 1,700 buses (0.7%). Three frames extracted from the dataset are presented in Figure 7.

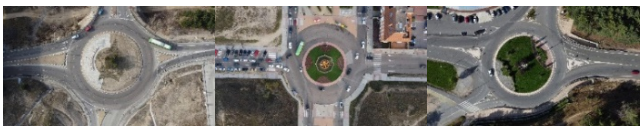

FIGURE 7. Frames extracted from the dataset, corresponding to three different roundabouts with light traffic, heavy traffic and very light traffic.

C. "VISDRONE" DATASET

This dataset is a largescale reference point with carefully annotated data for a computer vision of drone images. The VisDrone 2019 dataset was compiled by AISKYEYE at the Machine Learning and Data Mining Laboratory at the Tianjin University, China [24]. The complete dataset consists of 288 video clips with a total of 261,908 frames and 10,209 static images captured by various drone-mounted cameras with a wide range of different characteristics such as location (14 different cities), setting (urban and rural), objects (pedestrians, vehicles, bicycles, etc.) and density (dispersed or very congested scenes).

FIGURE 8. Fig. 8. Frames extracted from the dataset, corresponding to a parking lot, an intersection, and a rotunda with different intensities of traffic.

It should be noted that the data set was compiled using several different drones in various scenarios and under diverse weather and lighting conditions. These frames were manually annotated with specific objects of interest such as pedestrians, cars, bicycles, and tricycles. Other important attributes are also provided such as visibility of the scene, type of object and ambient occlusion for a better use of the data. Three sample frames from this dataset are provided in Figure 8.

TABLE I
TYPES AND THEIR OCCURRENCE (NUMBER AND PERCENTAGE) IN THE VISDRONE DATASET.

Type	Labels	Occurrence
Pedestrian	372,893	24.4%
People	146,303	9.6%
Bicycle	49,914	3.3%
Car	636,849	41.6%
Van	65,005	4.2%
Truck	45,565	3.0%
Tricycle	38,828	2.5%
Awning-tricycle	19,083	1.2%
Bus	11,443	0.7%
Motorcycle	141,554	9.2%
Extra	3,904	0.3%

For our study, we used only 79 sequences of video consisting of 33,600 frames. There are a total of over 1.5 million labelled items in the dataset, distributed as shown in Table I.

IV. PRE-PROCESSING OF THE DATASETS

The three datasets were pre-processed using the algorithm discussed in this paper, using the following equipment: an 9th

generation Intel i7 processor with 64Gb RAM, SSD hard drive and RTX 2060 graphics card with 8Gb RAM. For software, the study used Microsoft Visual C++ and the OpenCV v4.5 library for their facility in generating compilation files for both Windows and Linux.

A. PROCESSING THE "DRONE" DATASET

The dataset was processed as follows:

- Initial 640x360 image to maintain the same proportion as the images in the original dataset.
- Objects of interest were discarded with their position does not vary in 10px of the image.
- Deletion of counted objects when their area is less than 50%.
- Key image every 7 frames.

After pre-processing the original dataset, the set is reduced to some 15,000 images, with 43,000 labelled objects, of which 36,000 are cars (82.4%) and 7,600 motorcycles (17.6%). A comparison of the original and pre-processed datasets is provided in Figure 9 and Figure 10.

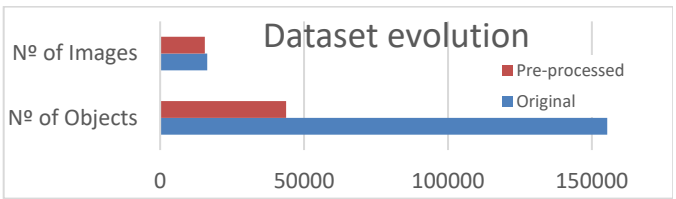

FIGURE 9. Evolution of the number of images and labels after pre-processing of the "Drone" dataset. There is a slight decrease in images and a significant decrease in labels.

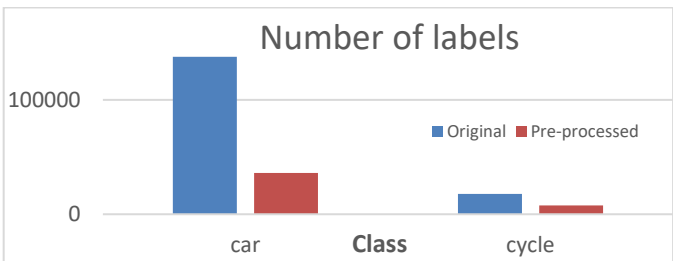

FIGURE 10. Evolution of the number of labels assigned to each type after pre-processing of the "Drone" dataset.

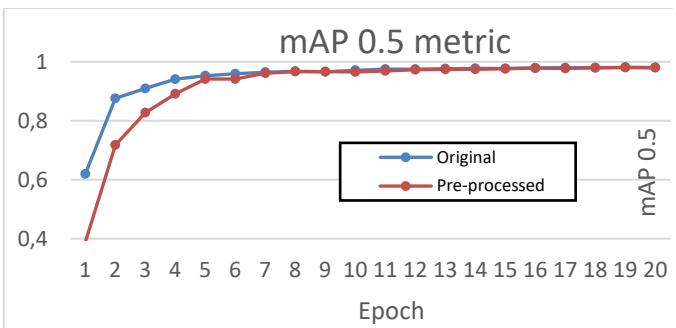

FIGURE 11. Training results for mAP_0.5 of the original and pre-processed images of the "Drone" dataset.

Both datasets, the original and the pre-processed, were used to train a "medium-sized" YoloV5 neural network with the mAP metric adjusted to the value 0.5. The training results in the different epochs are shown in Figure 11.

For this dataset, consisting of 17K images in 2K quality, the training time using the YOLO algorithm and "Yolov5m" network for 20 epochs, was 14 hours and 46 minutes, while the training time using the same computer for the pre-processed dataset was 1 hour and 35 minutes. If we reanalyze the graph of the mAP_0.5 metric but considering training time rather than epochs (Figure 12.), we see a time reduction of some 89.3%.

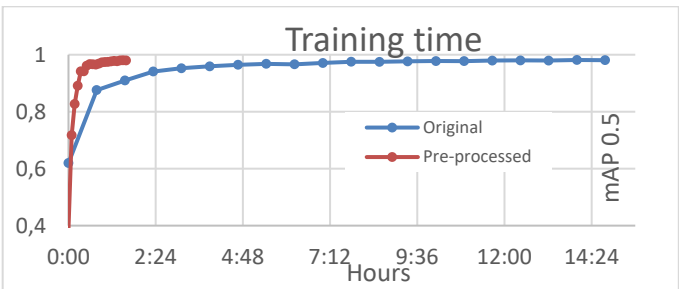

FIGURE 12. mAP_0.5 graph of the time differences in training. The hours of training are indicated on the horizontal axis.

There was a significant reduction in training time. The additional time used for pre-processing, for this dataset 14 minutes, is largely insignificant compared to total training time. For pre-processing, as opposed to the training process for the network, what is most important is not only the graphics card but also storage capacity since the algorithm loads a great deal of images. In our case, we used an SSD hard drive with a Read/Write speed of 600Mb/s.

B. PROCESSING THE "ROUNDABOUT" DATASET

The dataset was processed as follows:

- Initial 640x360 image to maintain the same proportion as the images in the original dataset.
- Objects of interest were discarded with their position does not vary in 10px of the image.
- Deletion of counted objects when their area is less than 50%.
- Key image every 7 frames.

After pre-processing the original dataset, the total number of images was increased to some 188,000, with 756,000 labelled objects, of which 727,000 were cars (96.2%), 10,000 were motorcycles (1.4%), 9,900 were trucks (1.3%) and 7,800 were buses (1.0%). Figure 13. and Figure 14. provide a comparison of the original and pre-processed datasets. In this case, the number of images increased by a rate of 1 to 3.04.

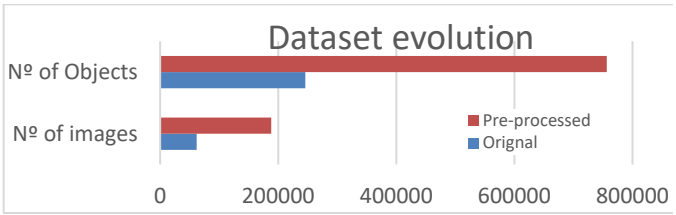

FIGURE 13. Evolution of the number of images and labels after pre-processing of the "Roundabout" dataset. There are significantly more images and labels.

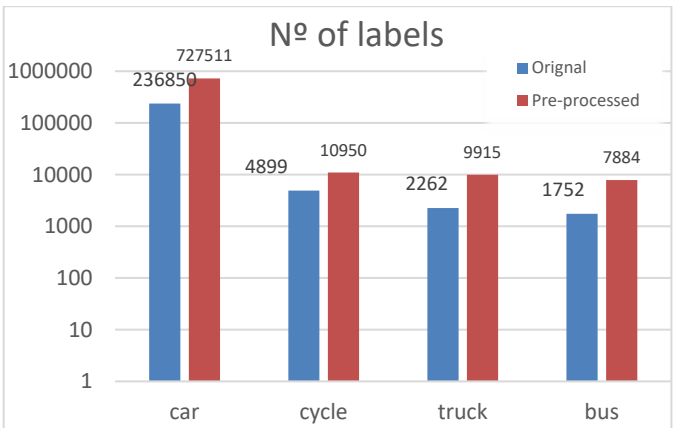

FIGURE 14. Evolution of the number of labels assigned to each type after pre-processing of the "Roundabout" dataset. Cars are the most affected type with a significant increase in the number of labels.

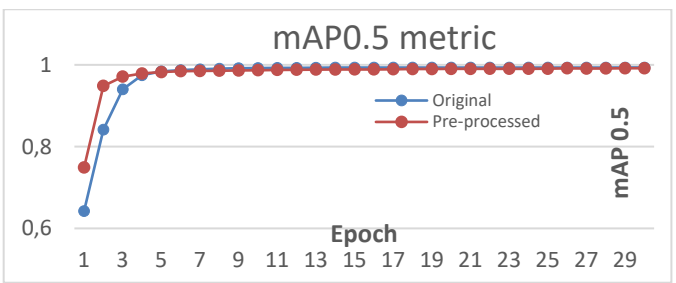

FIGURE 15. Training results for mAP_0.5 of the original and pre-processed images of the "Roundabout" dataset.

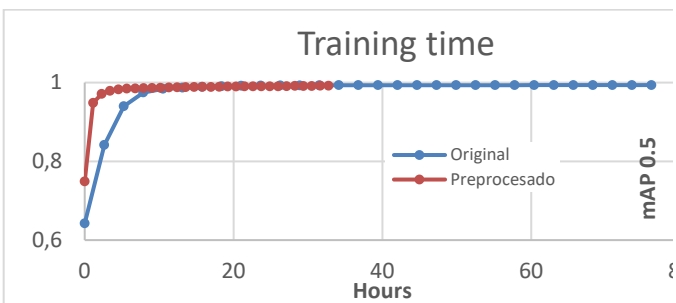

FIGURE 16. mAP_0.5 graph of the time differences in training. The hours of training for the "Roundabout" dataset are indicated on the horizontal axis.

Both datasets, the original and the pre-processed, were used to train a "medium-sized" YoloV5 neural network. The training results in the different epochs are shown in Figure 15.

If we reanalyze the graph of the mAP_0.5 metric but considering training time rather than epochs on the horizontal axis (Figure 16.), we see a time reduction of some 43.0%. To this time must be added an additional 30 minutes in pre-processing time for this dataset.

C. PROCESSING THE "VISDRONE" DATASET

The dataset was processed as follows:

- Initial 640x360 image to maintain the same proportion as the images in the original dataset.
- Objects of interest were discarded with their position does not vary in 10px of the image.
- Deletion of counted objects when their area is less than 50%.
- Key image every 7 frames.

After pre-processing the original dataset, the set of images is increased to 51.5k images, with 600K labelled objects (see Table II). A comparison between the original and pre-processed dataset is provided in Figure 17.

TABLE II
TYPES AND THEIR OCCURRENCE (NUMBER AND PERCENTAGE) IN THE VISDRONE DATASET AFTER PROCESSING FOR THE OPTIMIZATION OF THE TRAINING.

Type	Labels	Occurrence
Pedestrian	141,561	23.6%
People	61,232	10.2%
Bicycle	21,279	3.5%
Car	249,296	41.6%
Van	23,698	4.0%
Truck	16,774	2.8%
Tricycle	16,101	2.7%
Awning-tricycle	7,110	1.2%
Bus	4,402	0.7%
Motorcycle	57,326	9.6%
Extra	1,064	0.2%

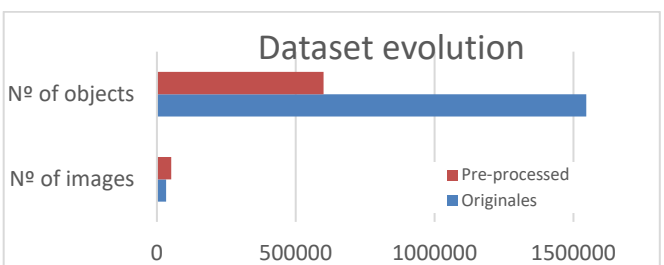

FIGURE 17. Evolution of the number of images and labels after pre-processing of the "Visdrone" dataset. There is a slight increase in images and a significant decrease in labels.

In this case, the number of images has increased by a rate of 1 to 1.543 (154.3%) while the number of labelled objects falls to 38.8%. Both datasets, the original and the pre-processed, were used to train a "medium-sized" YoloV5 neural network. Training results in the different epochs are shown in Figure 18.

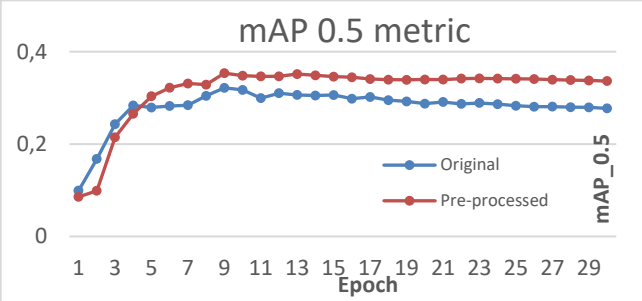

FIGURE 18. **Training results for mAP_0.5 of the original and pre-processed images of the "Visdrone" dataset.**

If we reanalyze the graph of the mAP_0.5 metric but considering training time rather than epochs on the horizontal axis (Figure 19.), we see a time reduction of some 75.0%. To this time must be added an additional 25 minutes in pre-processing time for this dataset.

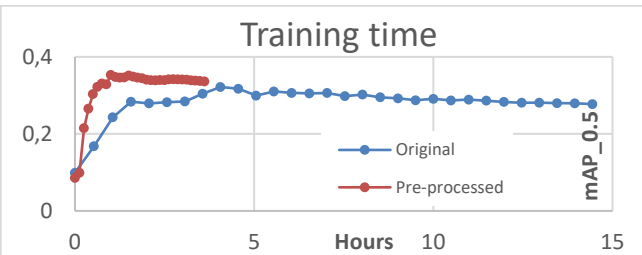

FIGURE 19. **mAP_0.5 graph of the time differences in training. The hours of training for the "Visdrone" dataset are indicated on the horizontal axis.**

V. RESULTS

The results were validated using two networks with different training procedures. Firstly, a network trained using original images without being reduced or cropped and, secondly, a network trained using pre-processed images using the algorithm discussed in this paper.

The validation was conducted not to determine the quality of the model, since it was validated against the same dataset with which it was trained. We note that the purpose of this article is not to determine the success of the training itself but rather whether the algorithm succeeds in reducing training times without any loss of effectiveness. The results in themselves are not significant but the differences between results if the network is trained using a pre-processed dataset or the original. Thus, both training results were validated to compare them. The terms used in this comparison are:

- **Network A**: Network resulting from the training based on the original dataset.
- **Network B**: Network resulting from the training based on the pre-processed dataset generated using the algorithm discussed in this paper.

A. "DRONE" CASE

Both networks used a validation process against the original images, generating the confusion matrices shown in Figure 20.

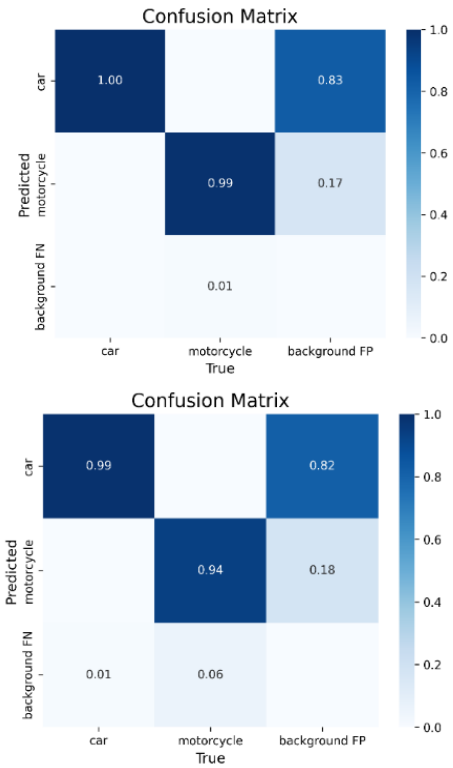

FIGURE 20. **Confusion matrices of the original (above) and pre-processed images (below).**

These matrices show, in the validation of Network B, that is, the network generated from pro-processed images, a slight increase in the number of "False Positives" especially in the type "car". But a closer analysis shows that this is not correct. In fact, the network has a higher success rate than the labelled original. In the original images, small and distant objects of interest are not labelled to avoid adding noise to the training process. In the training with the original images these objects are categorized correctly as true negatives, while with the cropped images these objects simply are not included in the training process (neither as true positives nor true negatives). But in validating the original images, these "true negatives" are detected as "true positives" by the network trained with the pre-processed dataset. That is, Network B has a greater sensibility to small, non-labelled objects, but positives, in the original images.

Figure 21 shows an original frame from the video without any labelled objects as these are very far from the camera. This image was analyzed by both neural networks (Network A and Network B). In the case of Network A, the objects were correctly learned as true negatives and were not marked (Figure 22.). But in the case of Network B, these distant objects were not inputted into the network, that is, they w ere

never marked as "selectable objects" and so were never marked as objects to be discarded as "true negative". Thus, in processing this image, Network B will detect these objects as a target if the resolution of the image permits.

FIGURE 21. Complete original image with not labelled objects as these are too far away.

FIGURE 22. Upper right corner amplification of Figure 21, where appear objects undetected by network A (trained with the original dataset).

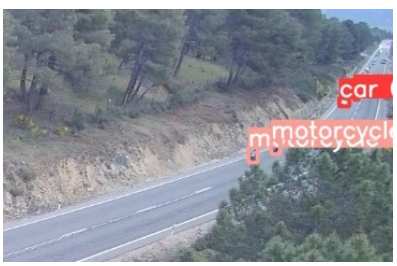

FIGURE 23. Upper right corner amplification of Figure 21, where appears objects detected by network B (trained with a pre-processed dataset).

Advantages obtained during training.

In line with the above, we found that both datasets produce a very similar trained network, even for this dataset. It may be said that the network generated using the pre-processed dataset is slightly better, detecting smaller objects of interest and with fewer false negatives.

Thus far, we have demonstrated that the training results are similar, the two networks are equivalent. But this is not the principal advantage of the algorithm which is the training process itself where better results are obtained.

B. "ROUNDABOUT" CASE

Both networks were used in a validation process against the original images, generating the Figure 24 confusion matrices.

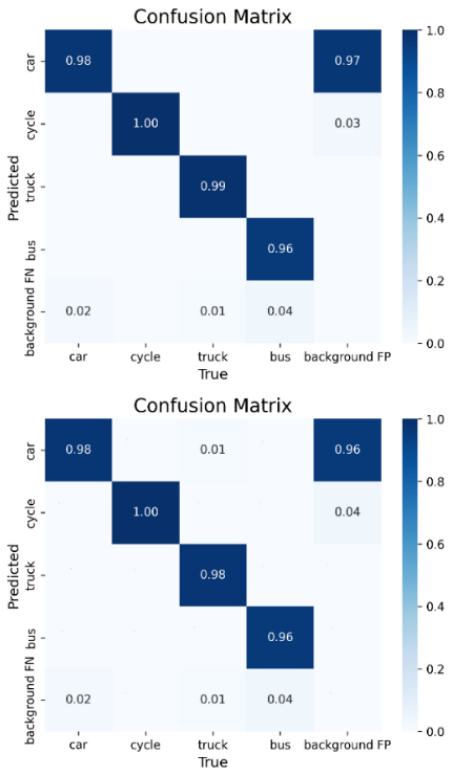

FIGURE 24. Confusion matrices of the original images (above) and the pre-processed images (below).

Advantages obtained during training.

For this dataset, consisting of 65K images in 2K quality, the training time using the YOLO algorithm and "Yolov5m" network for 30 epochs, was 3 days, 4 hours, and 3 minutes, while the training time using the same computer for the pre-processed dataset was 1 day, 8 hours and 46 minutes.

This is a perfect example of network training where the results are the virtually the same, with very little differences between them. The greatest difference, although minimum, is in the case of the label "car" where there was a slight confusion with "truck".

C. "VISDRONE" CASE

Both networks were validated using the original images, generating the confusion matrices shown in Figure 25.

Advantages obtained during training.

For this dataset, consisting of 33.6K images in FullHD quality, the training time using the YOLO algorithm and "Yolov5m" network for 30 epochs, was 14 hours and 26 minutes, while the training time using the same computer for the pre-processed dataset was 3 hours and 36 minutes.

Here it is important to note that this training exercise presented the largest differences, although these are not significant if we

consider that the network was not trained effectively. The results of the training process in both cases, for the original dataset and the pre-processed dataset, were approximately 0.3 in the mAP_0.5 metric, a very poor result.

We will explain the reasons for this poor performance although it is important to note that these results also validate the algorithm which is designed exclusively to reduce training times rather than improve the training process itself.

The reason for this poor training result is because the network was trained using values downloaded repository without any prior cleaning of the dataset. For this dataset, the labelled original (not using YOLO) includes special types and attributes. Thus, we have a 'type 0' to indicate "regions to ignore", see Figure 30, or attributes that indicate if the labelled object is hidden, as shown in Figure 27 and Figure 28, truncated or even confidence (score) of the labelled objects.

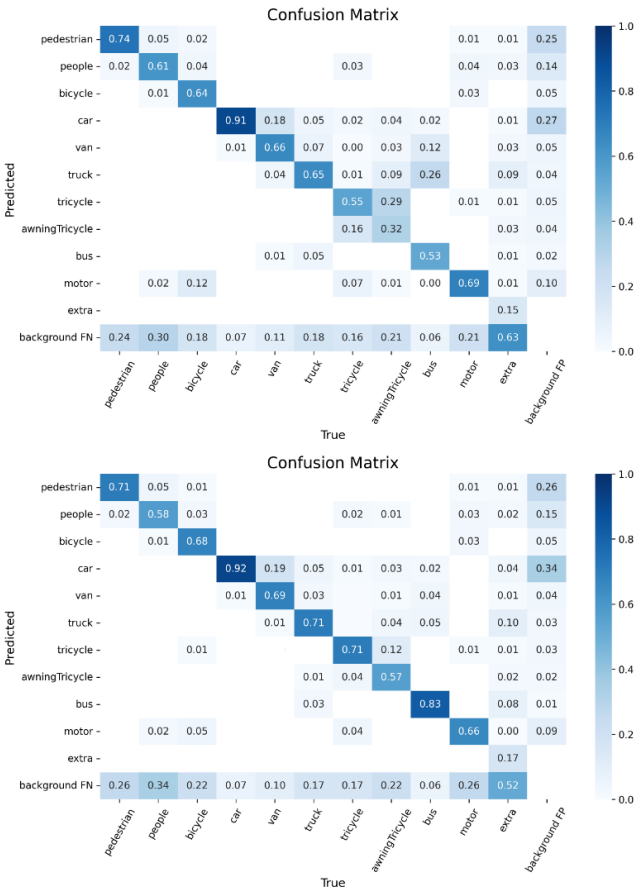

FIGURE 25. Confusion matrices of the original (above) and pre-processed images (below).

To improve the training results, it is essential that the dataset be initially cleaned and filtered of hidden objects, highly distorted or cut objects, and dubious labels, relabeling objects which are unlabeled but as perfectly recognizable in the images (see Figure 27, Figure 28, Figure 29 and Figure 30). This was not done here, firstly, because the purpose of this article is not to evaluate the quality of the training process of neural networks using known datasets but rather to evaluate the time reductions in training provided by the algorithm; secondly, a clean dataset with fewer labels can optimize the training process, thus, this is further evidence of the effectiveness of our pre-processing system. Regardless, the algorithm reduced the training time to one quarter of the original training time.

FIGURE 26. Sample frames from an uncleaned dataset.

This improvement in training times is particularly important given that the dataset in this case is not ideal for pre-processing. Figure 26 shows how the images do not meet some of the conditions for optimum effectiveness of the algorithm such as the lack of concentration of objects in a specific zone of the image. As can be seen, the labelled objects are distributed throughout the frame. In contrast, it does meet other parameters that allow the algorithm to be effective, such as the limited movement of objects between frames and many objects remaining immobile over many frames.

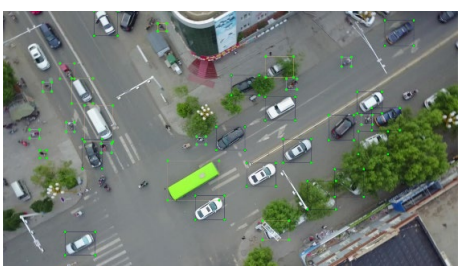

FIGURE 27. Sample frame from the labelled dataset.

FIGURE 28. Amplification of Fig. 27. showing totally hidden but labelled targets (cars).

FIGURE 29. Amplification of Figure 27 showing targets (cars and motorcycles) that are perfectly identifiable and not labelled in the dataset.

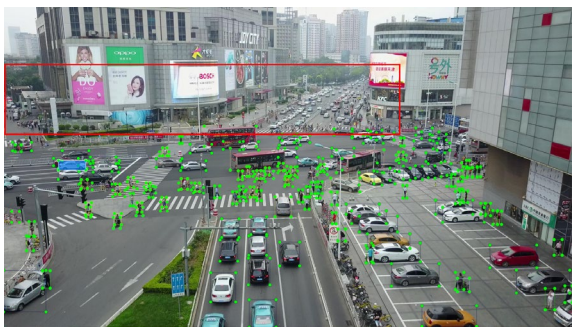

FIGURE 30. Sample frame from the labelled dataset. Here we see the upper part of the image is marked as not labelled (red box) while many objects can be perfectly recognized.

By contrast, these images demonstrate the poor results of the training which, while not a problem for pre-processing, should be taken into consideration. Certain objects are labelled but totally hidden (cars under trees, for example), mislabeled or unlabeled (motorcycles, for example). Oddly, these same motorcycles are labelled in other frames of the video. There are also zones of the image which are perfectly recognizable but marked as to be ignored.

VI. DISCUSION

How is it possible that partitioning an image into smaller images produces results which are inferior compared to the original? In other words, how is it possible that the dataset, in addition to being taken from smaller images, generates a set of smaller images?

The explanation is found in the first criterium for eliminating cut images. That is, in the discarding of cut images which only include objects of interest that do not move, for example parked cars. In many frames the only cars appearing are parked, with no other vehicles circulating. These cars are only labelled once in the "key" frame which the configuration established every 7 images (7 to 1 reduction).

FIGURE 31. Example of parked cars (in blue) and circulating cars (in red)

The result is that the pre-processed image is not only smaller but also more equal. A parked car will appear in all the frames of the video, giving it greater weight in the training process while a car moving in front of the camera only appears in the sequence of images for a few seconds. Thus, a false positive of an object appearing in all the images will be more highly penalized than a false positive of an object which is only labelled in 5 or 10 frames. This means the network can 'overlearn' some objects to the detriment of others.

It is important to note that the static objects of interest (parked cars, for example) are not only labelled once in the pre-processed dataset, as shown in Figure 31, discarding all other appearances because the object doesn't move, but are also labelled in every "key" image. By adjusting the configuration of this value in the algorithm the repercussion of static objects can be compensated, being very abundant in the dataset versus objects which appear only in a limited number of frames.

These two key points that the algorithm addresses primarily achieve:

- Reducing the dataset size in terms of storage space by 20%. As mentioned earlier, the original and processed image sets are not vastly different. In our tests, in the worst case, it doesn't even double the number of images. However, these images are much smaller, going from around 1.5MB (in jpg format) per original image to about 100KB per processed image. This translates to a significant reduction. It's worth noting that the labeling file size is negligible in these calculations, as it accounts for less than 0.01% of the total dataset size.
- With smaller images, a larger number of images can be loaded in parallel into the memory of the graphics cards. In our case, we were able to go from loading 4 images in parallel to loading 42 images. This makes the training process more efficient.

The consequence of these two points results in training times around 20% of the original time, with insignificant variation in the quality of the trained network. Sometimes, it even performs better than the original by avoiding overfitting in datasets with imbalanced and low-quality targets.

VII. CONCLUSIONS

An analysis of the results shows that the image pre-processing algorithm is more efficient in terms of time and computation, able to be executed using standard equipment without any outstanding characteristics. Additionally, very significant improvements were seen in training times with reductions from at least 50% to, depending on the dataset, reduction of 80%. If, for example, we focus on a success score of 0.95 in the mAP_0.5 metric, very significant time reductions were achieved, as shown in Figure 32:

- Drone Dataset. Training without improvement: 3 hours and 36 minutes, with pre-processing: 30 minutes. A reduction in training time of 87%.
- Roundabout Dataset. Training without improvement: 21 hours and 11 minutes, with pre-processing: 6 hours and 34 minutes. A reduction in training time of 72%.
- Visdrone Dataset. A success score of 0.95 for the metric was never achieved for this dataset. The highest success score was in epoch 9, after 4 hours and 5 minutes for the original dataset and 1 hour for the pre-processed dataset. A reduction in training time of 76%.

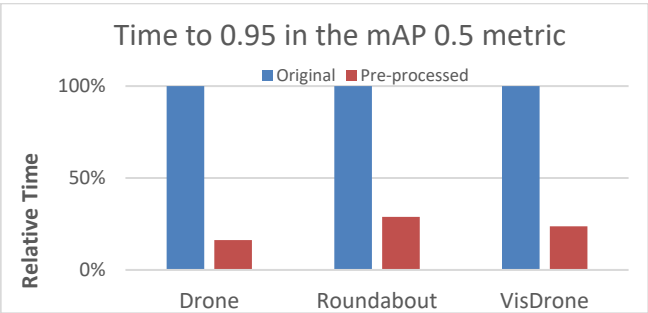

FIGURE 32. Comparison of time in achieving a score of 0.95 in the mAP_0.5 metric.

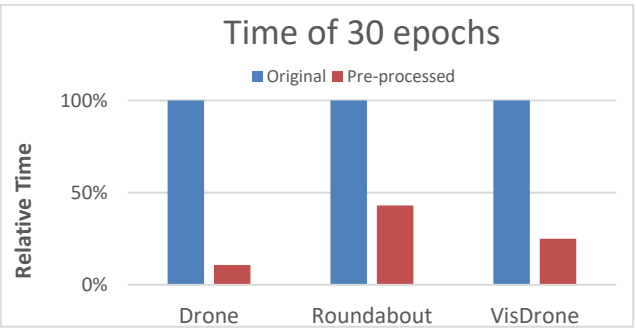

FIGURE 33. Comparison of time in the training of 30 epochs.

As shown in Figure 33, similar results can be obtained if the aim is simply a specific number of epochs.

Additionally, it was found that pre-processing does not alter the quality of the training. If the dataset is clean or well formatted, the training is successful in both cases, as seen in the Drone and Roundabout datasets while, if the dataset is not well labelled, the network trains with the same failures as with the original.

To conclude, it is important to note the added benefit that a network trained with a pre-processed dataset tends to be more precise in distant, unlabeled objects, as can be seen in Fig. 5, Figure 21 and Figure 22. In the complete images these objects are trained as true negatives while in the pre-processed network these objects are not part of the training. Thus, these objects are detected in the image during the training process, but in the validation, they are detected as false positives since these are not marked in the original dataset.

ACKNOWLEDGMENT

The authors would like to thank the Universidad Francisco de Vitoria and the Universidad Europea de Madrid for their support. They are especially grateful to the translation service of the Universidad Francisco de Vitoria for their help in translating and revising the manuscript.

REFERENCES

[1] Kovalev, Vassili & Kalinovsky, Alexander & Liauchuk, Vitali. (2016). Deep Learning in Big Image Data: Histology Image Classification for Breast Cancer Diagnosis.
[2] Strubell, E. & Ganesh, A. & and McCallum, A. (2019). Energy and Policy Considerations for Deep Learning in NLP.
[3] H. Mao, S. Yao, T. Tang, B. Li, J. Yao and Y. Wang, "Towards Real-Time Object Detection on Embedded Systems," in IEEE Transactions on Emerging Topics in Computing, vol. 6, no. 3, pp. 417-431, 1 July-Sept. 2018, doi: 10.1109/TETC.2016.2593643.
[4] Jose A. Carballo & Javier Bonilla & Manuel Berenguel & Jesús Fernández-Reche & Ginés García (2019). New approach for solar tracking systems based on computer vision, low cost hardware and deep learning,
[5] Moons, B. & Bankman, D. & Verhelst, M. (2019). Embedded deep learning. Embedded Deep Learning.
[6] K. Rungsuptaweekoon, V. Visoottiviseth and R. Takano, "Evaluating the power efficiency of deep learning inference on embedded GPU systems," 2017 2nd International Conference on Information Technology (INCIT), 2017, pp. 1-5, doi: 10.1109/INCIT.2017.8257866.
[7] Plastiras, George & Kyrkou, Christos & Theocharides, Theo. (2019). Efficient ConvNet-based Object Detection for Unmanned Aerial Vehicles by Selective Tile Processing.
[8] Rukundo, O (2021). Effects of Image Size on Deep Learning. Available online: https://arxiv.org/abs/2101.11508v6 (accessed on 1 februry 2023).
[9] Sabottke, C. F., & Spieler, B. M. (2020). The effect of image resolution on deep learning in radiography. Radiology: Artificial Intelligence, 2(1), e190015.
[10] Sai Wu, Mengdan Zhang, Gang Chen, and Ke Chen. 2017. A New Approach to Compute CNNs for Extremely Large Images. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17). Association for Computing Machinery, New York, NY, USA, 39–48. https://doi.org/10.1145/3132847.3132872
[11] Aravind Ramalingam (2021). How to Pick the Optimal Image Size for Training Convolution Neural Network?
[12] Lakhani, P. (2020). The importance of image resolution in building deep learning models for medical imaging. Radiology: Artificial Intelligence, 2(1), e190177.
[13] Reina G. Anthony & Panchumarthy Ravi & Thakur Siddhesh Pravin & Bastidas Alexei & Bakas Spyridon (2020). Systematic Evaluation of Image Tiling Adverse Effects on Deep Learning Semantic Segmentation
[14] Angus Lang Sun Lee & Curtis Chun Kit To & Alfred Lok Hang Lee & Joshua Jing Xi Li, Ronald Cheong Kin Chan (2022). Model architecture and tile size selection for convolutional neural network training for non-small cell lung cancer detection on whole slide images.
[15] Kang Tong & Yiquan Wu (2022). Deep learning-based detection from the perspective of small or tiny objects: A survey. https://doi.org/10.1016/j.imavis.2022.104471
[16] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[17] Fernández Cordeiro, L. (2019). Desarrollo de dataset personalizado para entrenamiento de YOLO como sistema de detección de objetos en tiempo real, para entorno con brazo robot.
[18] Zhao, X., Ni, Y., Jia, H. (2017). Modified Object Detection Method Based on YOLO. In: , et al. Computer Vision. CCCV 2017. Communications in Computer and Information Science, vol 773. Springer, Singapore. https://doi.org/10.1007/978-981-10-7305-2_21
[19] Joseph Redmon & Santosh Divvala & Ross Girshick & Ali Farhadi (2016). Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788
[20] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.
[21] Bemposta Rosende, S.; Ghisler, S.; Fernández-Andrés, J.; Sánchez-Soriano, J. Dataset: Traffic Images Captured from UAVs for Use in Training Machine Vision Algorithms for Traffic Management. Data 2022, 7, 53. https://doi.org/10.3390/data7050053
[22] Puertas, E.; De-Las-Heras, G.; Fernández-Andrés, J.; Sánchez-Soriano, J. Dataset: Roundabout Aerial Images for Vehicle Detection. Data 2022, 7, 47. https://doi.org/10.3390/data7040047
[23] P. Zhu, L. Wen, D. Du, X. Bian, H. Fan, Q. Hu, et al., "Detection and tracking meet drones challenge", IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1-1, 2021.

SERGIO BEMPOSTA ROSENDE received a degree in Computer Engineering from the European University of Madrid in 2002. He has a master's degree in Big Data Analytics from the European University of Madrid in 2018. In 2004 he joined the Computer Systems Department of the European University of Madrid as an associate professor, where he is associate professor. His research interests include robotics, drones, machine learning, computer vision and intelligent transportation systems.

JAVIER FERNÁNDEZ-ANDRÉS received a degree in Industrial Engineering from the Polytechnic University of Madrid, Spain in 1992 and a PhD in Robotics and Computer Vision from the Polytechnic University of Madrid, Spain in 1998. In 1998 he joined the Computer Systems Department of the European University of Madrid as an associate professor, where he has been a professor until 2004. From 2004 to 2012 he has been Chairman of the Department of Computer Systems and Automation. Since 2012 he is Full Professor in the Department of Engineering at the European University of Madrid. His research interests include computer vision, intelligent transportation systems, pattern recognition and machine learning.

JAVIER SÁNCHEZ-SORIANO received a degree in Computer Engineering from the Polytechnic University of Madrid, Spain in 2009 and master's degree in information technologies from the Polytechnic University of Madrid, Spain in 2010. He has a PhD in Artificial Intelligence from the Polytechnic University of Madrid, Spain in 2016. In 2012 he joined the Computer Systems Department of the European University of Madrid as an associate professor, where he has been a professor until 2022. Since 2022 he has been associate professor at the Polytechnic School of the Universidad Francisco de Vitoria. His research interests include machine learning, autonomous driving, computer vision and intelligent transportation systems.